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 CONTROLLER

Plugin Function

OPERATOR'S MANUAL

A-97606-6419EN/12

© FANUC CORPORATION, 2024

Translation of the original instruction
The original language of this manual is English. The other language manuals are a translation from the English one.

Thank you very much for purchasing FANUC Robot.
Before using the Robot, be sure to read the "FANUC Robot series SAFETY HANDBOOK (B-80687EN)" and understand the content.

- No part of this manual may be reproduced in any form.
- All specifications and designs are subject to change without notice.

The products in this manual are controlled based on Japan's "Foreign Exchange and Foreign Trade Law". The export from Japan may be subject to an export license by the government of Japan.
Further, re-export to another country may be subject to the license of the government of the country from where the product is re-exported. Furthermore, the product may also be controlled by re-export regulations of the United States government.
Should you wish to export or re-export these products, please contact FANUC for advice.

In this manual, we endeavor to include all pertinent matters. There are, however, a very large number of operations that must not or cannot be performed, and if the manual contained them all, it would be enormous in volume. It is, therefore, requested to assume that any operations that are not explicitly described as being possible are "not possible".

SAFETY PRECAUTIONS

This chapter describes the precautions which must be followed to enable the safe use of the robot. Before using the robot, be sure to read this chapter thoroughly.

For detailed functions of the robot operation, read the relevant operator's manual to understand fully its specification.

For the safety of the operator and the system, follow all safety precautions when operating a robot and its peripheral equipment installed in a work cell.
For safe use of FANUC robots, you must read and follow the instructions in the “FANUC Robot series SAFETY HANDBOOK (**B-80687EN)**”.

1 PERSONNEL

Personnel can be classified as follows.

Operator:
- Turns the robot controller power ON/OFF
- Starts the robot program from operator panel

Programmer or Teaching operator:
- Operates the robot
- Teaches the robot inside the safeguarded space

Maintenance technician:
- Operates the robot
- Teaches the robot inside the safeguarded space
- Performs maintenance (repair, adjustment, replacement)

- The operator is not allowed to work in the safeguarded space.
- The programmer or teaching operator and maintenance technician are allowed to work in the safeguarded space.
- Based on the "Ordinance on Industrial Safety and Health" in Japan or laws in each country, to work inside the robot set safeguarded space, the followings is required:
  - Perform risk assessment for the entire robot system.
  - Based on the risk assessment result, all personnel must take appropriate professional safety training courses.

Table 1 (a) lists the work outside the safeguarded space. In this table, the symbol "O" means the work allowed to be carried out by the specified personnel.

Table 1 (a) List of work outside the Safeguarded Space

	Operator	Programmer or Teaching operator	Maintenance technician
Turn power ON/OFF to Robot controller	O	O	O
Select operating mode (AUTO/T1/T2)		O	O
Select remote/local mode		O	O
Select robot program with teach pendant		O	O
Select robot program with external device		O	O
Start robot program with operator's panel	O	O	O
Start robot program with teach pendant		O	O
Reset alarm with operator's panel		O	O
Reset alarm with teach pendant		O	O
Set data on teach pendant		O	O
Teaching with teach pendant		O	O
Emergency stop with operator's panel	O	O	O
Emergency stop with teach pendant	O	O	O
Operator's panel maintenance			O
Teach pendant maintenance			O

During robot operation, programming and maintenance, the operator, programmer, teaching operator and maintenance technician take care of their safety using at least the following safety protectors:

- Use clothes, uniform, overall adequate for the work
- Safety shoes
- Helmet

2 DEFINITION OF SAFETY NOTATIONS

To ensure the safety of users and prevent damage to the machine, this manual indicates each precaution on safety with "**WARNING**" or "**CAUTION**" according to its severity. Supplementary information is indicated by "**NOTE**". Read the contents of each "**WARNING**", "**CAUTION**" and "**NOTE**" before using the robot.

Symbol	Definitions
⚠**WARNING**	Used if a danger resulting in the death or serious injury of users are expected to occur if they fail to observe the approved procedure.
⚠**CAUTION**	Used if a danger resulting in the minor or moderate injury of users or equipment damage is expected to occur if they fail to observe the approved procedure.
NOTE	Used if a supplementary explanation not related to any of WARNING and CAUTION is to be indicated.

TABLE OF CONTENTS

SAFETY PRECAUTIONS.....s-1

1 OVERVIEW..... 1
1.1 FEATURES FOR END USERS..... 1
1.2 FEATURES FOR PLUGIN DEVELOPERS..... 3
1.3 LIMITATIONS OF PLUGIN FUNCTION..... 4
2 PLUGIN SOFTWARE DEVELOPMENT ENVIRONMENT..... 5
2.1 CREATION AND TRANSLATION OF COMPONENT FILES..... 5
2.2 SDK FOR PLUGIN SOFTWARE DEVELOPMENT..... 5
3 INSTLLATION AND MANAGEMENT OF PLUGINS..... 7
3.1 PLUGIN INSTALL SCREEN..... 7
3.2 PLUGIN LIST SCREEN..... 9
4 FILES IN INSTALLATION PACKAGE..... 12
5 INSTALLATION DEFINITION FILE..... 14
5.1 FORMAT OF INSTALLATION DEFINITION FILE..... 14
5.2 COMMANDS FOR SETTING UP SYSTEM..... 21
5.3 OVERVIEW OF UNINSTALL OPERATION..... 34
6 PLUGIN DEVELOPMENT OVERVIEW..... 36
6.1 KAREL VARIABLES REFERENCED IN PLUGIN SCREEN..... 36
6.2 PLUGIN SCREENS IMPLEMENTATION..... 37
6.3 DICTIONARY FILES IMPLEMENTATION..... 38
6.4 EXAMPLE OF PLUGIN SCREEN..... 38
7 PLUGIN SCREEN CREATION..... 40
7.1 BASIC INFORMATION..... 40
7.2 BUILD TOOL..... 41
7.3 USING iHMI COMPONENTS..... 44
7.4 IHMI COMPONENTS..... 48
7.5 JAVASCRIPT APIS TO INTERFACE WITH ROBOT CONTROLLER..... 71
7.6 DATA TYPES IN top.frpl FUNCTIONS..... 119
7.7 ADD MENUS..... 124
7.8 VERIFICATION OF CREATED SCREEN USING ROBOGUIDE..... 127
8 CREATION OF PLUGIN INSTRUCTIONS..... 129
8.1 CONSTRUCTION FILES..... 129
8.2 PLUGIN INSTRUCTION DEFINITION FILE..... 130
8.3 PROGRAM CALLED WHEN PLUGIN INSTRUCTION IS EXECUTED..... 135
8.4 CREATION OF ICON FOR PLUGIN INSTRUCTION..... 146
8.5 CREATING DETAILS SCREEN OF PLUGIN INSTRUCTION..... 146
8.6 ERROR LIST FOR PLUGIN INSTRUCTION..... 149
9 ERROR MESSAGES AT INSTALLATION..... 150
9.1 LIST OF ERROR MESSAGES..... 150

10 TOOL I/F (CRX series only)·····152
10.1 TOOL I/F SPECIFICATIONS ····· 153
11 TOOL I/F COMMUNICATION (CRX series only)·····157
11.1 COMMON SETTINGS FOR TOOL I/F COMMUNICATION ····· 158
11.2 SETTINGS FOR DIGITAL OUTPUT SIGNAL ····· 159
11.3 SETTINGS FOR ANALOG INPUT SIGNAL ····· 160
11.4 SETTINGS FOR SERIAL COMMUNICATION ····· 163
11.5 KAREL BUILT-IN FOR SERIAL COMMUNICATION ····· 164
11.6 SAMPLE PROGRAM OF SERIAL COMMUNICATION····· 165
11.7 ALARM LIST ····· 169
12 TOOL CHANGE FUNCTION (CRX series only)·····171
12.1 MANUAL TOOL CHANGE SCREEN ····· 172
12.2 TOOL DETACH / ATTACH ICON INSTRUCTION····· 174
12.3 RESTRICTIONS OF TOOL CHANGE FUNCTION ····· 176
12.4 ALARM LIST FOR TOOL CHANGE FUNCTION····· 177
12.5 INFORMATION FOR PLUGIN SOFTWARE DEVELOPERS ····· 178
13 PERIPHERAL DEVICE INTERFACE ON R-30iB MINI PLUS ·····186
13.1 PERIPHERAL DEVIDE INTERFACE (JRM18) ····· 187
13.2 CONNECTION BETWEEN MAIN BOARD (JRM18) AND PERIPHERAL DEVICES 189
13.3 CONNECTION BETWEEN MAIN BOARD (JRM18) AND PERIPHERAL DEVICES (FOR TERMINAL CONVERSION MODULE) ····· 193
13.4 DIGITAL I/O SIGNAL SPECIFICATIONS OF PERIPHERAL DEVICES··· 198
13.5 24V POWER SUPPLY FOR EXTERNAL ····· 201
14 ECLIPSE DEVELOPMENT ENVIRONMENT ·····202
14.1 OPERATING ENVIRONMENT ····· 202
14.2 SETTINGS ECLIPSE PLUGINS····· 203
14.3 KAREL DEVELOPMENT ENVIRONMENT ····· 204
14.4 iHMI COMPONENT DEVELOPMENT ENVIRONMENT ····· 218
15 CRX SERIES MECHANICAL UNIT·····223
15.1 SPECIFICATIONS····· 223

1 OVERVIEW

The Plugin Function provides the way for the end user to load the "Plugin" software, which is provided by a 3rd party supplier, to a FANUC robot. Together with the standardized mechanical and electrical interface, this feature makes the connection and installation of peripheral equipment simple and easy. This manual provides all information that is needed to develop the plugin software.

The figure 1 (a) shows the overview of the plugin function. This uses a gripper as an example, but the plugin can be developed for any application, such as sensor system, vision system, and so forth.

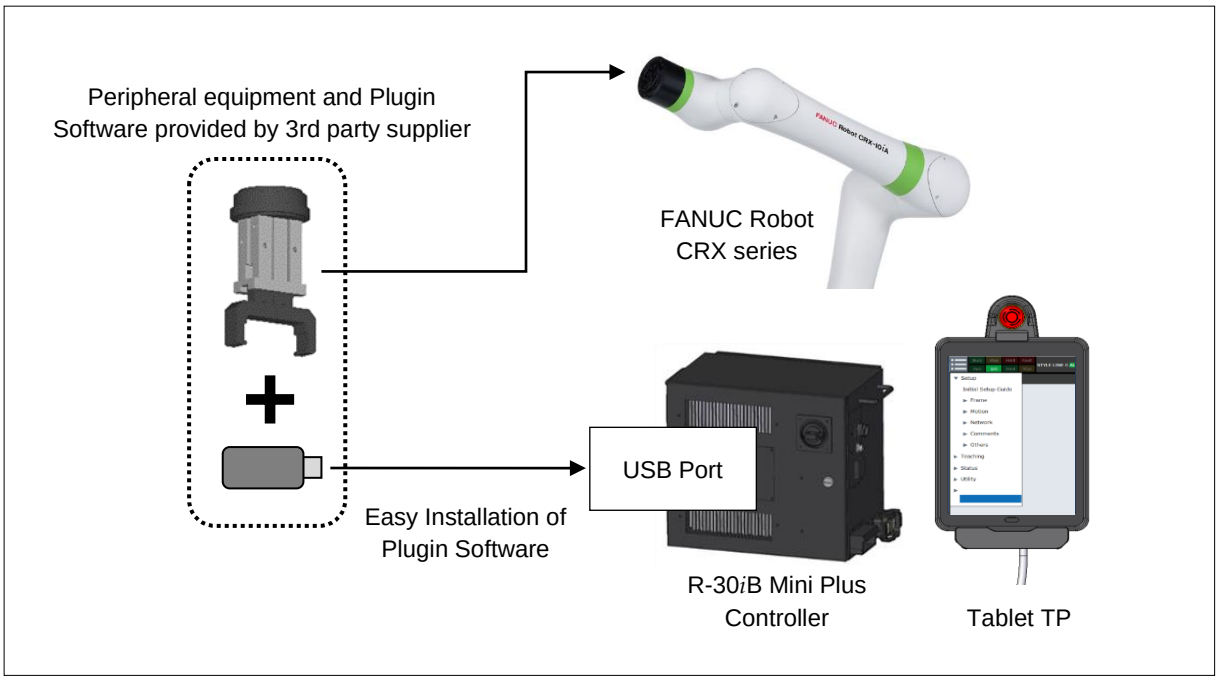


Fig 1 (a) Overview of plugin function

1.1 FEATURES FOR END USERS

The feature provides the following two screens for the end users. Installation of all plugin software and the management can be done in these screens.

- **Plugin Install screen**
  This screen enables the end user to install the plugin software provided from 3rd party supplier using a USB memory or FTP client devices.

- **Plugin List screen**
  This screen shows a list of the installed plugins and detail profile information. This screen also let you uninstall the installed plugin.

⚠ CAUTION
1 When connect the peripheral device compatible to Plugin, please ensure to complete the installation of plugin software before connecting the device to EE connector. If you connect the device before installation, the device might be broken.
2 When insert or remove the peripheral device connected to the EE connector,

please ensure to turn off the robot controller.
3 When you install plugin software, please be sure to put plugin software (*.ipl file) on the root directory of the USB memory or FTP client devices.

1.1.1 Installation of Plugin Software

A plugin software that has been developed by a 3rd party supplier can be easily installed using a USB memory or FTP client devices. Installation of plugin software is done in the Install screen. The screen shows the detail information of the plugin software to be installed, such as name of the provider of the plugin, the name of the equipment supported, the plugin software version, etc. By installing a plugin software, new screens specifically for the equipment and new program instructions to control the equipment may be added, and also the system may be set up for the equipment.

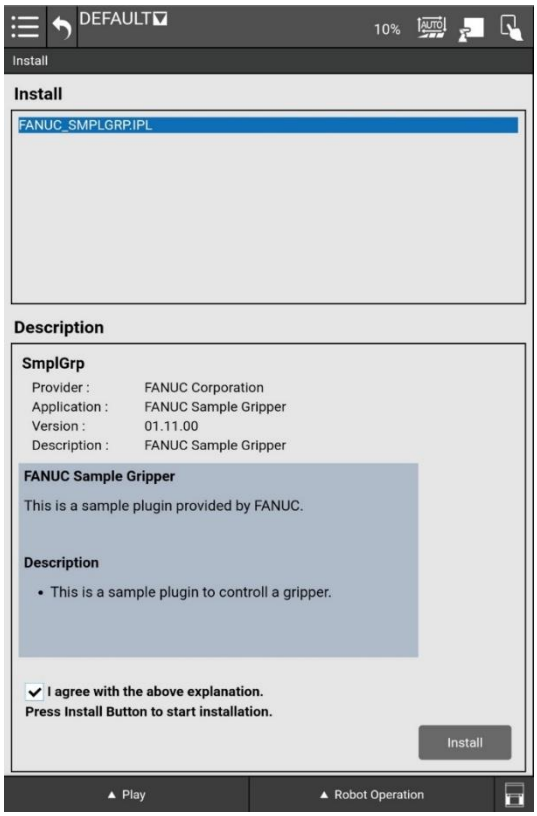


Fig 1.1.1 (a) Plugin Install Screen

NOTE
1 You can install a plugin software through UD1, UT1 or C1 to C8 device in the plugin installation menu. Note that the plugin software installation through UT1 or C1 to C8 device requires the robot software version V9.40P/58 or after.
2 When you use C1 to C8, please setup the FTP client devices correctly and confirm that the files in those devices can be shown in FILE screen. For setting up client devices, refer to the FTP OPERATIONS chapter of “Ethernet Function OPERATOR’S MANUAL (B-82974EN)”.
3 Only client devices with active mode are supported in this function.
4 Please use a Tablet TP to install plugin software through UT1. When you use iRProgrammer on a PC, a plugin software in UT1 will not be displayed on the screen.

1.1.2 Management of Plugin Software

The Plugin List Screen shows a list of installed plugins with the detail information, such as provider name, name of the peripheral equipment supported, plugin software version, etc. This screen also lets you uninstall the selected plugin by tapping the “Uninstall” button on the bottom right.

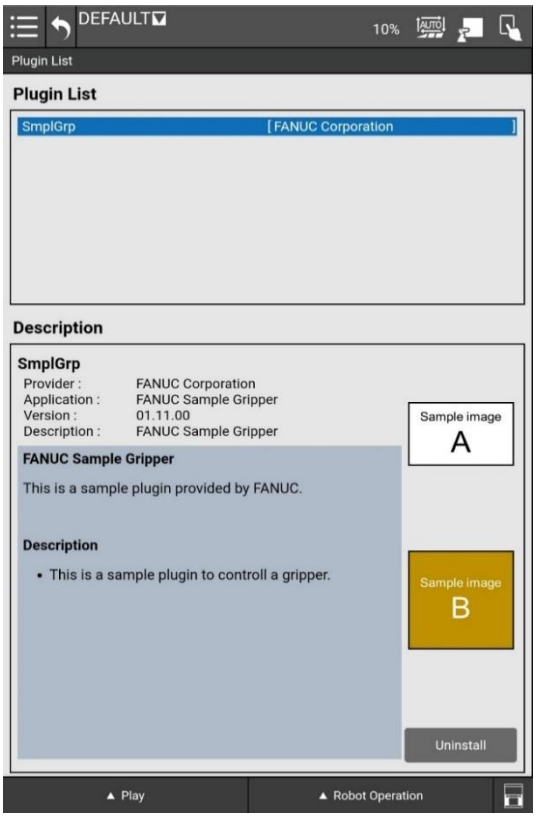


Fig 1.1.2 (a) Plugin List screen

1.2 FEATURES FOR PLUGIN DEVELOPERS

The providers of the plugin software can develop a plugin screens specifically for the equipment to do the setting up or status monitoring, and a plugin program instruction to control the equipment.

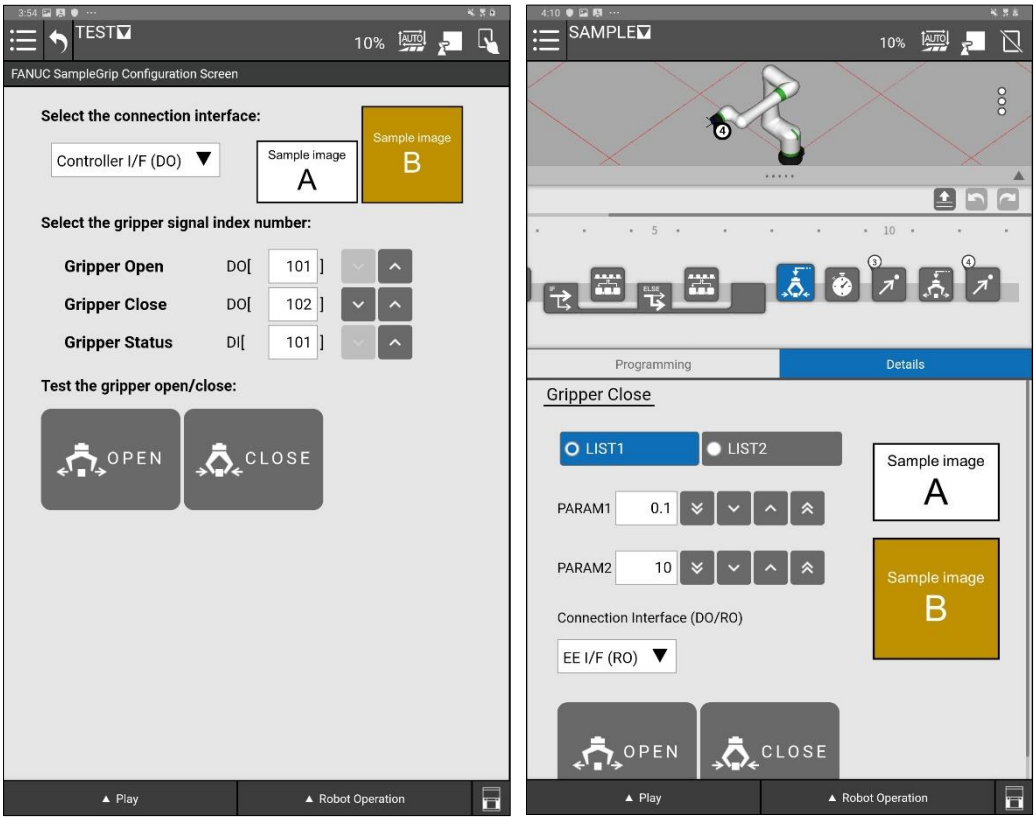


Fig 1.2 (a) Example of Plugin Software Development (Left: a plugin screen / Right: a plugin instruction)

Since the Tablet TP (Teach Pendant) uses the HTML based screens on the Web browser, one with general knowledge of Web (HTML, JavaScript, CSS) can develop a screen for the tablet. With the iHMI components that FANUC provides, the development of iHMI screens can be even easier without having to create the HTML, JavaScript and CSS codes from scratch.

A program instruction may also be developed to control the peripheral equipment such as a gripper. The end user may use the new instruction by dragging an icon into the user program as shown on the right on Figure 1.2 (a). A graphic screen may also be developed to set the parameters to pass to the instruction. An instruction may be developed as a TP program, or a KAREL program as necessary for complicated processing.

The following chapters of this manual explains the detail of the development of plugin software and its package.

1.3 LIMITATIONS OF PLUGIN FUNCTION

The following limitations apply to Plugin Function.

- Tablet TP is required for this feature.
- This feature is only available for V9.40 (7DF5) software series.

2 PLUGIN SOFTWARE DEVELOPMENT ENVIRONMENT

This chapter describes the plugin software development environment.

2.1 CREATION AND TRANSLATION OF COMPONENT FILES

This section describes the development environment for creating and translating the component files in a plugin installation package.

- TP program

You may create TP programs on ROBOGUIDE or an actual CRX controller (R-30iB Mini Plus controller). Please see chapter 8 “Creation of Plugin Instructions” for more information.

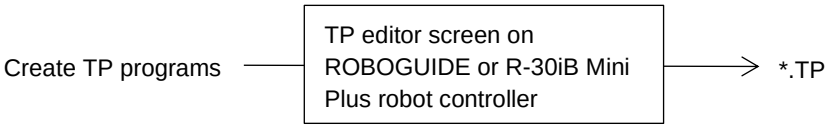


Fig 2.1 (a) TP programs creation

- KAREL program

You may develop and translate KAREL programs on ROBOGUIDE or Eclipse development environment (KAREL development environment). Please see chapter 13 “Eclipse development environment” for more information.

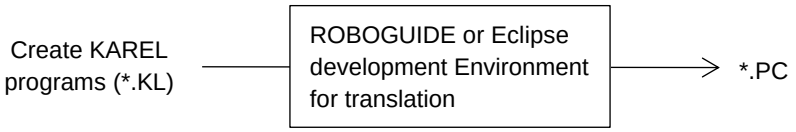


Fig 2.1 (b) KAREL programs development

- Screen data for details screen of plugin instructions and/or plugin screen, and menu/instruction definition file

You may translate the screen data and definition files on the build tools (ihmi_build) or Eclipse development environment (iHMI component development environment). Please see chapter 7 “Plugin Screen creation” for the build tools, and chapter 13 “Eclipse development environment” for iHMI component environment.

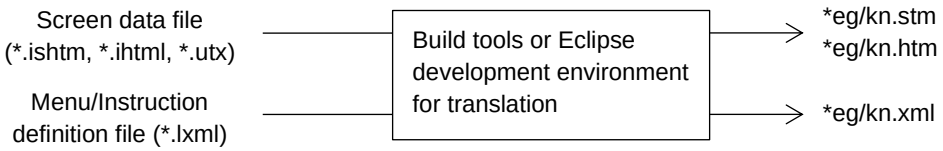


Fig 2.1 (c) Translation of the screen data and definition files

2.2 SDK FOR PLUGIN SOFTWARE DEVELOPMENT

The SDK (Software Development Kit) for plugin software development consists of the following files.

Table 2.2 (a) Contents of the plugin software development SDK

Contents		Description	Chapter
FANUC_SmplGrp	This folder contains a working example of the plugin installation package as well as its source code before building. This folder consists of the following files/folders.		
	output	This folder contains the contents of FANUC_SmplGrp.ipl. The files are packaged to an archive file with a name “(Arbitrary name).ipl” to be installed in the robot controller.	Chapter 4
	kl_ls	This folder contains the source files of TP programs in ASCII format and KAREL program files used in the sample plugin (FANUC_SmplGrp).	Chapter 6, 8
	src	This folder contains the source files that can be processed using the build tools (ihmi_build).	Chapter 7
	FANUC_SmplGrp.ipl	This is the sample plugin package. You may install this on ROBOGUIDE or an actual CRX robot to see how it works. Please note that this sample plugin is not for controlling any specific tool.	Chapter 3
fanuc_support		This folder contains the Eclipse plugins to assist with the plugin software development, although it is possible to develop the plugin software without using the Eclipse development environment. With this Eclipse plugins, KAREL and iHMI components development environment are available.	Chapter 13
ihmi_build		This folder contains the build tools for iHMI component. The iHMI component is UI library for screen development provided by FANUC to make web page development simpler and easier, although it is possible to create a web page without using iHMI component. If you have the iHMI component, you can develop a web page more easily without having to create all the codes in HTML, JavaScript, and CSS.	Chapter 7
A-97606-6419EN/JA_**.pdf		Plugin software development instruction manual (this document)	-
FANUC_Dev001_Enable_DevMode.ipl		This is a plugin installation package to activate developer mode on ROBOGUIDE or CRX contorller environment. • Installing this plugin allows editing the arguments of a plugin instruction on the conventional iPendant editor screen mode on V9.40P/18 or after. Once the plugin is uninstalled, this feature is disabled.	-

3 INSTLLATION AND MANAGEMENT OF PLUGINS

The feature provides the following two screens for the end users. Installation of all plugin software and the management can be done in these screens.

- **Plugin Install screen**
  This screen enables the end user to install the plugin software provided from 3rd party supplier using a USB memory or FTP client devices.

- **Plugin List screen**
  This screen shows a list of the installed plugins and detail profile information. This screen also let you uninstall the installed plugin.

⚠ CAUTION
1 When connect the peripheral device compatible to Plugin, please ensure to complete the installation of plugin software before connecting the device to EE connector. If you connect the device before installation, the device might be broken.
2 When insert or remove the peripheral device connected to the EE connector, please ensure to turn off the robot controller.
3 When you install plugin software, please be sure to put plugin software (*.ipl file) on the root directory of the USB memory or FTP client devices.

3.1 PLUGIN INSTALL SCREEN

A plugin software that has been developed by a 3rd party supplier can be easily installed using a USB memory or FTP client devices. Installation of plugin software is done in the Install screen. The screen shows the detail information of the plugin software to be installed, such as name of the provider of the plugin, the name of the equipment supported, the plugin software version, etc. By installing a plugin software, new screens specifically for the equipment and new program instructions to control the equipment may be added, and also the system may be set up for the equipment.

NOTE
1 You can install a plugin software through UD1, UT1 or C1 to C8 device in the plugin installation menu. Note that the plugin software installation through UT1 or C1 to C8 device requires the robot software version V9.40P/58 or after.
2 When you use C1 to C8, please setup the FTP client devices correctly and confirm that the files in those devices can be shown in FILE screen. For setting up client devices, refer to the FTP OPERATIONS chapter of “Ethernet Function OPERATOR’S MANUAL (B-82974EN)”.
3 Only client devices with active mode are supported in this function.
4 Please use a Tablet TP to install plugin software through UT1. When you use iRProgrammer on a PC, a plugin software in UT1 will not be displayed on the screen.

Procedure 3-1 Installing a Plugin Software

Procedure

1 Tap the menu icon on the status bar, and you will see the menu similar to the following.

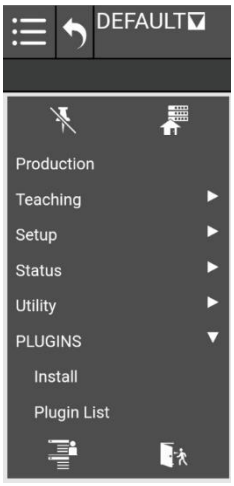


Fig 3.1 (a) Menu displayed when the menu icon is tapped

2 Tap the “Install” item under “Plugins”, and you will see the Install screen. The screen shows the information that is stored in the USB memory (UD1), the internal memory unique to the Tablet TP (UT1), and the FTP client devices (C1 to C8), such as the name of the plugin install package file, and the detail information of the plugin software. Please refer to section “USB MEMORY OPERATION (TABLET TP APP V1.12 OR LATER)” of the TABLET UI OPERATOR’S MANUAL (B-84274EN) regarding how to use the internal memory unique to the Tablet TP (UT1).

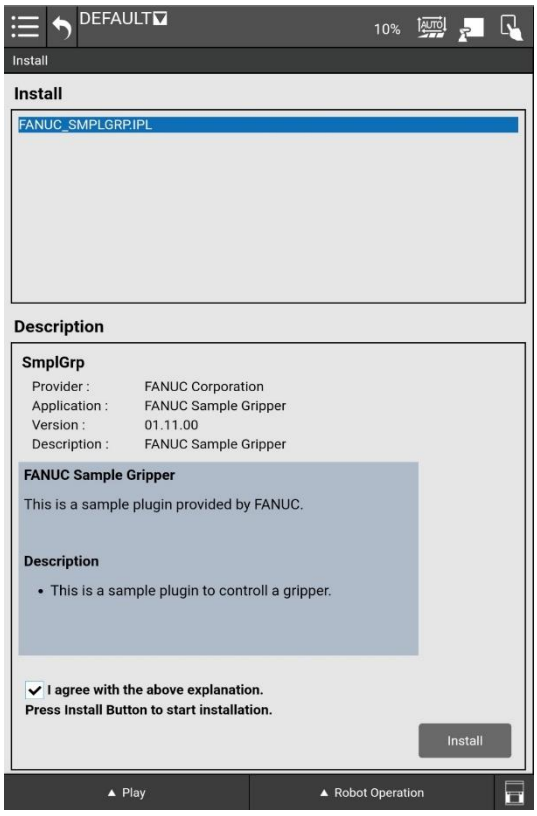


Fig 3.1 (b) Plugin Install Screen

3 Tap the “Install” button at the bottom right of the screen to start the installation process.

4 The following screen will be displayed when the installation processing is done.



Fig 3.1 (c) Screen showing Installation completion

5 Please re-start the controller by manually shutting down the controller power.

3.2 PLUGIN LIST SCREEN

The Plugin List Screen shows a list of installed plugins with the detail information, such as provider name, name of the peripheral equipment supported, plugin software version, etc. This screen also lets you uninstall the selected plugin by tapping the “Uninstall” button on the bottom right.

Procedure 3-2 Displaying a List of Installed Plugin Software

Procedure

1 Tap the menu icon on the status bar, and you will see the menu similar to the following.

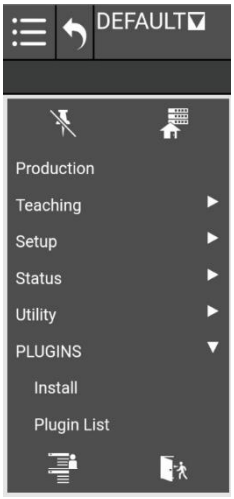


Fig 3.2 (a) Menu displayed when the menu icon is tapped

2 Tap the “Plugin List” item under “Plugins”, and you will see the Plugin List screen.

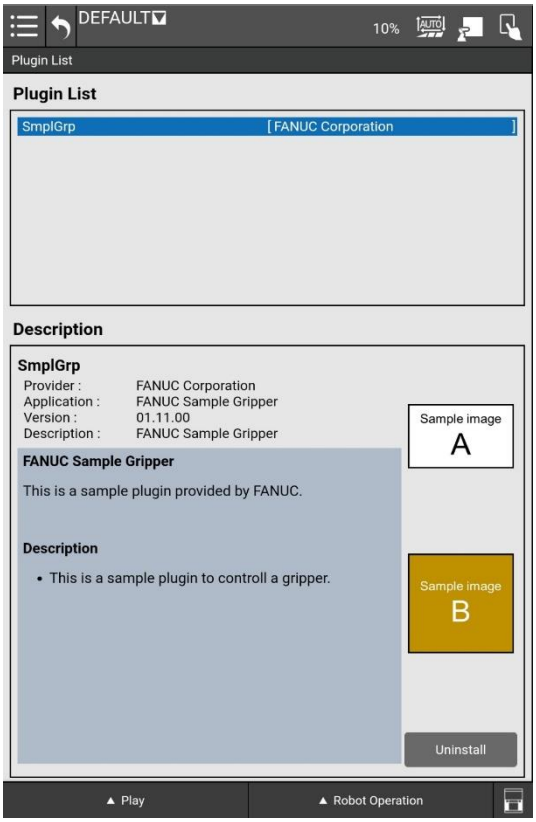


Fig 3.2 (b) Plugin List screen

3 Tap an item in the list to select a plugin software, and you will see the detail information of the selected plugin in the lower half of the screen.

4 You can uninstall the selected plugin by tapping the “Uninstall” button on the bottom right of the screen.

5 The following screen will be displayed when the uninstallation processing is done. Please re-start the controller by manually shutting down the controller power.



Fig 3.2 (c) Completion of Uninstall Operation

4 FILES IN INSTALLATION PACKAGE

An installation package consists of the files shown in the following table. The files in the table are packaged to an archive file, with a file extension of ".IPL" to constitute an installation package. The package file may be provided to the end user by various media such as USB memory, or being downloaded from the company web site.

Table 4 (a) Example of Files in an Installation Package

File	Name	Extension	Purpose	Chapter
Installation definition file	Autocfg.csv	*csv	It describes the files names and the definitions of the plugin software to be installed in the robot controller. It may also change the setting of the robot for the plugin. The file name must be "Autocfg.csv".	Chapter 5 Chapter 11
Installation definition file for multi-language	plginfo**.dat (** indicates the language)	*dat	The file may contain the PLGINFO command in the specified language, to be provided for each language. If the file does not exist for the current language, the PLGINFO command in the Autocfg.csv file is used as default. The file name must be "plginfo**.dat" (** indicates the language).	Chapter 5
Detail description file	Arbitrary	*txt *htm	The contents of this text file is displayed as the detail description in the Plugin List screen. Files may be added for additional supported languages.	Chapter 3 Chapter 5
License information file	Arbitrary	*txt *htm	The contents of this text file is displayed as the EULA or readme text in the Plugin Installation screen. Files may be added for additional supported languages.	Chapter 5
Image file 1 for Plugin List screen	Arbitrary	*png *jpg *bmp *gif	The image in this optional file is displayed as part of the detail description in the Plugin List screen. It may typically be the picture of the equipment that the plugin software controls or the company logo of the provider. The use of this file is optional. The picture fits to the display space when the aspect ratio is 5:4 (vertically oblong rectangle). The file size must be 100kB or less so that it can be displayed and installed within a reasonable amount of time.	Chapter 3 Chapter 5
Image file 2 for Plugin List screen	Arbitrary	*png *jpg *bmp *gif	The image in this optional file is displayed as part of the detail description in the Plugin List screen. It may typically be the picture of the equipment that the plugin software controls or the company logo of the provider. The use of this file is optional. The picture fits to the display space when the aspect ratio is 5:4 (vertically oblong rectangle). The file size must be 100kB or less so that it can be displayed and installed within a reasonable amount of time.	Chapter 3 Chapter 5
Command file	Arbitrary	*cf	A command file may be specified to run as part of the plugin installation/uninstallation processing. KCL commands can be specified in the command file to do various operations, such as loading a special program, etc.	Section 5.2
Menu item definition file	plmenu**.xml (** indicates the language)	*xml	Files to define/add an item in the pull down list. The file name must be "plmenu**.xml" (** indicates the language).	Section 7.6
Plugin screen data file	Arbitrary	*stm *htm *js *css	Screen data files for plugin screen, to be provided for each language. The file name is arbitrary but must be the same as the definition in the menu item definition file (*xml). The *js file may be added for checking the setting range, operating the related elements and describing callback function. The *css file may also be added to apply own style when creating plugin screen display, as necessary.	Chapter 6 Chapter 7 Chapter 13

File	Name	Extension	Purpose	Chapter
Image files for plugin screen	Arbitrary	*png *jpg *bmp *gif	Images in this optional file is displayed as part of plugin screen developed by the provider. The use of this file is optional. The file size must be 100kB or less so that it can be displayed and installed within a reasonable amount of time.	Chapter 6 Chapter 7
plugin instruction definition file	Arbitrary	*xml	The files are for being associated the following contents, to be provided for each language; ・ File names of each programs ・ Icon names for display ・ Detailed screen display name ・ Callback function name for detailed screen display Only half-width alphanumeric is applicable for file name to put in. The file name is arbitrary.	Chapter 8
Plugin instruction TP program	“IPL_(Plugin ID)_(Arbitrary Program Name)”.tp	*tp	TP Program that runs when plugin instruction is executed. The program name must be “IPL_(Plugin ID)_”(Arbitrary Program Name)”.tp.	Chapter 5 Chapter 8
KAREL Program	“IPL_(Plugin ID)_(Arbitrary Program Name)”.pc	*pc	KAREL Program that defines KAREL variables. It is also used for plugin instruction that requires complicated control process cannot be implemented by TP programs. The program name must be “IPL_(Plugin ID)_”(Arbitrary Program Name)”.pc.	Chapter 5 Chapter 6 Chapter 8 Chapter 11 Chapter 12
Plugin instruction icon image file	Arbitrary	*png	Icon image file for plugin instruction on iHMI editor. The file name is arbitrary but must be the same as the definition in the plugin instruction definition file (*.xml). The file size must be 100kB or less so that it can be displayed and installed within a reasonable amount of time.	Chapter 8
Plugin instruction detail screen data file	Arbitrary	*stm *htm *js *css	Screen data files for plugin instruction detail screen, to be provided for each language. The file name is arbitrary but must be the same as the definition in the plugin instruction definition file (*.xml). The *js file may be added for checking the setting range, operating the related elements and describing callback function. The *css file may also be added to apply own style when creating plugin screen display, as necessary.	Chapter 8 Chapter 13
Plugin specific alarm definition file	plgalm.dat plgalm**.dat (** indicates the language)	*dat	Files to predefine alarm message specific to the Plugin. The plgalm.dat file defines the English message texts as the default language. To support multiple language, a new file for each language should be created, and included in the IPL file, with the file name appended by the two letters indicating the language. For example, “plgalmkn.dat” for Japanese language.	Section 8.3.3

Note

- It is not allowed to have directory hierarchy in the installation package file (*.IPL file). Please place all component files in the root directory when packaging to an archive file.
- The size of image files in the installation package must be 100kB or less so that it can be displayed and installed within a reasonable amount of time.
- The file name of the installation package file (*.IPL file) must be the same name as the Plugin ID. See section 5.1 “FORMAT OF INSTALLATION DEFINITION FILE” for details of Plugin ID.

The following chapters explain about each file in the installation package.

5 INSTALLATION DEFINITION FILE

The installation definition file (Autocfg.csv) defines the plugin software in the robot controller. The file name must be “Autocfg.csv”. (CSV file: Comma Separated Values)

The same commands that are used in the Robot Setup Assistance Tool (A05B-2600-J737) can be used in this Autocfg.csv file, and therefore can be used to set up the robot configuration, such as defining I/O points, setting the tool frame, etc. Refer to chapter 4 “Robot Setup Assistance Tool” of System Design Tool Operator’s Manual (B-83274EN) for more detail information about the commands available in the Robot Setup Assistance Tool.

There have been a number of new commands added to support the plugin installation/uninstallation purposes, as listed in the table below. The same Autocfg.csv file is used when a plugin is uninstalled.

Table 5 (a) New Commands that are available for plugin Installation

Command	Description
PLGINFO command	Defines the information about the plugin software.
PLGID command	Defines the unique ID and the version number of the plugin software.
OPTION command	Modifies the way how the plugin software is installed.
REQUIRE command	Verifies that the feature required for the plugin to function is loaded in the system.
CHECKVER command	Verifies the robot controller software version required for the plugin to function.
DESCFILE command	Defines the names of the description files and image files to be used for displaying the detail description and/or the license information of the plugin.
RUNCM command	Runs the specified command file.
LOAD command	Loads the specified program (TP or KAREL).
RUNPROG command	Runs the specified program (TP or KAREL).
VAR command	Defines and creates a KAREL variable with optional initial value.
DSET command	Sets a KAREL variable to the specified value
PLGMTOP command	Moves the menu item “PLUGINS” to the top

Note
PLGINFO, PLGID, OPTION, REQUIRE, CHECKVER, DESCFILE and PLGMTOP commands may not be used in the “Robot Setup Assistance Tool” (A05B-2600-J737).

This chapter explains the content of the Autocfg.csv file, how they are used to install/uninstall a plugin software.

5.1 FORMAT OF INSTALLATION DEFINITION FILE

The figure 5.1 (a) shows an example of an Autocfg.csv file, to define a plugin software to be installed/uninstalled. The format of the file is shown here. Note that the terminator (!!!!) at end of each line is required due to the specification of CSV file processing (except for REM remark lines).

```
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
```

Fig 5.1 (a) Format of an Autocfg.csv file

1. The first line of Autocfg.csv must always be the following REM (remark) command.

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

   This line will cause the system to recognize this Autocfg.csv file as the installation/uninstallation definition file for a plugin.

2. PLGINFO command is specified in the second line of Autocfg.csv file, that defines the name of the plugin software, and other basic information of the plugin.

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

Name	Name of the plugin software
Desc	Brief description of the plugin software
Publisher	Name of the provider of the plugin software
Application	Brief description of the purpose of the plugin Software

   Please make each item within 32 characters. Please enclose the item in the double-quotes (“) if a comma (,) is used within the item.

3. The commands, PLGID, OPTION, REQUIRE, and CHECKVER may be specified in arbitrary order on or after the third line of Autocfg.csv file. PLGID command defines the Plugin ID of the plugin software, and it may be specified only once in the Autocfg.csv file. The command syntax is as follows. Please see section 5.1.3 for the OPTION command, section 5.1.4 for the REQUIRE commands, and section 5.1.5 for the CHECKVER commands.

   ```
   PLGID,PluginID,Version,!!!!
   ```

Plugin ID	Identifies the plugin from other plugins. This ID must be unique. Plugin ID must be “(Company Name)_(Plugin Name)” to make it unique. Alphanumeric and underscore “_” are allowed and up to 16 char or less. Ex) `FANUC_SmplGrp` **`1234567890123456`**
Version	Indicates the version MM.RR.SS, where MM: major version, RR: revision, SS: sub-revision respectively.

Note

- Plugin ID must be defined as "(Company Name)_(Plugin Name)", and the file name of TP program and KAREL program must be "IPL_(Plugin ID)_(Arbitrary program name)". For example, if you define the Plugin ID as **"FANUC_SmplGrp"**, the program name must be "**IPL_FANUC_SmplGrp_*****.TP" and "**IPL_FANUC_SmplGrp_*****.PC". Here "***" indicates arbitrary program name.
- The parameter instruction in TP program allow the number of chars up to 30 or less for parameter name. In case to specify KAREL variable, KAREL program name and KAREL variable name must be specified in this instruction as below. Plugin ID, KAREL program name, and KAREL variable name must be defined in accord with this limitation.

```
Ex) R[10001]=$[IPL_FANUC_SmplGrp_Cfg]OpenDo
              123456789012345678901234567890
```

4. DESCFILE command is specified after PLGID/OPTION/REQUIRE/CHECKVER commands in the Autocfg.csv file, to specify the file names to be used to describe the plugin. Please note the commands, PLGID, OPTION, REQUIRE or CHECKVER, may not be specified after the DESCFILE command.

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

desc_file	A text file that includes the detail description of the plugin software (*.txt or *.htm) should be written in English as the default language. You may add a file for each supported language, with the file name appended by the two letters indicating the country.
pic_file	Image file #1 (it typically shows the picture of the equipment that the plugin controls).
logo_file	Image file #2 (it typically shows the company logo)
eula_file (optional)	This indicates the name of a text file that includes the readme or the license information of the plugin software (*.txt or *.htm), that is displayed in the "Plugin Install" screen. It should be written in English as the default language. You may add a file for each supported language, with the file name appended by the two letters indicating the country. It is an optional field. The text box will not appear in the Install screen if this field is not specified.
NOCHKBOX (optional)	If the "eula_file" is specified, the operator must check the checkbox to agree with the license information before he/she can press the "Install" button. If NOCHKBOX is specified, however, the checkbox will not appear, and the installation will commence without checking the checkbox.

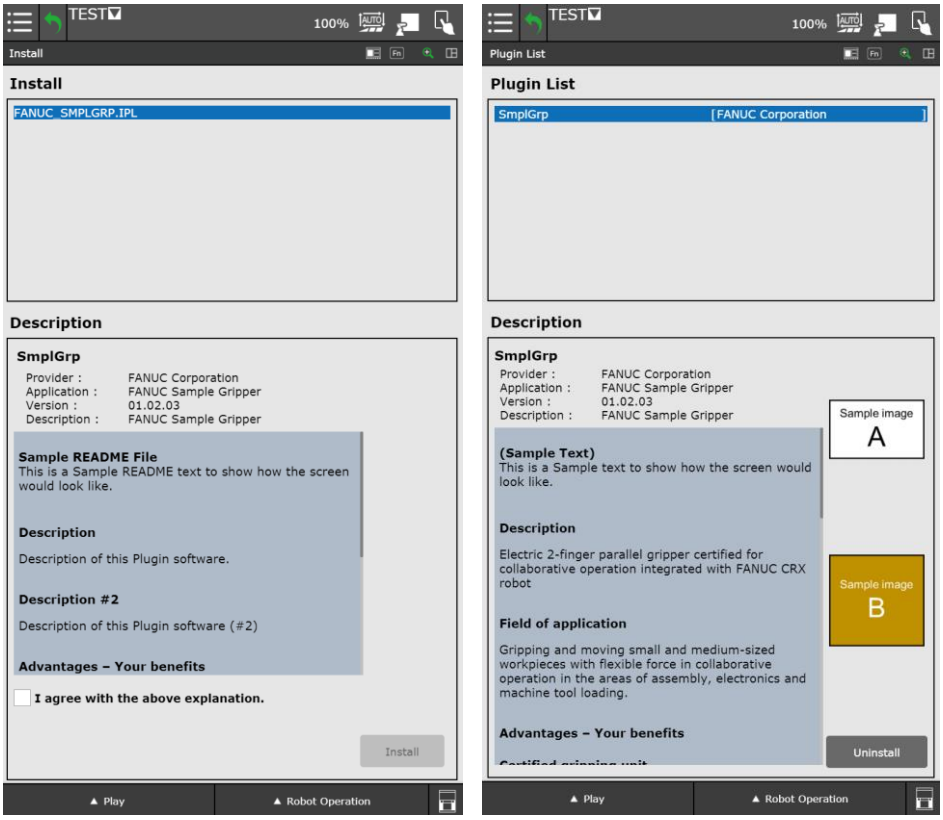


Fig 5.1 (b) Plugin Installation and List screen

Note
The commands, PLGID, OPTION, REQUIRE or CHECKVER, may not be specified after the DESCFILE command.

5. On the lines after the DESCFILE command, you may specify the commands for the Robot Setup Assistance Tool and the additional commands indicated in the Table 5 (a) to set up the robot for the plugin, and to load the components of the plugin software.

⚠ Caution
The license information, that is displayed in the Plugin Install screen, is available on the robot controller software V9.40P/05 or after. If a plugin uses this feature, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/04 or older.

Note
The linefeed code in the Autocfg.csv must be CR+LF.

5.1.1 Size of Image Files

The two image files specified in the DESCFILE command are displayed in the “Plugin List” screen. Regardless of the original size, the size of the picture is automatically adjusted to the size of the display area on the screen. The picture fits to the display space when the aspect ratio is 5:4 (vertically oblong rectangle). See the figure below.

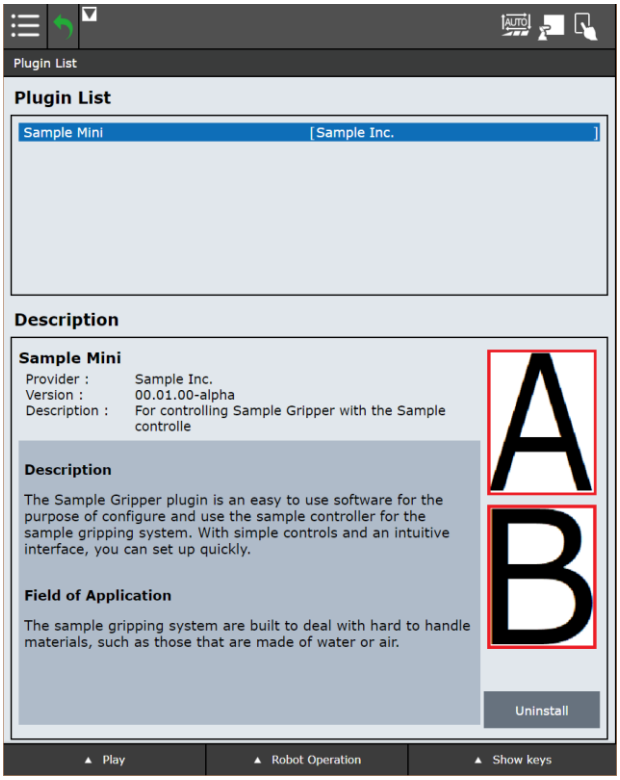


Fig 5.1.1 (a) Pictures displayed in the Plugin List screen

Note
The size of image files in the installation package must be 100kB or less so that it can be displayed and installed within a reasonable amount of time.

5.1.2 Multi-Language Support for PLGINFO/PLGSUB Command

The information in the PLGINFO command and PLGSUB command (see section 12.5.3) within Autocfg.csv should be described in English as the default language. As required, a plginfo**.dat file may be added to specify PLGINFO/PLGSUB command in a different language.

File name: plginfo**.dat (where "**" represents the two letters to indicate the country)

In order to switch languages according to the language setting in the robot controller, it is necessary to use the correct file suffix shown in the table below for the file name.

Table 5.1.1 (a) Language and File Name Suffix

Language	File Name Suffix	File Name	Character code
English	eg	(plginfo**eg**.dat is not required as the info in Autocfg.csv is used for English)	iso-8859-1
Japanese	kn	plginfo**kn**.dat	Shift_JIS
Chinese	ch	plginfo**ch**.dat	GB2312
Taiwanese	tw	plginfo**tw**.dat	chinese-big5
German	gr	plginfo**gr**.dat	iso-8859-1
French	fr	plginfo**fr**.dat	iso-8859-1
Italian	it	plginfo**it**.dat	iso-8859-1
Spanish	sp	plginfo**sp**.dat	iso-8859-1

The plginfo**.dat file may contain one line of PLGINFO command and required number of PLGSUB commands only. If the corresponding plginfo**.dat file exists for the currently selected language according

to the table 5.1.1(a), the information in plginfo**.dat file is used; otherwise the information in Autocfg.csv file is used.

For example, if files, plginfo**eg**.dat, plginfo**kn**.dat, exist in the system, and the current selected language in the robot controller is Japanese, the PLGINFO/PLGSUB command in plginfo**kn**.dat file is used. If plginfo**kn**.dat file does not exist, the PLGINFO/PLGSUB command in Autocfg.csv is used.

You may only specify one line of PLGINFO command and required number of PLGSUB commands in plginfo**.dat file.

Note
- You may specify PLGSUB command only if the number of tools using Tool I/F associated with the plugin software is two or more.
- The first line of plginfo**.dat file must be always PLGINFO command. When specifying PLGSUB command in plginfo**.dat file, PLGSUB command may be specified after the line of PLGINFO command.
- Blank line may not be inserted between the lines of PLGINFO/PLGSUB command.
- For further information about PLGSUB command, refer to section 12.5.3 “Setting Sub ID for the secondary and subsequent tools using Tool I/F” in this document.

5.1.3 OPTION Command

The OPTION command specifies the way how the installation of the plugin software is performed.

OPTION, ANYROBOT command:

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

Without specifying the OPTION,ANYROBOT command, the plugin may only be installed on the CRX robot. On the other hand, the plugin may be installed onto any robot as long as it is equipped with the Tablet TP if the OPTION,ANYROBOT command is specified,

The alarm SDTL-105 (Unsupported Robot type) will be posted if an attempt is made to install a plugin to a non-CRX robot without OPTION,ANYROBOT command. Please refer to section 9.1 for more information about the error.

Note
The commands, PLGID, OPTION, REQUIRE or CHECKVER, may not be specified after the DESCFILE command.

5.1.4 REQUIRE Command

The REQUIRE command indicates that a feature (option) is required for the plugin to function.

For example, the following line indicates that a plugin has to have the R648 “User Socket Messaging” feature in the system in order for the plugin software to function.

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

The plugin will not be installed if the required option is not loaded in the system, and SDTL-106 (Option is required) alarm will be posted. Please see section 9.1 for more information about the error.

Note
The commands, PLGID, OPTION, REQUIRE or CHECKVER, may not be specified after the DESCFILE command.

5.1.5 CHECKVER Command

The CHECKVER command specifies the robot controller software version required for the plugin to function.

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

series	Specifies the robot controller software series. Example) V9.40
version	Specifies the robot controller software version. Example) 68

Specify as follows to allow the plugin software to be installed to the robot controller that has system software version, for example, V9.40P/68 or after installed.

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

In the above case, if the robot controller software version is V9.40P/67 or older, you will get "PLUG-031 This plugin requires V9.40P/68 or later" alarm which refuses to install the plugin software when attempting to install it.

If you want to prevent the plugin software from being installed in all versions of the specified software series, specify NA for the version as follows.

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

In the above case, if the robot controller has V9.40 series software installed, you will get "PLUG-033 This plugin is not available on V9.40" alarm which refuses to install the plugin software when attempting to install it.

You can also specify multiple CHECKVER commands in one Autocfg.csv. In this case, if there is at least one line does not satisfy the specified software version, you will get an alarm as described above and the installation will be cancelled.

⚠ Caution
The CHECKVER command is available on the robot controller software V9.40P/58 and after. If a plugin uses this feature, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/57 or older.

Note
- The commands, PLGID, OPTION, REQUIRE or CHECKVER, may not be specified after the DESCFILE command.
- If the software series specified with CHECKVER command does not match the software series installed in the robot controller, the version check is skipped and no alarm occurs.

5.2 COMMANDS FOR SETTING UP SYSTEM

This chapter explains the new commands that have been added for Plugin installation.

5.2.1 RUNCM Command

This command runs the specified KCL command file (extention: *.cf)

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

cmd_file1	Runs during the installation processing.
cmd_file2	Runs during the uninstallation processing.

It assumes the same directory as the CSV file location.
The CD command may be used to change the default directory.

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

path_name	Specifies path name of the new default directory.

5.2.2 LOAD Command

This command loads the specified TP (*.TP) or KAREL program (*.PC).

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

prog_name.tp	Specifies the name of the TP program to be loaded.
prog_name.pc	Specifies the name of the KAREL program to be loaded.

Example)
```
LOAD,IPL_FANUC_SmplGrp_OPEN.TP,!!!!
```

The above example will load the TP program “IPL_FANUC_SmplGrp_OPEN”.

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

If “COLD” is specified, the loaded program will run at every cold start.
The programs loaded by this command will be deleted when the plugin is uninstalled.

⚠ Caution

- Be sure not to implement repetitive processing could results an infinite loop using WHILE statement or REPEAT statement in the KAREL or TP program to be loaded by a plugin software. Be sure to make a plugin to call a KAREL or TP program only when necessary such as when a specific operation is done in a plugin screen or when a plugin instruction is executed in a user TP program.
- If the COLD option is specified after the program name in the LOAD command, the loaded program will run at every cold start. A program loaded with the COLD option only assumes processing that ends immediately such as some KAREL variables settings.

5.2.3 RUNPROG/RUNPRGWT Command

This command runs the specified KAREL program during installation / uninstallation.

The following line runs prog_name1 (the KAREL program name) during installation, and runs prog_name2 (the KAREL program name) during uninstallation.

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

Here the prog_name1 is mandatory but the prog_name2 is optional. If you don't need any program to run during uninstallation, you can skip as follows.

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

prog_name1	Specify the KAREL program name which runs during install processing. The prog_name1 is mandatory.
prog_name2	Specify the KAREL program name which runs during uninstall processing. The prog_name2 is optional.

RUNPROG command will cause the system to continue processing without waiting for the program to finish its execution. If the processing should continue after waiting for its completion, please use RUNPRGWT command. For the RUNPROG/RUNPRGWT command, the prog_name1 is mandatory and the prog_name2 is optional.

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

⚠ Caution

A program specified with the RUNPROG/RUNPRGWT command only assumes processing that ends immediately such as some KAREL variables settings. Be sure not to implement processing could results an infinite loop using WHILE statement or REPEAT statement in the program.

Note

- The program specified with the RUNPROG/RUNPRGWT command must be a KAREL program. This command cannot be used for running a TP program.
- %NOLOCKGROUP attribute must be set in a KAREL program that is specified with the RUNPROG/RUNPRGWT command.
- The RUNPRGWT command is similar to the RUNPROG command except that this command will cause the CSV command processing to wait until the program finishes its execution. Please note ABORT statement may not be used in the program.
- The RUNPROG/RUNPRGWT command requires the KAREL program name which runs during installation to be specified, but the KAREL program name which runs during uninstallation is optional.

5.2.4 VAR/DSET Command

VAR command will define/create a new KAREL variable.

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

prog_name	Program name the new KAREL variable belongs to

var_name	Variable name to be created. It must be a valid variable name for a KAREL program.
type_name	Type of the variable. Available types are INTEGER, REAL, BOOLEAN, or STRING[n]. (n: string length). The table 5.2.4 (a) shows the available data types for this command.
def_value	Optional field to specify the initial value. It must be a valid value for the variable type.

VAR command will create the specified variable under the specified program name with the specified variable type within the CMOS memory. If the specified variable is uninitialized, then it is initialized to the specified default value. There is no need to load the KAREL program with the same name.

Example :
```
VAR,IPL_FANUC_SmplGrp_Cfg,OpenDo,INTEGER,12345,!!!!
```

With the above example, a new variable [IPL_FANUC_SmplGrp_Cfg] OpenDo is created as an integer in the CMOS memory. It is also set to its initial value 12345. If there is the same variable already existing in the system, it will just use the existing variable. If the variable has some valid value, the variable will not be set to the default value.

Note
System variable must not be specified with VAR command.

DSET command sets the KAREL variable or system variable to the specified value.

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

prog_name	Specifies the program name that the new KAREL variable belongs to. For system variable, specifies *system*.
var_name	Specifies the variable name to be set. It must be a valid variable name for a KAREL program or system variable.
type_name	Specifies the type of the variable. You may specify INTEGER, REAL, BOOLEAN, or STRING[n]. (where n indicates the string length). The table 5.2.4 (a) shows the available data types available for this command.
value	Specifies the value of the KAREL variable or system variable to be set. It must be a valid value for the variable type.

This command will set the KAREL variable to the specified value. Unlike the VAR command, it will not create the specified variable if it does not exist.

Example for the case to set KAREL variable (INTEGER) :
```
DSET,IPL_FANUC_SmplGrp_Cfg,OpenDo,INTEGER,0,!!!!
```

Example for the case to set KAREL variable (STRING) :
```
DSET,IPL_FANUC_SmplGrp_Cfg,Command,STRING[10],ABCDEF,!!!!
```

Example for the case to set system variable :
```
DSET,*system*,$MNUTOOLNUM[1],INTEGER,1,!!!!
```

The Table 5.2.4 (a) shows the variable types that can be used for the VAR/DSET commands.

Table 5.2.4 (a) Data type available for VAR/DSET commands

Data Type	Description
BOOLEAN	Boolean type with TRUE or FALSE values. 4 byte long

Data Type	Description
INTEGER	Integer type. 4 byte long
REAL	32 bit Floating point data type. 4 byte long
STRING[n]	String type data. The length of the variable is specified by the number “n”. Maximum 254 characters. Ex) tmp_str : STRING[10]

5.2.5 SYSC Command

SYSC command sets the setup items of the system configuration menu. It is not necessary to use SYSC command if a plugin software does not require to set the items described in this section.

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

item_number	Specifies the item number to set. Refer to section 4.8.17 “System Config” in System Design Tool Operator’s Manual (B-83274EN) for available items.
item_name	You can specify the item name just as a comment, but it is ignored.
value1	Specifies the value to set. In case of character string, specify in uppercase.To set enable/disable, specify “ENABLE” or “DISABLE”.
value2	Specifies the second value to set (when applicable).

5.2.6 SSET Command

SSET command sets the system configuration items. It is not necessary to use SSET command if a plugin software does not require to set the items described in this section.

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

item_name	Specifies the item to set. Use the predefined item name. Refer to the next sections for the available items.
Index	Specifies the item number to set.
value1	Specifies the value to set.
value2	Specifies the second value to set (when applicable).

This command set the specified value to the specified item.

Example: The following line will set the address of the Ethernet port #1 to "192.168.1.2".

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

The following sections explains the available system items to be used for SSET command.

> **⚠ Caution**
> SSET command is available on the robot controller software V9.40P/18 or after. If a plugin uses this command, make it clear in the manual that is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/17 or older. If you attempt to install the plugin software on the robot controller software V9.40P/17 or older, you’ll get SDTL-084 (Wrong data type) alarm.

5.2.6.1 Setting host communication (Ethernet) related items

The following Host Communication (Ethernet) related items may be set using the SSET command.

1. PORT_ADDR: IP address of the Ethernet port on the robot controller. Please enclose the IP address with the double-quotes.
```
Syntax:
   SSET,PORT_ADDR,port_num,IP_address,!!!!
Example:
   SSET,PORT_ADDR,1,”192.168.1.2”,!!!!
```

2. PORT_SNMASK: Subnet mask of the Ethernet port on the robot controller. Please enclose the subnet mask value with the double-quotes.
```
Syntax:
   SSET,PORT_SNMASK,port_num,mask,!!!!
Example:
   SSET,PORT_SNMASK,1,”255.255.0.0”,!!!!
```

3. PORT_RTADDR: router IP address of the Ethernet port on the robot controller. Specify “1” for the port_num (index). Please also enclose the IP address value with the double-quotes.
```
Syntax:
   SSET,PORT_RTADDR,(port_num),IP_address,!!!!
Example:
   SSET,PORT_RTADDR,1,"192.168.1.1",!!!!
```

4. PORT_HOSTNAME: Host name of the Ethernet port of the robot controller. The port_num (index) is ignored. Please specify “1” for it.
```
Syntax:
   SSET,PORT_HOSTNAME,(port_num),name,!!!!
Example:
   SSET,PORT_HOSTNAME,1,ROBOT,!!!!
```

5. PORT_DHCPENB: Enable/disable DHCP of the Ethernet port on the robot controller.
```
Syntax:
   SSET,PORT_DHCPENB,port_num,value,!!!!
Example:
   SSET,PORT_DHCPENB,1,TRUE,!!!!
```

5.2.6.2 Ethernet IP related setup items

The following Ethernet IP related items may be set using the SSET command.

1. EIP_ADAPTER: sets the specified slot to be used as ADAPTER. Use EIP_ADAPTER before setting the items as it will reset all items to their default values.
```
Syntax:
   SSET,EIP_ADAPTER,slot_num,!!!!
Example:
   SSET,EIP_ADAPTER,3,!!!!
```

2. EIP_SCANNER: sets the specified slot to be used as SCANNER. Use EIP_SCANNER before setting the items as it will reset all items to their default values.
```
Syntax:
   SSET,EIP_SCANNER,slot_num,!!!!
Example:
   SSET,EIP_SCANNER,3,!!!!
```

3. EIP_DESC: Set the description string of the specified slot #
```
Syntax:
   SSET,EIP_DESC,slot_num,description_string,!!!!
```

```
Example:
   SSET,EIP_DESC,2,Connection2,!!!!
```

4. EIP_ENABLE: Enable/Disable the specified slot #

```
Syntax:
   SSET,EIP_ENABLE,slot_num,boolean_data,!!!!
Example:
   SSET,EIP_ENABLE,2,TRUE,!!!!
```

5. EIP_HOST: Set the host name/IP address of the host of the specified slot # (for Scanner only)

```
Syntax:
   SSET,EIP_HOST,slot_num,ip_address,!!!!
Example:
   SSET,EIP_HOST,2,192.168.1.1,!!!!
```

6. EIP_VENDOR: Set the Vendor ID string of the specified slot # (for Scanner only)

```
Syntax:
   SSET,EIP_VENDOR,slot_num,integer_value,!!!!
Example:
   SSET,EIP_VENDOR,2,0,!!!!
```

7. EIP_DEVTYPE: Set the device type of the specified slot # (for Scanner only)

```
Syntax:
   SSET,EIP_DEVTYPE,slot_num,integer_value,!!!!
Example:
   SSET,EIP_DEVTYPE,2,0,!!!!
```

8. EIP_PRDCOD: Set the product code of the specified slot # (for Scanner only)

```
Syntax:
   SSET,EIP_PRDCOD,slot_num,integer_value,!!!!
Example:
   SSET,EIP_PRDCOD,2,0,!!!!
```

9. EIP_INSIZE: Set the input size of the specified slot #

```
Syntax:
   SSET,EIP_INSIZE,slot_num,integer_value,!!!!
Example:
   SSET,EIP_INSIZE,1,32,!!!!
```

10. EIP_OUTSIZE: Set the output size of the specified slot #

```
Syntax:
   SSET,EIP_OUTSIZE,slot_num,integer_value,!!!!
Example:
   SSET,EIP_OUTSIZE,1,32,!!!!
```

11. EIP_ALMSEV: Set the alarm severity of the specified slot #. Valid values are 0: STOP, 1: WARN, 2: PAUSE.

```
Syntax:
   SSET,EIP_ALMSEV,slot_num,integer_value,!!!!
Example:
   SSET,EIP_ALMSEV,1,0,!!!!
```

12. EIP_TORPI: Set the Target → Origin RPI of the specified slot # (for Scanner only)

```
Syntax:
   SSET,EIP_TORPI,slot_num,integer_value,!!!!
Example:
```

```
SSET,EIP_TORPI,2,32,!!!!
```

13. EIP_OTRPI: Set the Origin → Target RPI of the specified slot # (for Scanner only)

```
Syntax:
   SSET,EIP_OTRPI,slot_num,integer_value,!!!!
Example:
   SSET,EIP_OTRPI,2,32,!!!!
```

14. EIP_INASSY: Set the Assembly instance (input) of the specified slot # (for Scanner only)

```
Syntax:
   SSET,EIP_INASSY,slot_num,integer_value,!!!!
Example:
   SSET,EIP_INASSY,2,200,!!!!
```

15. EIP_OUTASSY: Set the Assembly instance (output) of the specified slot # (for Scanner only)

```
Syntax:
   SSET,EIP_OUTASSY,slot_num,integer_value,!!!!
Example:
   SSET,EIP_OUTASSY,2,201,!!!!
```

16. EIP_CONF: Set the Configuration instance of the specified slot # (for Scanner only)

```
Syntax:
   SSET,EIP_CONF,slot_num,integer_value,!!!!
Example:
   SSET,EIP_CONF,2,0,!!!!
```

17. EIP_DATATYP: Set the I/O data type of the specified slot # (for Scanner only). Valid values are, 8 (TRUE) or 16 (FALSE), meaning 8-bit or 16-bit.

```
Syntax:
   SSET,EIP_DATATYP,slot_num,(TRUE or FALSE),!!!!
   Or
   SSET,EIP_DATATYP,slot_num,(8 or 16),!!!!
Example:
   SSET,EIP_DATATYP,2,TRUE,!!!!
   True: 8-bit
   False: 16-bit
   Or
   SSET,EIP_DATATYP,2,8,!!!!
   8: 8-bit
   16: 16-bit
```

18. EIP_TMOMUL: Set the Timeout multiplier of the specified slot # (for Scanner only)

```
Syntax:
   SSET,EIP_TMOMUL,slot_num,integer_value,!!!!
   Or
   SSET,EIP_TMOMUL,slot_num,DEFAULT,!!!!
Example:
   SSET,EIP_TMOMUL,2,3,!!!!
   Or
   SSET,EIP_TMOMUL,2,DEFAULT,!!!!
```

Valid values are as follows

Timeout Multiplier
4
8

16
32
64
128
256
512
DEFAULT

19. EIP_RECONN: Set the Reconnect value of the specified slot # (for Scanner only)
```
Syntax:
    SSET,EIP_RECONN,slot_num,(TRUE or FALSE),!!!!
Example:
    SSET,EIP_RECONN,1,FALSE,!!!!
```

20. EIP_REVISION: Set the major/minor revision of the specified slot # (for Scanner only)
```
Syntax:
    SSET,EIP_REVISION,slot_num,major,minor,!!!!
Example:
    SSET,EIP_REVISION,1,1,0,!!!!
```
This example will set the major revision number as 1, and the minor revision number as 0.

21. EIP_QCONN: Set the quick connect of the specified slot # (for Scanner only)
```
Syntax:
    SSET,EIP_QCONN,slot_num,boolean_value,!!!!
Example:
    SSET,EIP_QCONN,2,FALSE,!!!!
```

22. EIP_TRPORT: Set the Transport type of the specified slot # (for Scanner only). Valid values are, FALSE: unicast, TRUE: Multicast. You may also specify unicast, or multicast.
```
Syntax:
    SSET,EIP_TRPORT,slot_num,boolean_value,!!!!
Example:
    SSET,EIP_TRPORT,2,FALSE,!!!!
```

23. EIP_FORMAT: Set the OT and TO format of the specified slot # (for Scanner only)
```
Syntax:
    SSET,EIP_FORMAT,slot_num,ot_form,to_form,!!!!
Example:
    SSET,EIP_FORMAT,1,3,0,!!!!
```

"O->T format" is set by ot_form parameter using the following values.
- 0: Run/Idle Header
- 1: Zero-Length
- 2: Modeless
- 3: Heartbeat

"T->O format" is set by to_form parameter using the following values.
- 0: Run/Idle Header
- 1: Zero-Length
- 2: Modeless
- 3: Heartbeat

5.2.6.3 Payload change distance, external force limit, safe I/O to disable each force limit (for CR and CRX series only)

The following setup items for Payload change distance, External force limit, Safe I/O to disable each force limit may be set using the SSET command.

⚠ Caution
Payload change distance, External force limit, and Safe I/O to disable each force limit using the SSET command is available on the robot controller software V9.40P/37 or after. If a plugin sets these items in Autocfg.csv, make it clear in the manual that is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/36 or older.

1. CR_PLD_CHNG_X: Sets the limit of Payload change distance for direction X. The valid range for both +/- Limit is 0.0 to 50.0.
```
Syntax:
   SSET,CR_PLD_CHNG_X,1,+Limit,-Limit,!!!!
Example:
   SSET,CR_PLD_CHNG_X,1,1.0,2.0,!!!!
```

2. CR_PLD_CHNG_Y: Sets the limit of Payload change distance for direction Y. The valid range for both +/- Limit is 0.0 to 50.0.
```
Syntax:
   SSET,CR_PLD_CHNG_Y,1,+Limit,-Limit,!!!!
Example:
   SSET,CR_PLD_CHNG_Y,1,1.0,2.0,!!!!
```

3. CR_PLD_CHNG_Z: Sets the limit of Payload change distance for direction Z. The valid range for both +/- Limit is 0.0 to 50.0.
```
Syntax:
   SSET,CR_PLD_CHNG_Z,1,+Limit,-Limit,!!!!
Example:
  SSET,CR_PLD_CHNG_Z,1,1.0,2.0,!!!!
```

4. CR_PLD_CHNG_R: Sets the limit of Payload change distance for rotation. The valid range for Limit is 0.0 to 50.0.
```
Syntax:
   SSET,CR_PLD_CHNG_R,1,Limit,!!!!
Example:
   SSET,CR_PLD_CHNG_R,1,1.0,!!!!
```

5. CR_FORCE_LIM1: Sets the limit of External force 1 for Collaborative robot.
```
Syntax:
   SSET,CR_FORCE_LIM1,1,Limit,!!!!
Example:
   SSET,CR_FORCE_LIM1,1,100.0,!!!!
```

Note
- External force limit 2, 3, 4 can be set by the command CR_FORCE_LIM2, 3, 4.
- The valid range for External force limit is 0.0 to 150.0, except for limit 1, for which 0.0 cannot be set.

6. CR_DISABLE_IO1: Sets Safe I/O signal 1 to disable each force limit for Collaborative robot.

```
Syntax:
   SSET,CR_DISABLE_IO1,1,Disabling input signal type,Disabling
  input signal number,!!!!
Example:
   SSET,CR_DISABLE_IO1,1,5,1,!!!!
```

Note

- Safe I/O signal 2, 3, 4 to disable each force limit can be set by the command CR_DISABLE_IO2, 3, and 4.
- The valid range for both disabling input signal type and disabling input signal number is 0 to 999. If invalid safe I/O signal is set, the DCS parameters cannot be applied.
- For disabling input signal type, please refer to section 9.1 “Safe I/O” in Dual Check Safety Function OPERATOR’S MANUAL (B-83184EN) to set the slot number corresponding to the safe I/O for the disabling input signal type.

5.2.6.4 Signal for manual operation in AUTO (for CRX series only)

The following setup items for Signal for Manual Operation in AUTO may be set using the SSET command.

⚠ Caution

Signal for Manual Operation in AUTO using the SSET command is available on the robot controller software V9.40P/53 or after. If a plugin sets these items in Autocfg.csv, make it clear in the manual that is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/52 or older.

1. CR_MNL_AUTO_IO_IDX: Sets Signal for Manual Operation in AUTO. The valid range for input index is 0 to 999

```
Syntax:
   SSET,CR_MNL_AUTO_IO_IDX,1,input index,!!!!
Example:
   SSET,CR_MNL_AUTO_IO_IDX,1,10,!!!!
```

5.2.6.5 Manual guided teaching related setup items (for CRX series only)

The following setup items for the manual guided teaching may be set using the SSET command.

⚠ Caution

Manual guided teaching configurations using the SSET command is available on the robot controller software V9.40P/51 or after. If a plugin sets these items in Autocfg.csv, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/50 or older.

Free Hand Teaching

1. MGT_FREE_HAND_IO: Sets the activation input for free hand teaching. (Input type : DI, RI, or F)

```
Syntax:
```

```
    SSET,MGT_FREE_HAND_IO,1,input type,input index,!!!!
Example:
    SSET,MGT_FREE_HAND_IO,1,RI,2,!!!!
```

Configuration Input Signals

1. MGT_IN_ACTIVE_TYPE: Sets the activation type. Valid types are DISABLE, ALWAYS and IO.

```
Syntax:
    SSET,MGT_IN_ACTIVE_TYPE,1,type,!!!!
Example:
    SSET,MGT_IN_ACTIVE_TYPE,1,IO,!!!!
```

2. MGT_IN_ACTIVE_IO: Sets the input signal for IO activation type. (Input type : DI, RI, or F)

```
Syntax:
    SSET,MGT_IN_ACTIVE_IO,1,input type,input index,!!!!
Example:
    SSET,MGT_IN_ACTIVE_IO,1,DI,4,!!!!
```

3. MGT_IN_MOTION_TYPE: Sets the input signal for motion type. (Input type : DI, RI, F, REG, or GI)

```
Syntax:
    SSET,MGT_IN_MOTION_TYPE,1,input type,input index,!!!!
Example:
    SSET,MGT_IN_MOTION_TYPE,1,REG,5,!!!!
```

4. MGT_IN_WEIGHT_FREE: Sets the input signal for operation weight for Free (Input type : REG, or GI)

```
Syntax:
    SSET,MGT_IN_WEIGHT_FREE,1,input type,input index,!!!!
Example:
    SSET,MGT_IN_WEIGHT_FREE,1,REG,5,!!!!
```

Note
You can set the input signal for each motion type by changing the word from FREE to TRANS, ROT or CSTM.

5. MGT_IN_CSTM_FRM: Sets the input signal for the frame of CUSTOM motion type (Input type : DI, RI, F, REG, or GI)

```
Syntax:
    SSET,MGT_IN_CSTM_FRM,1,input type,input index,!!!!
Example:
    SSET,MGT_IN_CSTM_FRM,1,REG,5,!!!!
```

6. MGT_IN_ CSTM_X: Sets the input signal for the movable direction of CUSTOM motion type. (Input type : DI, RI, or F)

```
Syntax:
    SSET,MGT_IN_CSTM_X,1,input type,input index,!!!!
Example:
    SSET,MGT_IN_CSTM_X,1,DI,4,!!!!
```

Note
You can set the input signal for each direction by changing the note from X to Y, Z, RX, RY, or RZ.

Status Output Signals

1. MGT_OUT_ACTIVE: Sets the output signal to indicate the manual guided teaching working. (Output type : DO, RO, or F)

```
Syntax:
   SSET,MGT_OUT_ACTIVE,1,type,!!!!
Example:
   SSET,MGT_OUT_ACTIVE,1,DO,3,!!!!
```

2. MGT_OUT_MOTION_TYPE: Sets the output signal for the motion type (Output type : GO, REG)

```
Syntax:
   SSET,MGT_OUT_MOTION_TYPE,1,Output type,output index,!!!!
Example:
   SSET,MGT_OUT_MOTION_TYPE,1,REG,3,!!!!
```

3. MGT_OUT_FREE: Sets the output signal to indicate the motion type is FREE (Output type : DO, RO, F)

```
Syntax:
   SSET,MGT_OUT_FREE,1,Output type,output index,!!!!
Example:
   SSET,MGT_OUT_FREE,1,DO,3,!!!!
```

Note
You can set the output signal for each motion type by changing the word from FREE to TRANS, ROT or CSTM.

4. MGT_OUT_WEIGHT: Sets the output signal for the current operation weight (Output type : GO, REG)

```
Syntax:
   SSET,MGT_OUT_WEIGHT,1,Output type,output index,!!!!
Example:
   SSET,MGT_OUT_WEIGHT,1,REG,3,!!!!
```

5. MGT_OUT_WEIGHT_FREE: Sets the output signal for FREE's operation weight (Output type : GO, REG)

```
Syntax:
   SSET,MGT_OUT_WEIGHT_FREE,1,Output type,output index,!!!!
Example:
   SSET,MGT_OUT_WEIGHT_FREE,1,REG,3,!!!!
```

Note
You can set the output signal for each motion type's operation weight by changing the word from FREE to TRANS, ROT or CSTM.

6. MGT_OUT_CSTM_FRM: Sets the output signal for the frame of CUSTOM motion type (Output type : GO, REG)

```
Syntax:
   SSET,MGT_OUT_CSTM_FRM,1,Output type,output index,!!!!
Example:
   SSET,MGT_OUT_CSTM_FRM,1,REG,3,!!!!
```

7. MGT_OUT_CSTM_X: Sets the output signal to indicate X direction is movable on CUSTOM motion type (Output type : DO, RO, F)

```
Syntax:
```

```
    SSET,MGT_OUT_CSTM_X,1,Output type,output index,!!!!
Example:
    SSET,MGT_OUT_CSTM_X,1,DO,3,!!!!
```

Note
You can set the output signal for each direction by changing the note from X to Y, Z, RX, RY, or RZ.

Disable Activation with TP

1. MGT_DSBL_TP_IO: Sets the input signal for Disable activation with TP. (Input type : DI, RI, or F)

```
Syntax:
    SSET,MGT_DSBL_TP_IO,1,input type,input index,!!!!
Example:
    SSET,MGT_DSBL_TP_IO,1,RI,2,!!!!
```

5.2.6.6 Tool I/F communication related setup items

For settings of Tool I/F communication in Autocfg.csv, please refer to chapter 11 “Tool I/F communication”.
For settings of Tool I/F communication for the secondary and subsequent devices in case the plugin software has settings of multiple devices that use Tool I/F, please refer to section 12.5 “Information for Plugin Software Developers”.

5.2.7 PLGMTOP Command

PLGMTOP command moves menu item “PLUGINS” to the top. Uninstalling the plugin software includes the PLGMTOP command in the Autocfg.csv moves menu item “PLUGINS” to the default position.

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

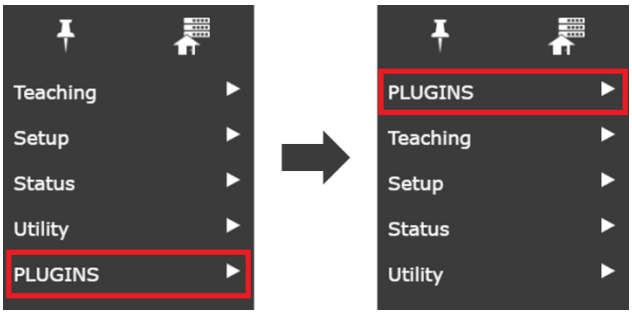


Fig. 5.2.7 (a) Move “PLUGINS” menu

⚠ Caution
PLGMTOP command is available on the robot controller software V9.40P/60 or after. If a plugin uses this command, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/59 or older. If you attempt to install the plugin software on the robot controller software V9.40P/59 or older, you’ll get SDTL-084 (Wrong data type) alarm.

5.2.8 Tool Coordinate System Setting

A tool coordinate system is a Cartesian coordinate system that defines the position of the tool center point (TCP) and the attitude of the tool. On the tool coordinate system, the zero point usually represents the TCP and the Z-axis usually represents the tool axis. When the tool coordinate system is not defined, the mechanical interface coordinate system substitutes for it. Tool coordinates include (x, y, z) indicating the position of the tool center point (TCP), and (w, p, r) indicating the attitude of the tool. Coordinates (x, y, z) [mm] indicate the position of TCP on the mechanical interface coordinate system. Coordinates (w, p, r) [deg] indicate the attitude of the tool and the angular displacement around the X-, Y-, and Z-axes of the mechanical interface coordinate system. The tool center point is used to specify the position data. The attitude of the tool is required to perform tool attitude control. $MNUTOOLNUM[group] is set to specify the tool frame number.

In case of a tool, such as gripper, please define the tool coordinate system in Autocfg.csv. It is recommended to set the tool frame number #1, although the tool frame is available up to #10. Please refer the following example.

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

GPn	Specify the group number. The first 2 characters must be “GP”. The last character specifies the group number. Please specify “GP1” (group 1).
Tool frame number	Specify the tool frame number. It is recommended to set the tool frame number “1”.
Comment	Specify the comment for the tool frame up to 20 characters.
x,y,z,w,p,r	Specify (x,y,z,w,p,r) value of the tool frame. The number of decimal places is 3.

Example)

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

5.3 OVERVIEW OF UNINSTALL OPERATION

This section goes over the uninstall operation how the commands in the Autocfg.csv file function.

For the following commands, the uninstall operation will delete the loaded plugins/programs.

- PLGINFO
- PLGID
- DESCFILE (will delete the specified files)
- LOAD (will delete the specified programs)

For the following commands, the uninstall operation will execute the program specified as the second argument.

- RUNCM
- RUNPROG/RUNPRGWT

For the following commands, the uninstall operation will delete the specified comments only, but the actual setting and entries will remain in the system.

- Register related commands REG/PREG/SREG

- I/O related commands DI/DO/AI/AO/GI/GO, etc.
- Frame related commands UFRM/JFRM/TFRM
- Reference point REF
- Payload setting PLD
- Space Check RSP
- Program Timer PTIM

For the following commands, the uninstall operation will delete the actual setting and entries as well as the comments.

- User alarm UALM
- Macro MCR

For the following commands, the uninstall operation will keep the setting by these commands without modification.

- DI/DO definition (DIA, DOA, GIA, GOA, AIA, AOA)
- Program comments (PRG)
- Position comments (POS)
- Arm load (ALD)
- Auto backup (ATBK)
- System setting (SYSC)
- Battery alarm (BLAL)
- Override selection (OVSL)
- No warning history (NOHIS)
- TCP/IP (TCPIP, TCPIP_HL)
- RSR, PNS
- Auto dimming of iPendant (BLNK)
- Alarm do not set FAULT output (ER_NOALM)
- Error table (ER_TBL)

6 PLUGIN DEVELOPMENT OVERVIEW

End users may use plugin screens that has been provided by 3rd party suppliers so it becomes possible to change the setting of the equipment, and to display the status of the equipment.

The plugin SDK (Software Development Kit) for plugin software development contains a working example of the plugin installation package as well as its source code before building including plugin screen and plugin instruction (FANUC_SmplGrp). This example shows how a plugin instruction (TP program) and a plugin screen can use KAREL variables to share the index information of the Digital Input/Output ports.

This chapter explains the overview of plugin software development along with the working example included in the plugin SDK.

6.1 KAREL VARIABLES REFERENCED IN PLUGIN SCREEN

The setup screen in this example sets the indices of digital input/output ports. The index values are stored in the KAREL variables in the CMOS memory (non-volatile memory). The TP program uses the KAREL variables to read/write the specified digital input/output ports. Plugin instruction (TP program) will be explained in a later chapter.

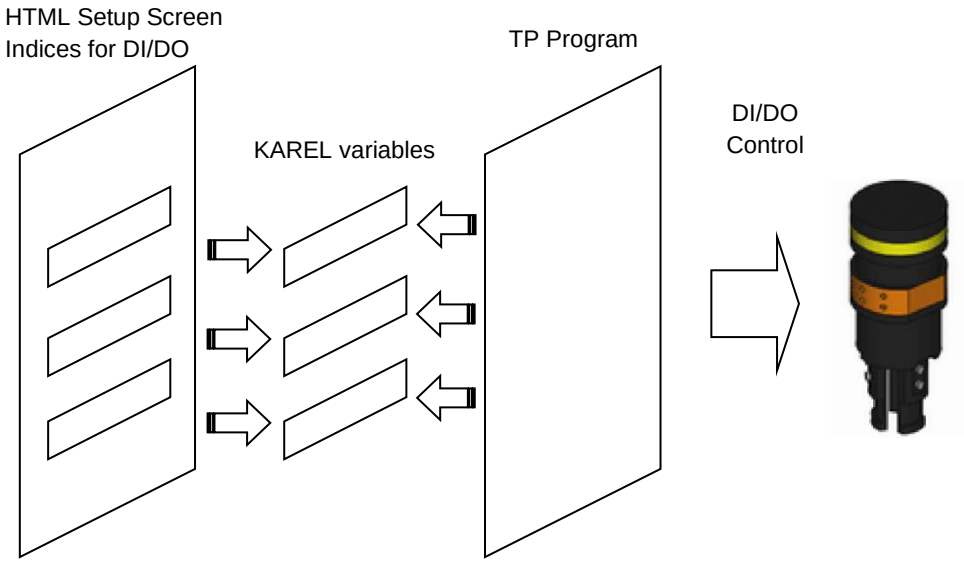


Fig 6.1 (a) TP program references indices of Digital Input/Output ports via KAREL variables

The following table shows the setup items and KAREL variables corresponding to each item to store the setup values in plugin screen in this example.

Table 6.1 (a) Setup Items and KAREL variable definitions

Setup Item	[KAREL program] KAREL variable	Data Type
Connection Interface	[IPL_FANUC_SmplGrp_Cfg] SelSig	INTEGER
Gripper Open (DO)	[IPL_FANUC_SmplGrp_Cfg] OpenDo	INTEGER
Gripper Close (DO)	[IPL_FANUC_SmplGrp_Cfg] ClosDo	INTEGER
Gripper Status (DI)	[IPL_FANUC_SmplGrp_Cfg] StatDi	INTEGER

Please see the following source file of the KAREL program example to define the above KAREL variables.

- ¥PluginSDK¥FANUC_SmplGrp¥kl_ls¥IPL_FANUC_SmplGrp_Cfg.kl (KAREL variables definition file)

Note
- Plugin ID must be defined as "(Company Name)_(Plugin Name)", and the file name of TP program and KAREL program must be "IPL_(Plugin ID)_(Arbitrary program name)". For example, if you define the Plugin ID as **"FANUC_SmplGrp"**, the program name must be "**IPL_FANUC_SmplGrp_*****.TP" and "**IPL_FANUC_SmplGrp_*****.PC". Here "***" indicates arbitrary program name.
- The parameter instruction in TP program allow the number of chars up to 30 or less for parameter name. In case to specify KAREL variable, KAREL program name and KAREL variable name must be specified in this instruction as below. Plugin ID, KAREL program name, and KAREL variable name must be defined in accord with this limitation.

```
Ex) R[10001]=$[IPL_FANUC_SmplGrp_Cfg]OpenDo
                1234567890123456789012345678 90
```

6.2 PLUGIN SCREENS IMPLEMENTATION

The providers of the plugin software can develop screens specifically for the equipment to do the setting up or status monitoring. Since the Tablet TP uses the HTML based screens on the Web browser, one with general knowledge of Web (HTML, JavaScript, CSS) can develop a screen for the tablet. With the iHMI components that FANUC provides, the development of iHMI screens can be even easier without having to create the HTML, JavaScript and CSS codes from scratch. Screen developers can create a *stm (*htm) file by creating and building a base file (*ishtm file) in which the iHMI Component's unique tag <ihmi type="part name"> is embedded.

The plugin SDK contains an implementation example of plugin screen and the details screen of plugin instruction with iHMI component (FANUC_SmplGrp). This example uses JavaScript APIs to read/write KAREL variables stored in the robot controller. Please see the following base files.

- ¥PluginSDK¥FANUC_SmplGrp¥src¥Sample_Config.ishtm (Base file of the plugin screen)
- ¥PluginSDK¥FANUC_SmplGrp¥src¥Sample_TP_detail.ishtm (Base file of the details screen of plugin instruction)

Please refer to chapter 7 "Plugin Screen creation" for the detail of development instruction and the build tool, and section 7.5 "JavaScript APIs to interface with the robot controller" for the interface APIs with the robot controller.

Note
The file name of the base file (*ishtm) in which iHMI Component is embedeed and *stm file (*htm file) which is generated by building the base file is arbitrary. Alphanumeric and underscore "_" are allowed and up to 42 char or less.

6.3 DICTIONARY FILES IMPLEMENTATION

A dictionary file (*.utx) must be generated to switch the display language according to the language setting. In the plugin screen, you can create dictionary files for languages listed in Table 6.3 (a). By creating and building the dictionary files for each language and base file (*.ishtm) which contains dictionary elements defined in the dictionary files, *.stm (*.htm) file with a specific suffix is generated. Please refer to section 7.1.3 “Dictionary Files” for switching the display language.

The dictionary file, that the screen base file refers, must have at least English dictionary (*eg.utx) as the default language. As required, a different language dictionary file may be added as optional. In order to switch languages according to the language setting on robot controller, it is necessary to use the correct file suffix for the file name as specified below in the following table.

File name: “Arbitrary Name” + **.utx (where “**” represents the two letters to indicate the country)

Table 6.3 (a) Language and File Name Suffix

Language	File Name Suffix	File Name	Character code
English	eg	“Arbitrary Name” + **eg**.utx	iso-8859-1
Japanese	kn	“Arbitrary Name” + **kn**.utx	Shift_JIS
Chinese	ch	“Arbitrary Name” + **ch**.utx	GB2312
Taiwanese	tw	“Arbitrary Name” + **tw**.utx	chinese-big5
German	gr	“Arbitrary Name” + **gr**.utx	iso-8859-1
French	fr	“Arbitrary Name” + **fr**.utx	iso-8859-1
Italian	it	“Arbitrary Name” + **it**.utx	iso-8859-1
Spanish	sp	“Arbitrary Name” + **sp**.utx	iso-8859-1

Please see the following example of the dictionary files (*utx) included in the plugin SDK. Non-English dictionary files are optional.

- ¥PluginSDK¥FANUC_SmplGrp¥src¥Sample_dicteg.utx (English dictionary file)
- ¥PluginSDK¥FANUC_SmplGrp¥src¥Sample_dictkn.utx (Japanese dictionary file)
- ¥PluginSDK¥FANUC_SmplGrp¥src¥Sample_dictch.utx (Chinese dictionary file)

6.4 EXAMPLE OF PLUGIN SCREEN

The following screenshot shows how plugin screen functions. In this example, you can select the connection interface at the top of this screen, and set the index number for digital input/output port for the controller I/O interface at the middle part of this screen. The buttons at the bottom of this screen are available for commissioning of the gripper open/close.

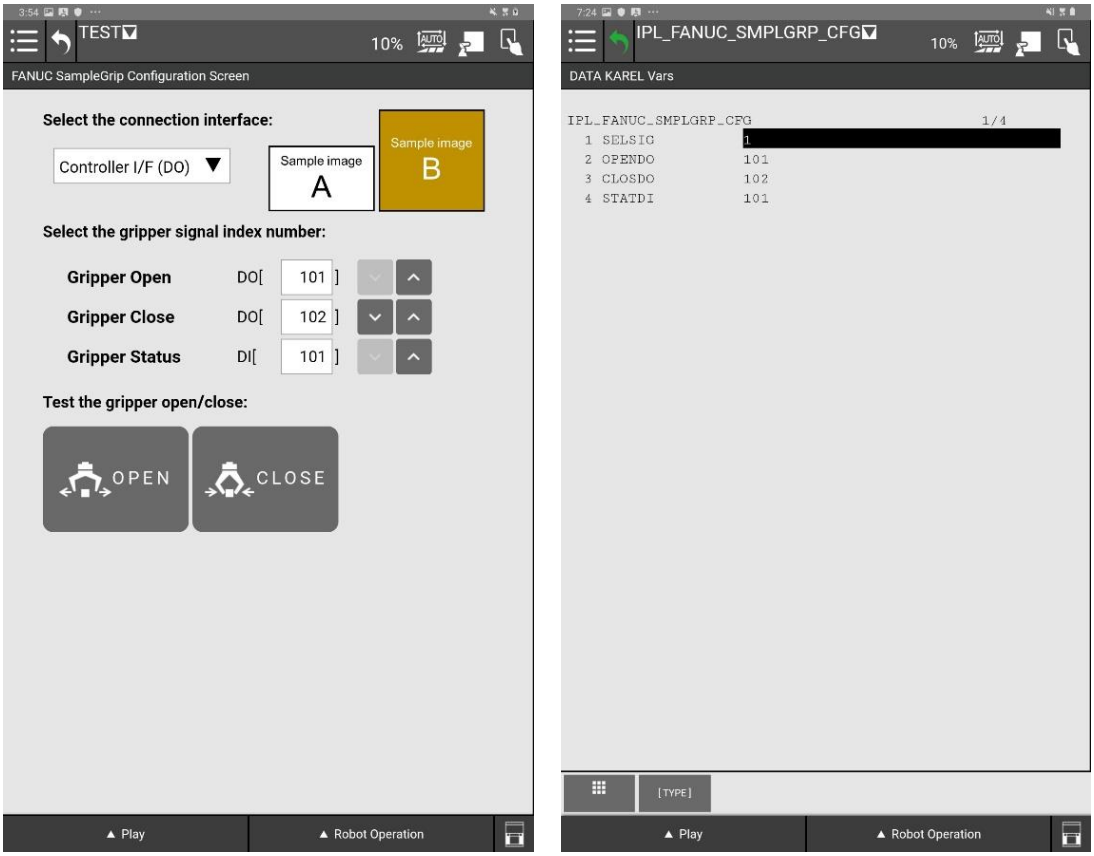


Fig 6.4 (a) Plugin screen example and corresponding KAREL variables

7 PLUGIN SCREEN CREATION

This chapter explains how to create a plugin screen on a Tablet TP. On the Tablet TP, the screen is displayed with a web browser. Therefore, screen development requires users to have general web knowledge (HTML, JavaScript, CSS).

With iHMI components, it is not necessary to create all the code required for screen development in HTML, JavaScript, and CSS therefore iHMI design screen development can be proceeded smoothly, although screen development does not necessarily need to use iHMI components.

7.1 BASIC INFORMATION

This chapter describes how to create a plugin screen on a Tablet TP.

7.1.1 Target Web Browsers

On the Tablet TP, the screen is displayed with a web browser. The target browsers are as follows.

- Chrome (Windows Tablet)
- Chrome (iOS)
- Chrome (Android)
- Web View (Web view browser in Android apps)

7.1.2 File Types

The typical files required for screen creation are as follows.

Table 7.1.2 (a) File types

File Category	File types (extension)	Browser
Display and operation	*.STM	HTML web page using Server Side Includes.
	*.ISHTM	STM base file. By building, it is converted to an STM file.
	*.HTM	HTML web page not including Server Side Includes.
	*.IHTML	HTM base file. By building, it is converted to an HTM file.
	*.JS	Script that dynamically updates content.
	*.CSS	Specifying the style of web page.
Image	*.PNG	Image file. Effective if you need to use an image that makes use of transparency.
	*.GIF	Image file. Suitable for small graphics such as buttons, icons, banners (images with many pure colors and few details).
	*.JPG	Image file. Suitable for images with fine details and a large number of colors.
Resource	*.UTX	Dictionary file. Necessary to switch languages according to settings. For details, refer to section 7.3.2 “Generating dictionary files”.
KAREL	*.VR	KAREL variable file.
	*.PC	KAREL program execution format file.

7.1.3 Dictionary Files

In order to switch languages according to the language setting on the robot controller, it is necessary to name the file name with a specific suffix.

Table 7.1.3 (a) Language and File Name Suffix

Language	File Name Suffix	File Name	Character code
English	eg	"Arbitrary Name" + **eg**.utx	iso-8859-1
Japanese	kn	"Arbitrary Name" + **kn**.utx	Shift_JIS
Chinese	ch	"Arbitrary Name" + **ch**.utx	GB2312
Taiwanese	tw	"Arbitrary Name" + **tw**.utx	chinese-big5
German	gr	"Arbitrary Name" + **gr**.utx	iso-8859-1
French	fr	"Arbitrary Name" + **fr**.utx	iso-8859-1
Italian	it	"Arbitrary Name" + **it**.utx	iso-8859-1
Spanish	sp	"Arbitrary Name" + **sp**.utx	iso-8859-1

By following the above naming convention, the file corresponding to the language setting of the robot controller is automatically selected and displayed. For example, if there are "arc**eg**.stm" and "arc**kn**.stm" installed and English language is selected, then "arc**eg**.stm" is used. However, if the file "arc.stm" without the suffix exists, "arc.stm" is used regardless of the language setting.

Note

- Image file name (*.png, *.gif, *.jpg), plugin screen data file name (*.ishtm, *.ihtml, *.stm, *.htm, *.js, *.css), and dictionary file name (*.utx) are all arbitrary. Alphanumeric and underscore "_" are allowed and up to 42 char or less.
- Plugin ID must be defined as "(Company Name)_(Plugin Name)", and the file name of TP program and KAREL program must be "IPL_(Plugin ID)_(Arbitrary program name)". For example, if you define the Plugin ID as **"FANUC_SmplGrp"**, the program name must be "**IPL_FANUC_SmplGrp_*****.TP" and "**IPL_FANUC_SmplGrp_*****.PC". Here "***" indicates arbitrary program name.
- Do not include the suffix in the link.
  (Correct) <link rel="stylesheet" type="text/css" href="arc.css"/>
  (Incorrect) <link rel="stylesheet" type="text/css" href="arc**eg**.css"/>

7.2 BUILD TOOL

Follow the steps below to create a screen using the build tool (build_ihmi.bat).

1. **Environmental preparation**
   Prepare the environment necessary to use the build tool.

2. **Creating a base file**
   Create a base file (ishtm/ihtml file). In the base file, the iHMI component to be used is embedded with the original tag <ihmi type = "~">.

3. **Build**
   Execute the build tool to expand the original tag embedded in the base file.

Note

Please refer to chapter 3 "INSTLLATION AND MANAGEMENT" for details on how to load and display the software on the robot controller.

7.2.1 Environmental Preparation

Prepare an environment in which Perl runs on Windows in order to use the screen creation tool. The operations described in this chapter have been confirmed with the following software.

StrawberryPerl
http://strawberryperl.com/

After installation, enter "perl -v" on the command prompt to confirm that the perl version is displayed.

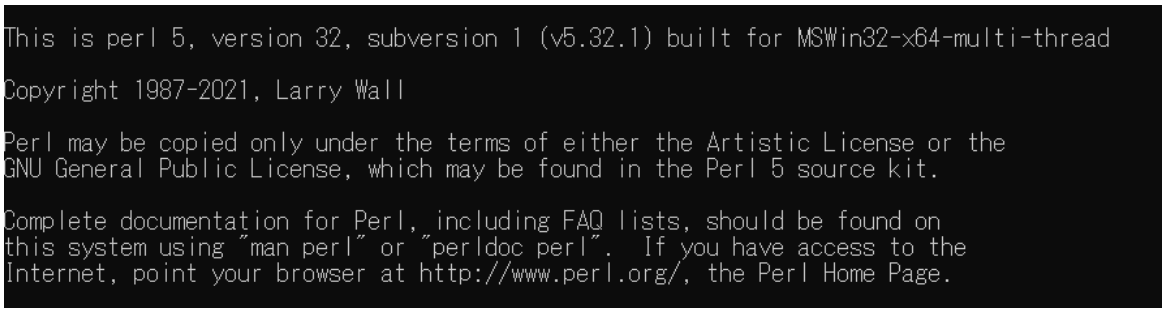


Fig. 7.2.1 (a) Results when commanding perl -v

Note
The screen creation tool has been tested on computers running the 64-bit versions of Microsoft Windows 7 and Windows 10.

Note
If the above screen is not displayed, add the following description to the environment variable Path.
C:¥(Install folder)¥bin;
C:¥(Install folder)¥site¥bin¥
Please specify "C:¥(Install folder)" accordingly with the directory where Perl is installed.

7.2.2 Creating a Base File

The folder structure of screen creation tool is described below. Screen developers can create screens by creating files in the “src” folder.

Fig. 7.2.2 (a) Folder structure

File name, older name	Overview
ihmicomponent	iHMI Components relation.
src	Storing of necessary files at the Web screen Please refer to section 7.3 “USING iHMI COMPONENTS” for the implementation method using iHMI components.
build_ihmi.bat	Build file

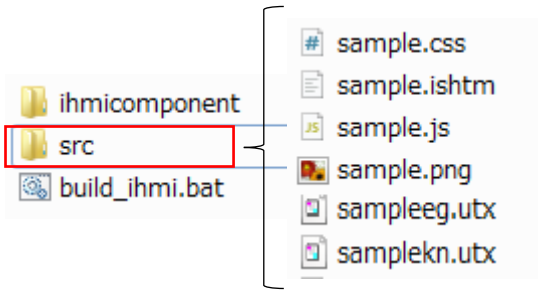


Fig. 7.2.2 (a) Folder structure

Note

- Under the "src" folder, you can create folders up to the first directory level. However, all files in the src folder are generated in the same folder after building. For this reason, when including external files such as css or js in the src folder with ishtm (ihtml), please specify the path in the current directory.
- Tablet TP uses cached data for speeding up unless clearing the cache manually. If developer want to see the updated screen without needing the cache clear when installing the plugin that contains updated JavaScript and CSS files, please add a query parameter in the <script> or <link> tag as below, and change the parameter value. The browser loads the updated JavaScript and CSS files as different files from the cached data by changing the parameter value. Here it doesn't matter what the query parameter is as long as the query parameter is written in the form of "x=y".
  Example)
  <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 Build

The following procedure generates the stm (htm) files to incorporate into the robot controller from the base files ishtm (ihtml) and the dictionary files.

1. Double-click Build_ihmi.bat.
2. The stm (htm) file is generated directly under the output folder.
3. At the same time, an output.ipl file that compresses the files in the output folder is also generated.
4. You can install the plugin software to the robot controller using the ipl file generated at the above step.

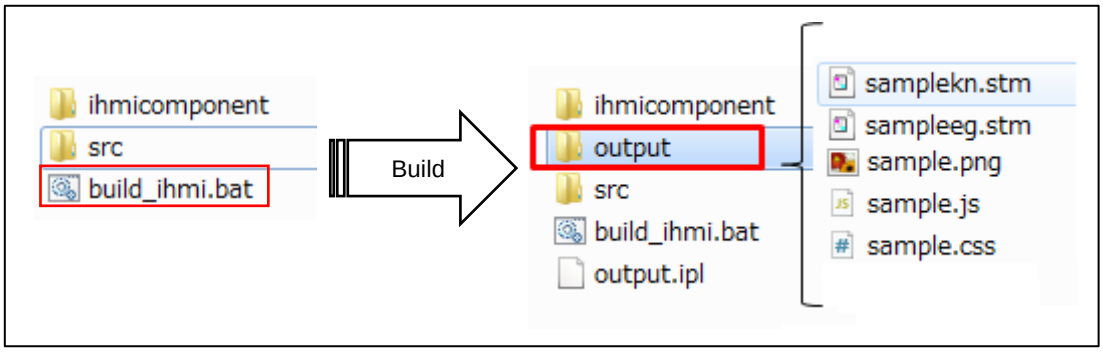


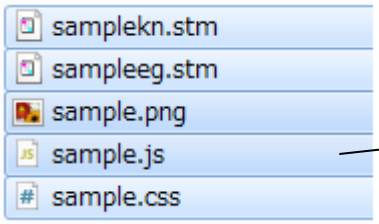
Fig. 7.2.3 (a) Before and after Build

Note

If the following compression error occurs, manually compress all files in the output folder.

```
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 After selecting all the files in the output folder, right-click and execute compression (zip).



Zip compression with all files selected.

2 Change the extension from zip to ipl.

7.3 USING iHMI COMPONENTS

By using iHMI components, iHMI design screen development can proceed smoothly. The iHMI component consists of the following three parts.

1. **HTML template engine**
   By specifying the iHMI component part with the original tag <ihmi>, it is expanded to standard HTML.

2. **CSS framework**
   CSS is provided on the iHMI component side. The style definition for placing the iHMI component parts is necessary, but the style definition of the part itself is not necessary.

3. **JavaScript library**
   A JavaScript library for updating the display and registering a callback function during user operation is provided for each iHMI component.

The following browsers are supported for iHMI components.

- Chrome (Windows 7 or Windows10)
- Chrome (Android OS standard, Chrome browser)
- Chrome (iOS)
- Web View (Web view browser in Android apps)

7.3.1 STM/HTM File Creation

The screen displayed on the Tablet TP consists of multiple frames. The part that the screen developer can customize using iHMI Components is the content area. The screen developer needs to create the stm (htm) file to display in the content area.

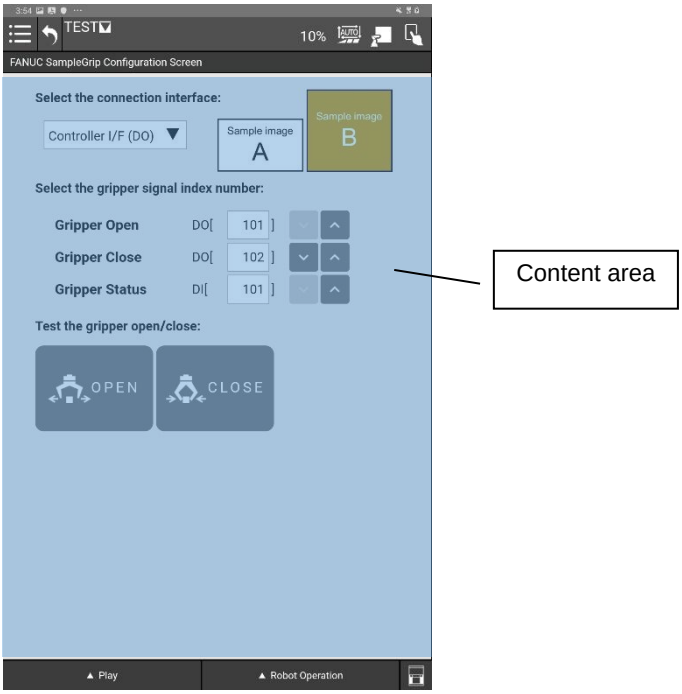


Fig. 7.3.1 (a) Screen structure

Screen developers can create a stm (htm) file by creating and building a base file in which the iHMI Component's unique tag <ihmi type="part name"> is embedded.

In the isthm file, load the iHMI Component's css file (ex. ihmicomponentstyle.css) related to the part to be used before building.

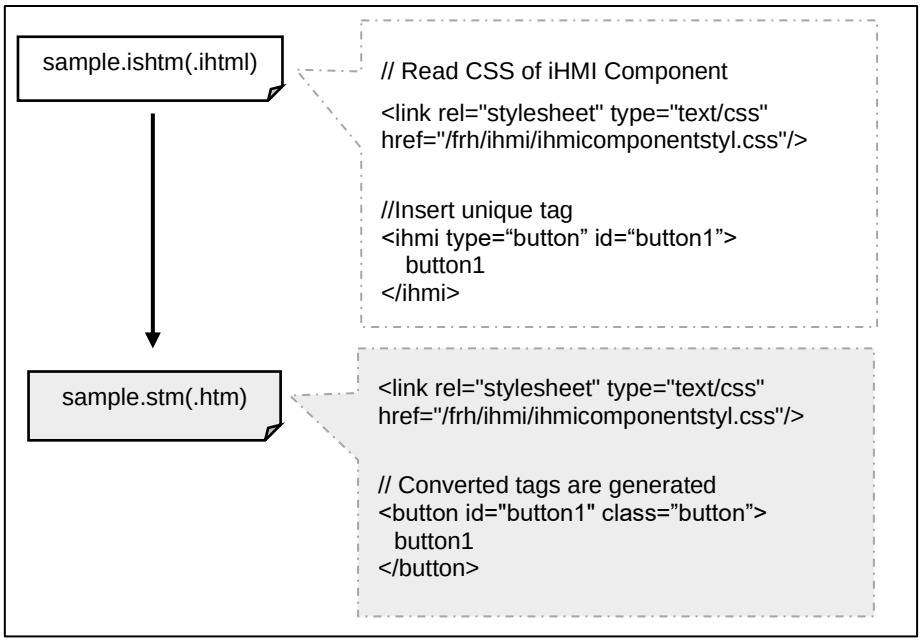


Fig. 7.3.1 (b) Flow of stm (htm) file generation using button part

You can generate a language-specific stm (htm) file by building a dictionary file that describes how to create it in section 7.3.2 "Generating dictionary files".

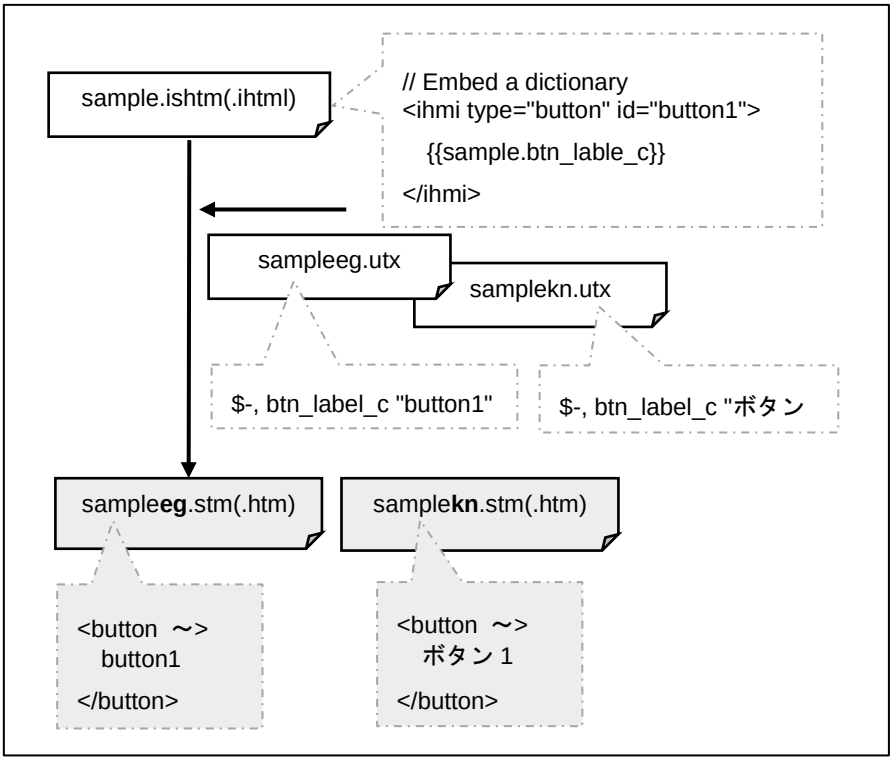


Fig. 7.3.1 (c) Generation of language-specific stm (htm) files

7.3.2 Generating Dictionary Files

A dictionary file must be generated to switch the display language according to the language setting. Please refer to section 7.1.3 “Dictionary Files” for switching the display language.

Describe the dictionary file in the form ""$-",[Dictionary element name] [Translation text]”. Please refer to Fig. 7.3.1 (c).

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

By either of the below methods, you can specify ishtm (ihtml). Embed by specifying the dictionary file name and dictionary element name.

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

Note

- The character code is different for each language. Be careful when editing dictionary files with an editor. Please refer to section 7.1.3 “Dictionary Files” for the character code.
- Embedding dictionary symbols from dictionary files is also available for JavaScript files.

7.3.3 Generating CSS Files

iHMI component provides the following CSS files. The storage location on the robot controller is "/frh/ihmi/".

Table 7.3.3 (a) iHMI Component's css

File name	Overview	Related parts
ihmicomponentstylkn.css ihmicomponentstyleg.css	Style such as color and size	General
ihmicomponentfrm.css	Configuration related of screen	textboxInteger textboxReal textboxString

When the iHMI Components part is specified with the <ihmi> tag, the selector of the common design defined in the CSS file is automatically embedded in the class attribute at build time.

7.3.4 Creating JavaScript Files

Use JavaScript to dynamically update the screen contents.

iHMI Components provides:

- Design and behavior common to all screens, such as the selection color when the button is pressed, and the removal of the focus frame.
- Member functions for controlling screen parts, such as Enable/Disable switching, dynamic change of labels and colors.

The basic flow from user operation to screen update is as follows.

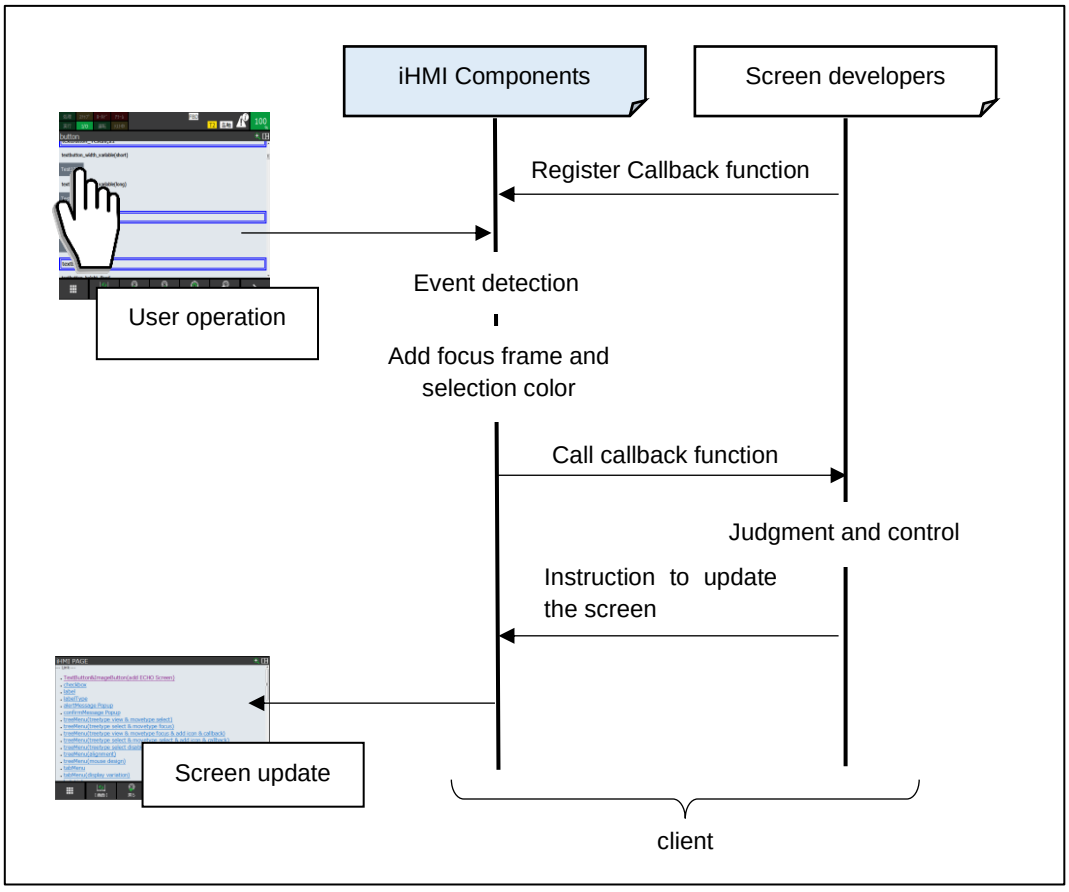


Fig. 7.3.4 (a) Screen update sequence

When you call the member function provided by the iHMI Components part, you can call it like the following

{DOM element obtained in getElementById()}.{member function()}

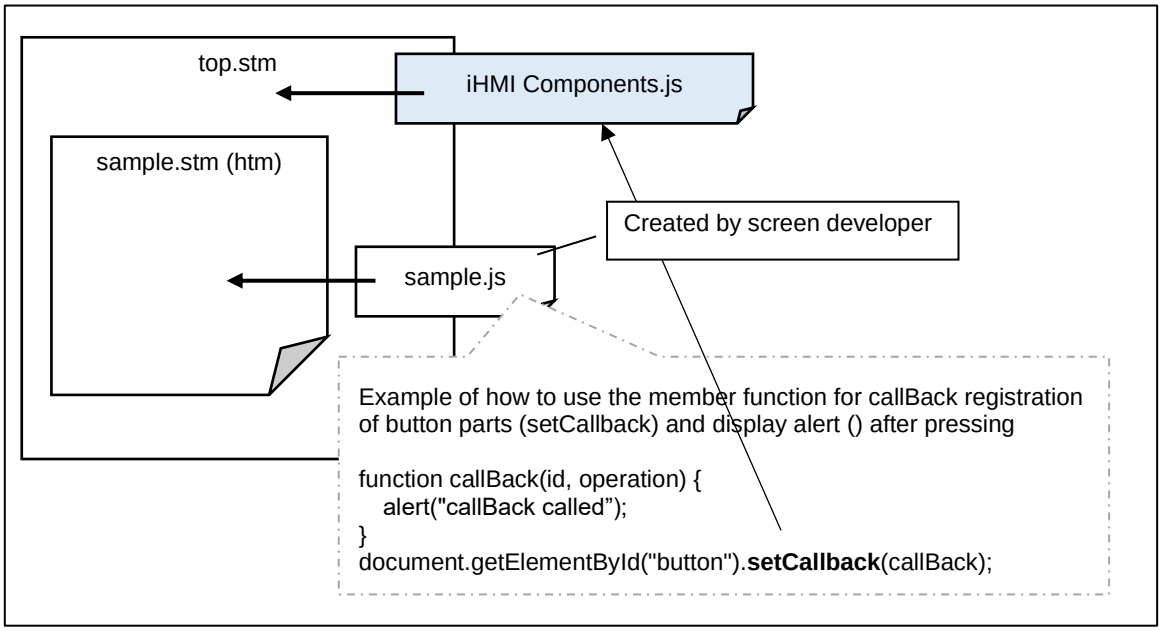


Fig. 7.3.4 (b) How to call the component member functions

When ishtm (ihtml) file is converted to stm (htm) file, the code that links the iHMI Components part operation function is embedded in the DOM element embedded in the <ihmi> tag. Because there is the initialization process, you can call the above. Call member functions after loading HTML.

By registering a callback function to the iHMI components, the screen detects an event of user operation. Fig 7.3.4 (a) shows an example to detect a user event you can know that a user operation has occurred.

7.4 IHMI COMPONENTS

The following iHMI component parts are available.

Table 7.4 (a) iHMI component

Part name	Overview
button	Button
textboxInteger	Int type text box
textboxReal	Real number type text box
textboxString	String type text box
listSelect & listSelectOption	List selection. Selecting one from multiple items
select	Drop down button

This chapter describes the settings of the HTML template engine for each component and JavaScript functions to operate the components.

Note

Be sure to use the I/F of iHMI components instead of directly manipulating the DOM for elements created using Components. The display size may differ under different resolutions.

Example) Setting a value in the textbox input form.
(HTML)
<ihmi type="textboxInteger"></ihmi>
(NG)
document.getElementById(textboxInteger").val = "123"; //Standard properties
(OK)
document.getElementById(textboxInteger").refresh("123", ~); //iHMI component I/F

7.4.1 button

Controller Overview

Design
touch / mouse 
Overview
Button
Note
Buttons that can add text and images

HTML template engine

ihmi tag type			
button			
Parameters			
○: Required △: Option	**Parameter name**	**Type**	**Description**
△	designpattern	String	Selection of design pattern. • “touch”: Design for touch operation. • “mouse”: Design for mouse operation.

△	class	String	An extension class added to the entire <button>.
			Example)
			Box vertical / horizontal size, text size / position, box design pattern.
			
			Note that when adding an extension class, it must be defined as below to make the specificity higher, because it needs to override the default size defined by the iHMI component.
			/* Define extension class "AAA" */
			.button.AAA {
			height ***px;
			}

△	classtext	String	An extension class added to the text in the button.

△	width	Integer	The horizontal width of the button. • The value of the argument width is reflected in the width of the style attribute. • If not specified, the width is variable depending on the length of the set text.
△	height	Integer	The vertical width of the button. The value of the argument height is reflected in the height of the style attribute.
△	image	String	Background image Specify when displaying a background image.
△	imageleft	String	Image Specify either an absolute path or a relative path to display an image on the left side of the text.
△	Value set in the tag <ihmi type="button ">xxx</ihmi>	String	The text in the button.
△	imageright	String	Image Specify either an absolute path or a relative path to display an image on the right side of the text.
△	vertical	Boolean	Specify true only if the image and text are placed vertically.
△	onclick	String	Register callback function When this parameter is specified, the function registered by setCallback() will become invalid.

Note

- In case of specifying a background image, basically an img image is not specified.
- Spriteleft (spriteright) and imageleft (imageright) cannot be specified at the same time.
- In case of specifying the “image”, “imageleft”, and “imageright” arguments, specify either an absolute path or a relative path.
- Use imageleft to specify the image path when the image and text are arranged vertically by specifying the vertical argument.
- 8px padding is fixed on the left and right inside the button.
- Because the <button> tag is used, the contained elements are basically centered horizontally and vertically.
- In case of using an image-only button (button without a label), specify "image-button" in the class argument of the extension class.
- In case of using a button with a side-by-side image and text, if you specify "text-image-button" with the “class” argument of the extension class, the image and text will be positioned vertically and in the center in the vertical direction.
- Do not specify “text-image-button” in the extended class argument “class” for buttons that display images and text vertically, or buttons that specify image position where images and text are aligned horizontally.

JavaScript libraries

refresh

Overview

An update.

Function name

refresh

Parameters

Order	Type	Description
1 text	String	The text in the button. The Components sets this argument to the innerHTML property. Therefore, this argument is interpreted as HTML code.
2 color	String	The text color in the button.

		Specify the property name described in section 7.4.7 “An object that combines multiple color codes” as the value.
3	--	null fixed
4	-	null fixed
5 disabled	Boolean	Button disabled (true) / enabled (false) switching.
6	-	null fixed
Return value		
Variable name	**Type**	**Description**
-	-	-
Note		
Specify null in the argument of the item that does not the refresh. However, null cannot be specified for disabled, be sure to specify either true or false.		

setCallback

Overview		
Register a callback function.		
Function name		
setCallback		
Parameters		
Order	**Type**	**Description**
1 callback	Function	callback function
Parameters (callback)		
Order	**Type**	**Description**
1 id	String	Id name.
2 operation	String	The string “clicked”.
Return value		
Type	**Description**	
-	-	
Note		
If the onclick attribute is specified, the callback function registered with this function does not work.		

setFocus

Overview		
Focus the button.		
Function name		
setFocus		
Parameters		
Variable name	**Type**	**Description**
-	-	-
Return value		
Type	**Description**	
-	-	
Note		
-		

7.4.2 textboxInteger

Controller Overview

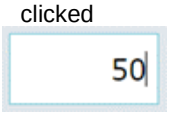
Design
touch



Overview
Integer type text box
Note
For numerical input. You can check the minimum / maximum value of the input value and change the input value with the increase / decrease buttons. • Validation check after input If anything other than the above is entered, an error message “Please enter a numerical value” is displayed. In case of the maximum / minimum value range is exceeded, the error message “(Input value) is out of range. (Minimum value)-(Maximum value)” is displayed. If you use textboxInteger, the minimum and maximum value must be specified with refresh function.

HTML template engine

ihmi tag type			
textboxInteger			
Parameters			
○: Required △: Option	**Parameter name**	**Type**	**Description**
△	designpattern	String	Selection of design pattern. • “touch”: Design for touch operation. • “mouse”: Design for mouse operation.
△	height	String	Specify the height of the text box. • The text position is fixed at the top and bottom center. • This value does not include the border width of two pixels above and below. Example) In case of height = "30px" is specified, the height is 30px+(2px+2px) (border width).
△	width	String	Specify the width of the text box. This value does not include the border width of two pixels above and below and padding on the left and right. Example) If width = ”100px” is specified, • when the design pattern is "touch", the width is 100px + 2px + 16px = 118px • when the design pattern is "mouse", the width is 100px + 2px + 8px = 110px

△	unSelect	String	Do not select the entire character in the text box when clicking • "true": Do not select the entire character when clicked  • Other than "true" or not specified: Select the entire character when clicked. 
△	units	String	The text that appears next to the box.
△	buttons	String	Display increase / decrease buttons • “step” displays one increment / decrement button. • “step 2” displays two increments / decrements button each.
△	step1	Integer	• Pressing the increase / decrease button increases or decreases the input value by the specified value. • If this argument is omitted, The increase / decrease width will be 1
△	step2	Integer	• Pressing the increase / decrease button increases or decreases the input value by the specified value. • When the value of buttons is “step”, the corresponding button is not displayed. • If this argument is omitted, The increase / decrease width is 10 times of step1.
△	errPopPos	Integer	Specifies the hierarchy in which the popup is displayed from the textbox window when a validation error occurs.
△	inputType	String	Specify the type of software keyboard when using a tablet. • “text”: General keyboard for character input (default value) • “number”: For numerical input. • “tel”: For numerical input. Key input of decimal point and negative value is possible.
Note			
By default, the text position in the text box is right-justified.			

The software keyboard specified by “inputType” parameter for textboxInteger are shown below.



“text” “number” “tel”

Fig. 7.4.2 (a) Software keyboard specified by “inputType” parameter for textboxInteger

Note
For textboxInteger, specify inputType=”number” regardless of whether or not the value range allows negative value.

JavaScript libraries

refresh

Overview		
An update.		
Function name		
refresh		
Parameters		
Order	**Type**	**Description**
1 cur	String	The input value in the text box.
2 min	String	Minimum value when checking the input value.
3 max	String	Maximum value when checking the input value.
4 disabled	Boolean	Disable (true) / Enable (false) switching.
Return value		
Type	**Description**	
-	-	
Note		
Input value check • In case of cur is less than min, the decrease button is disabled. • In case of cur is greater than max, the increase button is disabled. • In case of cur is less than min or exceeds max, the character color turns red. • In case of cur is not a number, the input value becomes empty and the background color turns red, and the increase / decrease buttons are disabled.		

setCallback

Overview		
Register callback function.		
Function name		
setCallback		
Parameters		
Variable name	**Type**	**Description**
callback	Function	callback function
Parameters (callback)		
Order	**Type**	**Description**
1 id	String	id name ("id specified for the part".textbox).
2 operation	String	Character string of “changed"
3 value	String	Input value
Return value		
Type	**Description**	
-	-	
Note		
-		

setFocus

Overview
Put focus on the text box
Function name
setFocus

Parameters		
Variable name	**Type**	**Description**
-	-	-
Return value		
Type	**Description**	
-	-	
Note		
-		

getValue

Overview		
Get input value of text box		
Function name		
getValue		
Parameters		
Variable name	**Type**	**Description**
-	-	-
Return value		
Type	**Description**	
-	-	
Note		
-		

setCursorPos

Overview		
The text position in the text box		
Function name		
setCursorPos		
Parameters		
Order	**Type**	**Description**
1 targetStart	Number	Starting position to set the cursor
2 targetEnd	Number	End position to set the cursor If no range is specified, set the same value as the start position. Specify a value greater than or equal to the start position for the end position.
Return value		
Type	**Description**	
-	-	
Note		
-		

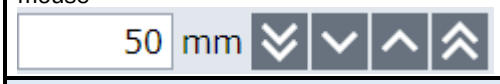
7.4.3 textboxReal

Controller Overview

Design
touch



mouse



Overview
Real number type text box
Note
For numerical input. You can check the minimum / maximum value of the input value and change the input value with the increase / decrease buttons. Validation check after input • You can enter numbers from 0 to 9 and -(minus) and .(dot) • If anything other than the above is entered, an error message “Please enter a numerical value” is displayed. In case of the maximum / minimum value range is exceeded, the error message “(Input value) is out of range. (Minimum value)-(Maximum value)” is displayed.

HTML template engine

ihmi tag type			
textboxReal			
Parameters			
○ : Required △ : Option	**Parameter name**	**Type**	**Description**
△	designpattern	String	Selection of design pattern. • “touch”: Design for touch operation. • “mouse”: Design for mouse operation.
○	decimals	Integer	Number of decimal places that can be entered.
△	height	String	Specify the height of the text box. • The text position is fixed at the top and bottom center. • This value does not include the border width of two pixels above and below. Example) In case of height = "30px" is specified, the height is 30px+(2px+2px) (border width).
△	width	String	Specify the width of the text box. This value does not include the border width of two pixels above and below and padding on the left and right. Example) If width = ”100px” is specified, • when the design pattern is "touch", the width is 100px + 2px + 16px = 118px • when the design pattern is "mouse", the width is 100px + 2px + 8px = 110px
△	unSelect	String	Do not select the entire character in the text box when clicking • "true": Do not select the entire character when clicked  • Other than "true" or not specified: Select the entire character when clicked. 
△	units	String	The text that appears next to the box.

△	buttons	String	Display increase / decrease buttons “step” or “scale” displays one increment / decrement button and “step2” or “scale2” displays two increments / decrements button each.
△	step1	Integer	Pressing the increase / decrease button increases or decreases the input value by the specified value. If this argument is omitted, The increase / decrease width will be 1
△	step2	Integer	• Pressing the increase / decrease button increases or decreases the input value by the specified value. • When the value of buttons is “step”, the corresponding button is not displayed. • If this argument is omitted, The increase / decrease width is 10 times of step1
△	scale1	Integer	• Increase/decrease the input value by multiplying or dividing the input value by this value when the increase/decrease button is pressed • If this argument is omitted, the increment / decrement is the value obtained by multiplying the input value by √2 with multiplication or division
△	scale2	Integer	• Increase/decrease the input value by multiplying or dividing the input value by this value when the increase/decrease button is pressed • If this argument is omitted. The increment / decrement range is the value obtained by multiplying scale1 by the square of multiplied or divided • When the value of buttons is “scale”, the corresponding button is not displayed.
△	errPopPos	Integer	Specifies the hierarchy in which the popup is displayed from the textbox window when a validation error occurs.
△	inputType	String	Specify the type of software keyboard when using a tablet. • “text”: General keyboard for character input (default value) • “number”: For numerical input. • “tel”: For numerical input. Key input of decimal point and negative value is possible.
Note			
By default, the text position in the text box is right-justified.			

The software keyboard specified by “inputType” parameter for textboxReal are shown below.



“text” “number” “tel”

Fig. 7.4.3 (a) Software keyboard specified by “inputType” parameter for textboxReal

Note
- For textboxReal that the value range allows negative value, specify inputType=”tel”.
- For textboxReal that the value range does not allow negative value, specify inputType=”number”.

JavaScript libraries

refresh

Overview		
An update.		
Function name		
refresh		
Parameters		
Order	**Type**	**Description**
1 cur	String	The input value in the text box.
2 min	String	Minimum value when checking the input value.
3 max	String	Maximum value when checking the input value.
4 disabled	Boolean	Disable (true) / Enable (false) switching.
Return value		
Type	**Description**	
-	-	
Note		
Input value check • in case of cur is less than min, the decrease button is disabled. • in case of cur is greater than max, the increase button is disabled. • in case of cur is less than min or exceeds max, the character color turns red. • in case of cur is not a number, the input value becomes empty and the background color turns red, and the increase / decrease buttons are disabled.		

setCallback

Overview		
Register callback function.		
Function name		
setCallback		
Parameters		
Order	**Type**	**Description**
1 callback	Function	callback function
Parameters (callback)		
Order	**Type**	**Description**
1 id	String	id name ("id specified for the part".textbox).
2 operation	String	Character string of “changed"
3 value	String	Input value.
Return value		
Type	**Description**	
-	-	
Note		
-		

setFocus

Overview
Put focus on the text box

Function name		
setFocus		
Parameters		
Order	**Type**	**Description**
-	-	-
Return value		
Type	**Description**	
-	-	
Note		
-		

getValue

Overview		
Get input value of text box		
Function name		
getValue		
Parameters		
Order	**Type**	**Description**
-	-	-
Return value		
Type	**Description**	
String	The input value in the text box. When the decimal is specified, the value according to the number of decimal places will be returned.	
Note		
-		
Restriction		
-		

setCursorPos

Overview		
The text position in the text box		
Function name		
setCursorPos		
Parameters		
Order	**Type**	**Description**
1 targetStart	Number	Starting position to set the cursor
2 targetEnd	Number	End position to set the cursor If no range is specified, set the same value as the start position. Specify a value greater than or equal to the start position for the end position.
Return value		
Type	**Description**	
-	-	
Note		
-		

7.4.4 textboxString

Controller Overview

Part name
touch  mouse 
Overview
String type text box
Note
For strings. By switching the mode, the contents to check the input value can be switched. The checking details under default settings are as follows. • Validation check after input 「& ‘ “ / ￥」Cannot be entered. If anything other than the above is entered, an error message “The following characters cannot be used” is displayed. • Minimum number of characters One character. • A half-width character is counted as one character and a full-width character is counted as two characters.

HTML template engine

ihmi tag type			
textboxString			
Parameters			
○ : Required △ : Option	**Parameter name**	**Type**	**Description**
△	designpattern	String	Selection of design pattern. • “touch”: Design for touch operation. • “mouse”: Design for mouse operation.
○	mode	String	Input value check mode. • Check mode names prepared by default are"dataname","datanamefilter","comment","path","filepath", and"allallowed”. For details, refer to “setMode” described later in JavaScript libraries. • When using user mode, an arbitrary mode name. For details, refer to “setConfig”.
△	height	String	Specify the height of the text box. • The text position is fixed at the top and bottom center. • When the height of the text box is specified, the increase / decrease buttons are also adjusted to the same height. • This value does not include the border width of two pixels above and below. Example) In case of height = "30px" is specified, the height is 30px+(2px+2px) (border width).

△	width	String	Specify the width of the text box. This value does not include the border width of two pixels above and below and padding on the left and right. Example) If width = ”100px” is specified, • when the design pattern is "touch", the width is 100px + 2px + 16px = 118px • when the design pattern is "mouse", the width is 100px + 2px + 8px = 110px
△	errPopPos	Integer	Specifies the hierarchy in which the popup is displayed from the textbox window when a validation error occurs.
Note			
By default, the text position in the text box is right-justified.			

JavaScript libraries

refresh

Overview		
An update.		
Function name		
refresh		
Parameters		
Order	**Type**	**Description**
1 text	String	The input value in the text box.
2 maxLen	Number	Maximum number of characters. A half-width character is counted as one character and a full-width character is counted as two characters.
3 disabled	Boolean	Disable (true) / Enable (false) switching.
Return value		
Type	**Description**	
-	-	
Note		
When the minimum number of characters is 1 or more and the number of characters in text is 0, the background color is changed.		

setConfig

Overview		
Set user-specified key input check conditions, input value check conditions, and minimum number of characters.		
Function name		
setConfig		
Parameters		
Order	**Type**	**Description**

1 filter	Function	Please create a function that always returns true and set the address of that function.
		
		function dummyFunc(keyCode) {
		  return true;
		}

2 validate	Function	Function used for input value check.
		
		Example: If an error occurs when inputting 7, 8, 9,
		function orgValidate(text) {
		  if ((/[789]/).test(text)) {
		  return new top.Error("validattion error");
		  }

		return text; }
3 minLen	Number	Minimum number of characters

Return value	
Type	**Description**
-	-

Note
When the minimum number of characters is 1 or more and the number of characters in text is 0, the background color is changed.

setMode

Overview
Change key input check conditions, input value check conditions, and minimum number of characters settings depending on the mode

Function name
setMode

Parameters		
Order	**Type**	**Description**
1 mode	String	Input restriction mode name.

Return value	
Type	**Description**
-	-

Note

Check condition of each mode.

comment

- Validation check after input
  「&‘ “/ ￥」are cannot be entered.

  If anything other than the above is entered, an error message “The following characters cannot be used” is displayed.
- Minimum number of characters
  Zero character.

dataname

- Validation check after input
  When a character other than half-width characters “a to z, A to Z, 0 to 9, half-width kana katakana (including-,",゜)” is entered, an error message “Invalid characters are used” is displayed.
- Minimum number of characters
  One character.

datanamefilter

- Validation check after input
  When a character other than half-width characters “a to z, A to Z, 0 to 9, space, half-width kana katakana (including-,",゜)” is entered, an error message “Invalid characters are used” is displayed.
- Minimum number of characters
  Zero character.
- Other
  In case of a full-width space is included, convert it to a half-width space.

path (Example of input MC:ihmi/)

- Validation check after input
  When a character other than half-width characters “a to z, A to Z, 0 to 9, : (colon) , ￥(backslash), kana katakana (including-,",゜) is entered, an error message “Invalid characters are used” is displayed.
  In case of the half-width characters “a to z, A to Z, 0 to 9” are not entered in the format with 2 or 3 characters followed by: (half-width colon), an error message “Inappropriate device name” appears.
- Minimum number of characters
  One character.
- Other

In case of ¥ (half-width backslash) is not added at the end of the input value, add ¥.

filepath (Example of input MC:ihmi/ihmipate.stm)

- Validation check after input
  When a character other than half-width characters “a to z, A to Z, 0 to 9, : (colon) , ¥ (backslash), .(period), kana katakana (including-,",゜) is entered, an error message “Invalid characters are used” is displayed.
  In case of the half-width characters “a to z, A to Z, 0 to 9” are not entered in the format with 2 or 3 characters followed by: (half-width colon), an error message “Inappropriate device name” appears.
- Minimum number of characters
  One character.

allallowed

- Validation check after input
  N/A
- Minimum number of characters
  Zero character.

setCallback

Overview		
Register callback function.		
Function name		
setCallback		
Parameters		
Order	**Type**	**Description**
1 callback	Function	callback function
Parameters (callback)		
Order	**Type**	**Description**
1 id	String	Id name.
2 operation	String	Character string of “changed"
3 value	String	Input value
Return value		
Type	**Description**	
-	-	
Note		
-		

setFocus

Overview		
Put focus on the text box		
Function name		
setFocus		
Parameters		
Order	**Type**	**Description**
-	-	-
Return value		
Type	**Description**	
-	-	
Note		
-		

getValue

Overview
Get input value of text box

Function name		
getValue		
Parameters		
Order	**Type**	**Description**
-	-	-
Return value		
Type	**Description**	
String	The input value in the text box.	
Note		
-		

setCursorPos

Overview		
The text position in the text box		
Function name		
setCursorPos		
Parameters		
Order	**Type**	**Description**
1 targetStart	Number	Starting position to set the cursor
2 targetEnd	Number	End position to set the cursor If no range is specified, set the same value as the start position. Specify a value greater than or equal to the start position for the end position.
Return value		
Type	**Description**	
-	-	
Note		
-		

7.4.5 listSelect & listSelectOption

Controller Overview

Design

touch



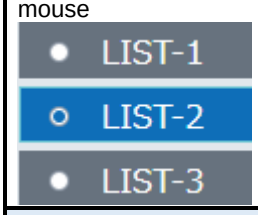
touch



mouse



mouse



Overview

- Controller for selecting one from multiple items.
- Create with two types of ihmi tags: listSelect (outer frame of the list) and listSelectOption (item in the list)

Note

-

HTML template engine

ihmi tag type			
listSelect			
Parameters			
○ : Required △ : Option	**Parameter name**	**Type**	**Description**
△	designpattern	String	Selection of design pattern. • “touch”: Design for touch operation • “mouse”: Design for mouse operation
△	class	String	• An extension class added to the entire list. • The height of the listSelect may be specified by adding an extention class. Note that it must be defined as below to make the specificity higher when adding an extention class, because it needs to override the default size defined by the iHMI component. /* Add an extiontion class AAA */ .list-select.AAA { height: ***px;

			}
△	icon	Boolean	• In case of “false” is specified, the ON / OFF icon is not displayed. • In case of “true (other than false)” is specified, the ON / OFF icon is displayed.
△	horizontal	Boolean	• Specify true only when list items are arranged side by side. • The width of the button is set to be equal to the width of the parent block.
△	Value set in the tag <ihmitype="listSelect"> xxx</ihmi>	String	• <ihmi type="listSelectOption"> tag.
Note			
-			

ihmi tag type			
listSelectOption			
Parameters			
○ : Required △ : Option	**Parameter name**	**Type**	**Description**
○	value	String	List item value.

△	class	String	• An extension class to be added to the item.
			• In case of side by side (horizontal=”true”) is specified, do not specify the selector (padding or margin) that adjusts the width.
			• The height of the listSelect may be specified by adding an extention class. Note that it must be defined as below to make the specificity higher when adding an extention class, because it needs to override the default size defined by the iHMI component.
			/* Add an extiontion class AAA */
			.list-select .AAA {
			height: ***px;
			}

△	width	Integer	The horizontal width of the button.
△	Value set in the tag <ihmi type="listSelectOption"> xxx</ihmi>	String	Text in the item
Note			
-			

JavaScript libraries

refresh

Overview		
An update		
Function name		
refresh		
Parameters		
Order	**Type**	**Description**
1 options	Object	An array of list items.

		• In case of refresh () is called with this parameter specified when side by side is specified (horizontal="true"), call with setWidth () to adjust the width of the item. • In case of this parameter is specified, the original list item is deleted, and the list item is updated with this value.
2 selected	String	Specify the list item to be set to selected state in the initial display by the value of the argument value.
3 disabled	Boolean	Switching between disable (true) / Enable (false)
Parameters (options)		
Order	**Type**	**Description**
1 value	String	An array of list items. The value returned from the Components to the callback function is this value.
2 text	String	The text to display for the item.
3 className	String	An extension class for item.
Return value		
Type	**Description**	
-	-	
Note		
Specify null in the argument of the item that does not the refresh. However, null cannot be specified for disabled, be sure to specify either true or false.		

setWidth

Overview		
Recalculate the width of the list.		
Function name		
setWidth		
Parameters		
Order	**Type**	**Description**
-	-	-
Return value		
Type	**Description**	
-	-	
Note		
Call this function under the list is created by refresh () under the side-by-side setting. The width of the list and the width of the text in the list are recalculated by this function.		

setCallback

Overview		
Register callback function.		
Function name		
setCallback		
Parameters		
Order	**Type**	**Description**
1 callback	Function	callback function
Parameters (callback)		
Order	**Type**	**Description**
1 id	String	Id name.
2 type	String	Character string of "changed"
3 value	String	The value of the selection item.

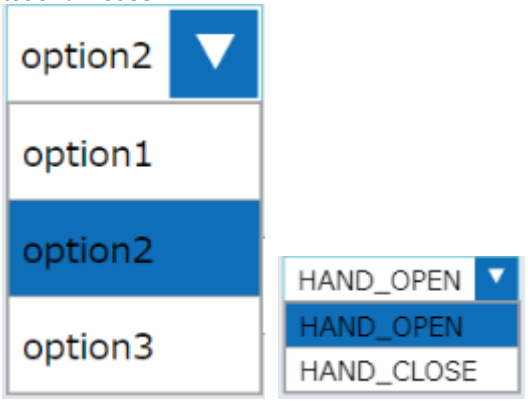
Return value	
Type	**Description**
-	-
Note	
-	

getValue

Overview		
Get currently selected value.		
Function name		
getValue		
Parameters		
Order	**Type**	**Description**
-	-	-
Return value		
Type	**Description**	
String	Currently selected value.	
Note		
-		

7.4.6 select

Controller Overview

Design
touch / mouse 
Overview
• Select box for selecting input items from multiple candidates. • Create with two types of ihmi tags: select (outer frame, expand button) and selectOption (item)
Note
-

HTML template engine

ihmi tag type			
select			
Parameters			
○ : Required △ : Option	**Parameter name**	**Type**	**Description**
△	designpattern	String	Selection of design pattern. • “touch”: Design for touch operation. • “mouse”: Design for mouse operation.

△	width	String	The width of the select box. It is not the width of the entire box. Width excluding expand button, padding, margine, border.
△	class	String	Class to be added to the label in the select box.
Note			
-			

ihmi tag type			
selectOption			
Parameters			
○ : Required △ : Option	**Parameter name**	**Type**	**Description**
○	value	String	The value of the selection item.
△	class	String	Class to be added to the selection item.
△	image	String	An image to display to the left of the value of the selected item. Specify as a relative or absolute path.
△	Value set in the tag <ihmi type="selectOption ">xxx</ihmi>	String	The text of the selection item.
Note			
-			

JavaScript libraries

refresh

Overview		
An update.		
Function name		
refresh		
Parameters		
Order	**Type**	**Description**
1 options	Object	Selection item. • In case of the selected item has already been generated in HTML, a list will be added to the generated item. In case of there is no need to add, set the empty array “[]”. Example 1) Text + value of the selection item + image: ["text1", 0 , "AAA.png"], ["text2", 1 , "BBB.png"], Example 2) In case of omitting the list item number, specify an array of text only: ["addoption4","addoption5","addoption6"] • In case of there is no need to add, set the empty array “[]”.
2 selectedValue	Number String	Selected item value (list item number or text to display in list). • The list item number is the second argument of the argument “options”. • The text displayed in the list is the first argument of the argument "options".
3 selectedText	String	To indicate that the previously selected value was deleted when the previously selected value disappeared from the selection item by some other operation. Example) In case of you select a data name with this controller and then delete the data, that data name is no longer selected. In such cases, the text representing the selected value (selectedText) is displayed in red.

4 disabled	boolean	Disable (true) / Enable (false) switching.
Parameters (options)		
Order	**Type**	**Description**
1	String	The text of the selection item.
2	Number String	The value of the selection item. • If this argument is omitted, the value of the first argument will be used as the value of the selected item • The value returned from the Components by the callback function is this value.
3	String	The image to display in the list. • This argument can be omitted. • The image size is defined by the Components CSS as follows. designpattern=”touch” : 40px designpattern=”mouse” : 26px
Return value		
Type	**Description**	
-	-	
Note		
-		

getValue

Overview		
Get currently selected value.		
Function name		
getValue		
Parameters		
Order	**Type**	**Description**
-	-	-
Return value		
Type	**Description**	
Number String	Currently selected value.	
Note		
-		

setCallback

Overview		
Register callback function.		
Function name		
setCallback		
Parameters		
Order	**Type**	**Description**
1 callback	Function	callback function
Parameters (callback)		
Order	**Type**	**Description**
1 id	String	Id name.
2 type	String	Character string of “changed"
3 value	String	Selected value
Return value		
Type	**Description**	
-	-	

Note
-

setFocus

Overview		
Focus the select button		
Function name		
setFocus		
Parameters		
Order	**Type**	**Description**
-	-	-
Return value		
Type	**Description**	
-	-	
Note		
-		

7.4.7 An object that combines multiple color codes

Color table

Overview		
• Common information between controllers • Create a table of colors used between parts. • This table is used when color is changed by using refresh() function.		
Property name and value		
Property name	Color code	Application
red02	#E60012	This color represents negation or danger.
green03	#079A3E	This color represents safety and OK.
orange01	#EB6100	This color indicates stop or cancel.
yellow02	#FFE109	This color indicates attention.
blue01	#0E6EB8	This color indicates that it is selected.
black01	#000000	This color is used for light background text. This color is the color of the letter that indicates a valid state.
white01	#FFFFFF	This color is used for text on a dark background color.
Note		
Parts used in this table: button, textboxInteger, textboxReal		

7.5 JAVASCRIPT APIS TO INTERFACE WITH ROBOT CONTROLLER

JavaScript APIs to interface with the robot controller are available. The APIs allow developers to get and set KAREL variables and I/O values of the robot controller (server). These APIs are loaded under the top frame so can be implemented like “top.ihmi_setVar”.

7.5.1 top.ihmi_setVar

top.ihmi_setVar

Overview
Sets value to the specified KAREL variable.

Function name		
ihmi_setVar		
Parameters		
Name	**Type**	**Description**
1 prog_name	String	KAREL program name
2 var_name	String	KAREL variable name
3 value	String	Setting value
Note		
When setting BOOLEAN type KAREL variable, please specify “1” for TRUE and “0” for FALSE.		

Note
top.ihmi_setVar does not support KAREL variable with variable-sized arrays.

7.5.2 top.ihmi_getVar

top.ihmi_getVar

Overview		
Acquires value of the specified KAREL variable.		
Function name		
ihmi_getVar		
Parameters		
Name	**Type**	**Description**
1 prog_name	String	KAREL program name
2 var_name	String	KAREL variable name
3 callback	Function	Callback function
4 callbackarg1	String	Value to pass as the 5th argument for the callback function
5 callbackarg2	String	Value to pass as the 6th argument for the callback function
Parameters (callback)		
Name	**Type**	**Description**
1 prog_name	String	KAREL program name
2 var_name	String	KAREL variable name
3 type_code	String	Type of the KAREL variable, usually this parameter is not used.
4 value	String	Acquired value
5 callbackarg1	String	Value specified in the 4th argument for ihmi_getVar
6 callbackarg2	String	Value specified in the 5th argument for ihmi_getVar
Note		
-		

Note
top.ihmi_getVar does not support KAREL variable with variable-sized arrays.

7.5.3 top.ihmi_iovalset

top.ihmi_iovalset

Overview		
Sets value to the specified I/O.		
Function name		
ihmi_ iovalset		
Parameters		
Name	**Type**	**Description**
1 io_type	String	I/O type. Specify assigned number for the kind of I/O as followings.

		DO:2, RO:9, GO:19, AO:4
2 index	String	I/O index number
3 value	String	I/O Value
Note		
When setting DO or RO value, please specify “1” for ON and “0” for OFF.		

7.5.4 top.ihmi_iovalrd

top.ihmi_iovalrd

Overview		
Acquires value of the specified I/O.		
Function name		
ihmi_iovalrd		
Parameters		
Name	**Type**	**Description**
1 io_type	String	IO type. Specify assigned number for the kind of I/O as followings. DI:1, DO:2, AI:3, AO:4, RI:8, RO:9, SO:12, GI:18, GO:19
2 index	String	I/O index number
3 callback	Function	Callback function
4 callbackarg1	String	Value to pass as the 4th argument for the callback function
5 callbackarg2	String	Value to pass as the 5th argument for the callback function
Parameters (callback)		
Name	**Type**	**Description**
1 io_type	String	Acquired I/O type
2 index	String	Acquired I/O index number
3 value	String	Acquired I/O value
4 callbackarg1	String	Value specified in the 4th argument for ihmi_iovalrd
5 callbackarg2	String	Value specified in the 5th argument for ihmi_iovalrd
Note		
-		

7.5.5 top.ihmi_setVar2

top.ihmi_setVar2

Overview		
Sets value to the specified KAREL variable.		
Function name		
ihmi_setVar2		
Parameters		
Name	**Type**	**Description**
1 prog_name	String	KAREL program name
2 var_name	String	KAREL variable name
3 value	String	Setting value
4 callback	Function	Callback function
5 callbackarg1	String	Value to pass as the 5th argument for the callback function
6 callbackarg2	String	Value to pass as the 6th argument for the callback function
Parameters (callback)		
Name	**Type**	**Description**
1 prog_name	String	KAREL program name
2 var_name	String	KAREL variable name
3 value	String	Setting value
4 status	String	Status returns zero on success and non-zero on failure.

5 callbackarg1	String	Value specified in the 5th argument for ihmi_setVar2
6 callbackarg2	String	Value specified in the 6th argument for ihmi_setVar2
Note		
When setting BOOLEAN type KAREL variable, please specify “1” for TRUE and “0” for FALSE.		

⚠ Caution
top.ihmi_setVar2 is available on the robot controller software V9.40P/06 or after. If a plugin uses these APIs, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/05 or older.

Note
top.ihmi_setVar2 does not support KAREL variable with variable-sized arrays.

7.5.6 top.ihmi_getVar2

top.ihmi_getVar2

Overview		
Acquires value of the specified KAREL variable.		
Function name		
ihmi_getVar2		
Parameters		
Name	**Type**	**Description**
1 prog_name	String	KAREL program name
2 var_name	String	KAREL variable name
3 callback	Function	Callback function
4 callbackarg1	String	Value to pass as the 6th argument for the callback function
5 callbackarg2	String	Value to pass as the 7th argument for the callback function
Parameters (callback)		
Name	**Type**	**Description**
1 prog_name	String	KAREL program name
2 var_name	String	KAREL variable name
3 type_code	String	Type of the KAREL variable, usually this parameter is not used.
4 value	String	Acquired value
5 status	String	Status returns zero on success and non-zero on failure.
6 callbackarg1	String	Value specified in the 4th argument for ihmi_getVar2
7 callbackarg2	String	Value specified in the 5th argument for ihmi_getVar2
Note		
-		

⚠ Caution
top.ihmi_getVar2 is available on the robot controller software V9.40P/06 or after. If a plugin uses these APIs, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/05 or older.

Note
top.ihmi_getVar2 does not support KAREL variable with variable-sized arrays.

7.5.7 top.ihmi_iovalset2

top.ihmi_iovalset2

Overview		
Sets value to the specified I/O.		
Function name		
ihmi_ iovalset2		
Parameters		
Name	**Type**	**Description**
1 io_type	String	IO type. Specify assigned number for the kind of I/O as followings. DO:2, RO:9, GO:19, AO:4
2 index	String	I/O index number
3 value	String	Setting value
4 callback	Function	Callback function
5 callbackarg1	String	Value to pass as the 5th argument for the callback function
6 callbackarg2	String	Value to pass as the 6th argument for the callback function
Parameters (callback)		
Name	**Type**	**Description**
1 io_type	String	Specified I/O type. DO:2, RO:9, GO:19, AO:4
2 index	String	I/O index number
3 value	String	Setting value
4 status	String	Status returns zero on success and non-zero on failure.
5 callbackarg1	String	Value specified in the 5th argument for ihmi_iovalset2
6 callbackarg2	String	Value specified in the 6th argument for ihmi_iovalset2
Note		
When setting DO or RO value, please specify “1” for ON and “0” for OFF.		

> ⚠ **Caution**
> top.ihmi_iovalset2 is available on the robot controller software V9.40P/06 or after. If a plugin uses these APIs, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/05 or older.

7.5.8 top.ihmi_iovalrd2

top.ihmi_iovalrd2

Overview		
Acquires value of the specified I/O.		
Function name		
ihmi_iovalrd2		
Parameters		
Name	**Type**	**Description**
1 io_type	String	I/O type. Specify assigned number for the kind of I/O as followings. DI:1, DO:2, AI:3, AO:4, RI:8, RO:9, SO:12, GI:18, GO:19
2 index	String	I/O index number
3 callback	Function	Callback function
4 callbackarg1	String	Value to pass as the 5th argument for the callback function
5 callbackarg2	String	Value to pass as the 6th argument for the callback function

Parameters (callback)		
Name	**Type**	**Description**
1 io_type	String	Specified IO type. DI:1, DO:2, RI:8, RO:9, GI:18, GO:19, AI:3, AO:4
2 index	String	I/O index number
3 value	String	Acquired value
4 status	String	Status returns zero on success and non-zero on failure.
5 callbackarg1	String	Value specified in the 4th argument for ihmi_iovalrd2
6 callbackarg2	String	Value specified in the 5th argument for ihmi_iovalrd2
Note		
-		

⚠ Caution

top.ihmi_iovalrd2 is available on the robot controller software V9.40P/06 or after. If a plugin uses these APIs, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/05 or older.

7.5.9 top.frpl_utl.startIOGroupMonitor

top.frpl_utl.startIOGroupMonitor

Overview		
Start I/O monitoring for the specified range of I/O points. When it detects change with I/O value within the specified range, it calls the callback function to notify the screen of the change and its value.		
Function name		
frpl_utl.startIOGroupMonitor(type, index, count, callback, callFrame, callbackarg)		
Parameters		
Name	**Type**	**Description**
1 type	Number	IO type. Specify assigned number for the kind of I/O as followings. DI:1, DO:2, RI:8, RO:9, SO:12
2 index	Number	Specify starting index number of I/O points to monitor.
3 count	Number	Specify the number of I/O points to monitor. The maximum number of I/O points to monitor is up to 500. You cannot monitor I/O ports more than 500.
4 callback	Function	Specify callback function to call when it detects change with I/O value within the specified range.
5 callFrame	Object	Windows object that calls this API
6 callbackarg	String / Number	Value to pass as an argument to the callback function (optional)

Return values		
Name	**Type**	**Description**
1 promise	Promise	Returns respObj as a PromiseResult. PromiseState is as follows. • resolve: Success to start I/O monitor • reject: Failed or error occurs when starting the I/O monitor
Details: Return value (respObj)		
Name	**Type**	**Description**
1 status	String	Returns the results status. See table [Details: Status list] described in this section for details.
2 errorDescription	String	Message explaining reject. Returns “null” when resolved. See table [Details: Status list] described in this section for details.

3 type	Number	Returns the I/O type that you started monitoring. DI:1, DO:2, RI:8, RO:9
4 ioValueArray	Array	Initial value of the I/O points to monitor. Returns initial values of I/O points when starting the monitor as an object for the I/O counts from the starting index number specified in the parameter. See table [Return value (respObj.ioValueArray)] for the details.

Details: Return value (respObj.ioValueArray)

Name	Type	Description
1 index	String	I/O index number.
2 val	String	I/O value. Returns the following values according to the specified I/O type. • I/O value (Number) • I/O status (ON: 1, OFF: 0)

Note

Returns blank for I/O ports failed to get the value due to, for example, unassigned.

Example) Start monitoring for DO[1] to DO[5]: 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 : “” } // Unassigned
]

Details: Callback

Name	Type	Description
1 type	String	I/O type that it detected change in value. DI:1, DO:2, RI:8, RO:9
2 index	String	I/O index number that it detected change in value.
3 value	String	I/O value that it detected change in value. Returns the followings according to the specified I/O type. • I/O value (Number) • I/O status (ON: 1, OFF: 0)
4 callbackarg	String / Number	Argument object specified in the parameter.

Note

Output a console error in case failed to call the callback function due to, for example, the specified callback function is not found.

Details: Status list

status	errorDescription	Description
0	Null	Success
1	RequestError	Failed to send a request to the robot controller.
2	TypeError: Argument (Argument number (parameter name)) is not of type (Correct data type). Example) Argument 1(type) is not of type Number.	Missing necessary arguments or incorrect data type specified.
3	IOTypeError: I/O type (I/O type specified in the argument parameters) does not exist. Example) I/O type 999 does not exist.	Invalid I/O type specified.
4	IORangeError: I/O range does not exist.	Invalid I/O range specified.

5	StartMonitorError: The maximum number of monitors that can be configured has been exceeded.	Exceeds the maximum number of I/O monitors.
6	StartMonitorError: The same argument has already been set.	Detects duplicated I/O monitor.
7	StopMonitorError: The designated monitor does not exist.	Error when stopping I/O monitor that does not exist (the specified I/O monitor has not been started using frpl_utl.startIOGroupMonitor API)

⚠ Caution
top.frpl_utl.startIOGroupMonitor and top.frpl_utl.stopIOGroupMonitor are available on the robot controller software V9.40P/49 or after. If a plugin uses these APIs, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/48 or older.

7.5.10 top.frpl_utl.stopIOGroupMonitor

top.frpl_utl.stopIOGroupMonitor

Overview		
Stop I/O monitoring for the specified range of I/O points.		
Function name		
frpl_utl.stopIOGroupMonitor(type, index, count, callback, callFrame)		
Parameters		
Name	**Type**	**Description**
1 type	Number	IO type. Specify assigned number for the kind of I/O as followings. DI:1, DO:2, RI:8, RO:9, SO:12
2 index	Number	Specify starting index number of I/O points to monitor.
3 count	Number	Specify the number of I/O points to monitor. The maximum number of I/O points to monitor is up to 500. You cannot monitor I/O ports more than 500.
4 callback	Function	Specify callback function to call when it detects change with I/O value within the specified range.
5 callFrame	Object	Windows object that calls this API
Note		
• When stopping I/O monitor, the exact same argument parameters must be specified as the parameters specified when starting the I/O monitor. For example, if you start an I/O monitor with index=1 and count=10, you cannot stop the monitor partially for 1 to 9 specifying the parameters of index=1 and count=9. • When the invoker screen is unloaded, the I/O monitor is automatically stopped by API.		

Return values		
Name	**Type**	**Description**
1 promise	Promise	Returns respObj as a PromiseResult. PromiseState is as follows. • resolve: Success to stop I/O monitor • reject: Failed or error occurs when stopping the I/O monitor
Details: Return value (respObj)		
Name	**Type**	**Description**
1 status	String	Returns the results status. See table [Details: Status list] described in the previous section for details.
2 errorDescription	String	Message explaining reject. Returns “null” when resolved.

		See table [Details: Status list] described in the previous section for details.

Details: Callback		
Name	**Type**	**Description**
1 type	String	I/O type that it detected change in value. DI:1, DO:2, RI:8, RO:9
2 index	String	I/O index number that it detected change in value.
3 value	String	I/O value that it detected change in value. Returns the followings according to the specified I/O type. • I/O value (Number) • I/O status (ON: 1, OFF: 0)
4 callbackarg	String / Number	Argument object specified in the parameter.
Note		
Output a console error in case failed to call the callback function due to, for example, the specified callback function is not found.		

⚠ Caution
top.frpl_utl.startIOGroupMonitor and top.frpl_utl.stopIOGroupMonitor are available on the robot controller software V9.40P/49 or after. If a plugin uses these APIs, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/48 or older.

7.5.11 top.frpl_utl.jumpPage

top.frpl_utl.jumpPage

Overview		
Makes page transition to any of the FANUC Robot standard page. This API has no return value.		
Function name		
frpl_utl.jumpPage		
Parameters		
Name	**Type**	**Description**
1 spId	String / Number	Specify the Softpart ID of the screen $TP_CURSCRN[].$SP_ID provides the Softpart ID number.
2 scrnId	String / Number	Specify the Screen ID of the screen $TP_CURSCRN[].$SCRN_ID provides the Screen ID number.
Note		
• Pressing the Return button located in the Status bar brings you to the previous screen after transiting to any screen except for the Tablet UI Editor screen. • Please refer to section 9.11.6.1 “Software parts ID, Screen ID” in R-30iB Plus operator’s manual (Basic Function) (B-83284EN) for how to get $TP_CURSCRN[].$SP_ID and $TP_CURSCRN[].$SCRN_ID.		

⚠ Caution
top.frpl_utl.jumpPage is available on the robot controller software V9.40P/45 or after. If a plugin uses these APIs, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/44 or older.

7.5.12 top.frpl_utl.getMacAddr

top.frpl_utl.getMacAddr

Overview		
This API returns the MAC address of the robot controller port #1 in STRING data. This API has no argument.		
Function name		
frpl_utl.getMacAddr		
Parameters		
Name	**Type**	**Description**
-	-	-

Return values		
Name	**Type**	**Description**
1 promise	Promise	Returns a respObj as a PromiseResult. PromiseState is the following: • resolve: Success to get the MAC address. • reject: Request failed due to internal error.
Details: Return value when resolved (respObj)		
Name	**Type**	**Description**
1 macaddr	String	Returns the MAC address of the robot controller port #1 in STRING data. Example) 00:e00:e4:80:8f:2d
Details: Return value when rejected (respObj)		
Name	**Type**	**Description**
1 errorCode	String	Returns the error code value if any error occurs.
2 errorDescription	String	Returns the error details as below: • getMacAddr is not available. : The internal program to get MAC address is not found. • Failed to get MAC address with errorCode. : An error has occurred when getting MAC address. • Failed to get expected response. : Failed to send XHR or JSON data could not be obtained successfully.
Note		
-		

> ⚠ **Caution**
> top.frpl_utl.getMacAddr is available on the robot controller software V9.40P/48 or after. If a plugin uses these APIs, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/47 or older.

7.5.13 top.frpl_utl.initPayloadIdentData

top.frpl_utl.initPayloadIdentData

Overview
Initialize the payload Identification data measured by APIs of the payload identification.
Function name
frpl_utl.initPayloadIdentData(grpNum)

Parameters		
Name	**Type**	**Description**
grpNum	String	Group number
Return values		
Name	**Type**	**Description**
promise	Promise	Returns the respObj as PromiseResult PromiseState is as follows. • resolve: Success • reject: Cancel or error
Detail: Return values (respObj)		
Name	**Type**	**Description**
status	String	Returns the result status. See table [Details: Status list] described in this section for details.
errorDescription	String	Message explaining reject. Returns “null” when resolved. See table [Details: Status list] described in this section for details on message.

Note

- Please execute this API when you start the payload identification using the API.
- The data measured by the Utool Payload Identification does not initialized by executing this API.
- The flow of the payload identification of group 1 using the API is as follows.
  1. Initialized the measured data: top.frpl_utl.initPayloadIdentData("1")
  2. Move to position #1
  3. Measure at position #1: top.frpl_utl.recordPayloadIdentData("1")
  4. Move to position #2
  5. Measure at position #2: top.frpl_utl.recordPayloadIdentData("2")
  6. Move to position #3
  7. Measure at position #3: top.frpl_utl.recordPayloadIdentData("3")
  8. Move to position #4
  9. Measure at position #4: top.frpl_utl.recordPayloadIdentData("4")
  10. Identify the payload based on the measured results.
      - Identify as mass unknown: top.frpl_utl.getPayloadIdentResult("0", "0")
      - Identify as mass "10kg": top.frpl_utl.getPayloadIdentResult("1", "10")
  11. Apply the result to No.5 of payload settings in group 1: top.frpl_utl.setPayloadData("1", "5", "comment", m, x, y, z, "0", "0", "0")

Details: Status list		
status	**errorDescription**	**Description**
0	Null	Success
1	RequestError	Failed to send a request to the robot controller.
2	ArgumentError	Missing arguments or type errors.

⚠ Caution

top.frpl_utl.initPayloadIdentData is available on the robot controller software V9.40P/60 or after. If a plugin uses these APIs, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/59 or older.

7.5.14 top.frpl_utl.recordPayloadIdentData

top.frpl_utl.recordPayloadIdentData

Overview
Measure the data at the current position to identify the mass and center of gravity position.

Function name		
frpl_utl.recordPayloadIdentData(posNum)		
Parameters		
Name	**Type**	**Description**
posNum	String	Measured position number. Position #1:1, Position #2:2, Position #3:3, Position #4:4
Return values		
Name	**Type**	**Description**
promise	Promise	Returns the respObj as PromiseResult PromiseState is as follows. • resolve: Success • reject: Cancel or error
Details: Return values (respObj)		
Name	**Type**	**Description**
status	String	Returns the result status. See table [Details: Status list] in this section for details.
errorDescription	String	Message explaining reject. Returns “null” when resolved. See table [Details: Status list] in this section for details.
Note		
• Execute this API at four locations with very different postures.		

Details: Status list		
status	**errorDescription**	**Description**
0	Null	Success.
1	RequestError	Failure of the request to controller.
2	TypeError	Missing arguments or type errors.
3	NonCollaboError	Failure of the measurement by disabling the Contact Stop function.
4	InvDcsPoschError	Failure of the measurement by disabling the DCS Position / Speed check function.
5	UninitError	The API for initialization has not been executed.

⚠ Caution

top.frpl_utl.recordPayloadIdentData is available on the robot controller software V9.40P/60 or after. If a plugin uses these APIs, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/59 or older.

7.5.15 top.frpl_utl.checkPayloadIdentData

top.frpl_utl.checkPayloadIdentData

Overview		
Check the results measured by the APIs of payload identification.		
Function name		
frpl_utl.recordPayloadIdentData()		
Parameters		
Name	**Type**	**Description**
-	-	-

Return values		
Name	**Type**	**Description**
promise	Promise	Returns the respObj as promiseResult Promise Status is as follows. • Resolve: Success • Reject: Cancel or error
Detail: Return values (respObj)		
Name	**Type**	**Description**
errorDescription	String	Message explaining reject. Returns “null” when resolved. See table [Details: errorDescription list] in this section for details on message.
statusInPos1	String	Result measured at position #1 Completed:0, Failed to measure:1, Unmeasured:2, Failed to check:3
statusInPos2	String	Result measured at position #2 Completed:0, Failed to measure:1, Unmeasured:2, Failed to check:3
statusInPos3	String	Result measured at position #3 Completed:0, Failed to measure:1, Unmeasured:2, Failed to check:3
statusInPos4	String	Result measured at position #4 Completed:0, Failed to measure:1, Unmeasured:2, Failed to check:3
Note		
• Please execute this API when you confirm the result of the API “frpl_utl.recordPayloadIdentData". • It is not necessary to execute this API if you do not need to check the measured results.		

Details: errorDescription list	
errorDescription	**Description**
null	Success.
RequestError	Failure of the request to controller.

⚠ Caution

top.frpl_utl.checkPayloadIdentData is available on the robot controller software V9.40P/60 or after. If a plugin uses these APIs, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/59 or older.

7.5.16 top.frpl_utl.getPayloadIdentResult

top.frpl_utl.getPayloadIdentResult

Overview		
Identify the payload based on the results measured by the APIs of the payload identification.		
Function name		
frpl_utl.getPayloadIdentResult(massKnown, loadMass)		
Parameters		
Name	**Type**	**Description**
massKnown	String	Whether mass is known, or not. Unknown:0, Known:1
loadMass	String	Mass[kg]

Return values		
Name	**Type**	**Description**
promise	Promise	Returns the respObj as promiseResult Promise Status is as follows. • resolve: Success • reject: Cancel or error
Details: Return values (respObj)		
Name	**Type**	**Description**
judgement	String	Judgment
status	String	Returns the results status. See table [Details: Status list] described in this section for details.
errorDescription	String	Message explaining reject. Returns “null” when resolved. See table [Details: Status list] described in this section for details on message.
mass	String	Payload mass[kg]
momentJ4	String	Axis moment(J4) [Nm]
momentJ5	String	Axis moment (J5) [Nm]
momentJ6	String	Axis moment (J6) [Nm]
centerX	String	Center of gravity (X) [m]
centerY	String	Center of gravity (Y) [m]
centerZ	String	Center of gravity (Z) [m]
allowMass	String	Allowable value (Payload mass [kg])
allowMomJ4	String	Allowable value (Axis moment (J4) [Nm])
allowMomJ5	String	Allowable value (Axis moment (J5) [Nm])
allowMomJ6	String	Allowable value (Axis moment (J6) [Nm])
Note		
• Please execute this API after executing the API "frpl_utl.recordPayloadIdentData" at four positions. • The second argument is valid only when the first argument is set to 1(known).		

Details: Status list			
judgement	**status**	**errorDescription**	**Description**
0	0	null	Success(OK)
	1	null	Success(Load is close to capacity)
	2	null	Success(Overload)
1	3	RequestError	Failure of the request to controller.
	4	ArgumentError	Missing arguments or type errors.
	5	MeasurementError	There are unmeasured positions.
	6	InternalError	Internal error.

⚠ Caution

top.frpl_utl.getPayloadIdentResult is available on the robot controller software V9.40P/60 or after. If a plugin uses these APIs, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/59 or older.

7.5.17 top.frpl_utl.setPayloadData

top.frpl_utl.setPayloadData

Overview
Set the payload data.

Function name		
frpl_utl.setPayloadData(grpNum, sheduleNum, comment, loadMass, centerX, centerY, centerZ, inertiaX, inertiaY, inertiaZ)		
Parameters		
Name	**Type**	**Description**
grpNum	String	Group number
sheduleNum	String	Payload number
comment	String	Comment
loadMass	String	Payload mass[kg]
centerX	String	Center of gravity (X) [m]
centerY	String	Center of gravity (Y) [m]
centerZ	String	Center of gravity (Z) [m]
inertiaX	String	LoadInertia (X) [kgm^2]
inertiaY	String	LoadInertia (Y) [kgm^2]
inertiaZ	String	LoadInertia (Z) [kgm^2]
Return values		
Name	**Type**	**Description**
promise	Promise	Returns the respObj as PromiseResult PromiseState is as follows. • Resolve: Success • Reject: Cancel or error

Detail: Return values (respObj)		
Name	**Type**	**Description**
status	String	Returns the result status. See table [Details: Status list] described in this section for details.
errorDescription	String	Message explaining reject. Returns “null” when resolved. See table [Details: Status list] described in this section for details on message.
Note		
-		

Details: Status list		
status	**errorDescription**	**Description**
0	null	Success
1	RequestError	Failure of the request to controller.
2	TypeError	Missing arguments or type errors.
3	PayloadSetError	Failure of setting the payload data.

⚠ Caution

top.frpl_utl.setPayloadData is available on the robot controller software V9.40P/60 or after. If a plugin uses these APIs, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/59 or older.

7.5.18 top.frpl_utl.applyDcsParam

top.frpl_utl.applyDcsParam

Overview
Show a pop-up diagram to allow the user to apply DCS parameters. • This API displays a pop-up diagram to let the user enter a DCS password and confirmation for the parameters. See figure 7.5.18 (a).

• See the Dual Check Safety Function OPERATOR'S MANUAL (B-83184EN) for more details.
Function name
frpl_utl.applyDcsParam
Parameters

Name	Type	Description
popupWin	Object	Window object to display the pop-up to apply DCS parameters. If not specified (if null specified), it displays the pop-up to the top.
cbArgs	Object	Parameter to pass to the return value of respObj (optional)

Details: Return values

Name	Type	Description
Promise	Promise	Returns respObj as a PromiseResult. See table [Details: Return values (respObj)] in this section for more details. PromiseState is as follows. • resolve: Success • reject: Failed or error occurs. See the following figure for details.

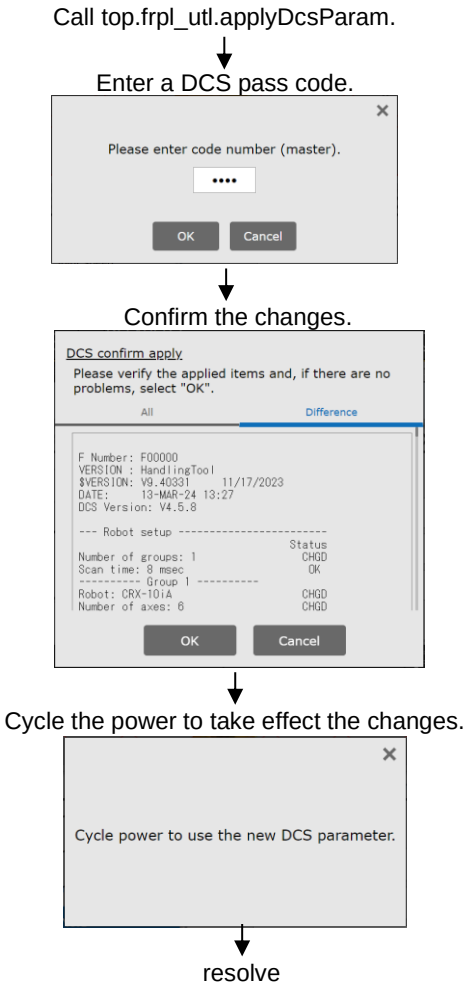
Details: Return values (respObj)

Parameters

Name	Type	Description
cbArgs	Object	Object specified when this API is called.

Note

-



In case of reject you will get messages like the followings.

- There are no items that need to be applied.
- There is a mistake in the settings.
- The processing time factor has been exceeded. Please change the settings of the areas or models so that the processing time factor is within the range.
- The application process is already in progress. Please complete the application process in progress.
- Incorrect code number.

Fig 7.5.18 (a) Pop-up diagram to apply DCS parameters

> ⚠ **Caution**
> top.frpl_utl.applyDcsParam is available on the robot controller software V9.40P/60 or after. If a plugin uses these APIs, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/59 or older.

7.5.19 top.frpl_utl.getMultiKarelVars

top.frpl_utl.getMultiKarelVars

Overview		
Get multiple KAREL variables.		
Function name		
frpl_utl.getMultiKarelVars		
Parameters		
Name	**Type**	**Description**
jsonData	Object	Json formatted data containing the following: • KAREL program name • KAREL variable names
Details: jsonData		
Name	**Type**	**Description**
data	Object / Array	Object or array containing KAREL program name and KAREL variable names
Details: data[x]		
Name	**Type**	**Description**
prog	String	Name of the KAREL program where the variable is defined
vars	Array	Array of KAREL variable names to get. Refer to table [Examples of KAREL Array and Structure] in the following section for specifying KAREL structures and arrays.
Examples: jsonData		

data = Object
{
“data”: {
“prog”: ”KAREL program name 1”,
“vars”: [“pos_x”, “pos_y”, “pos_z”]
}
}
		data = Array
		{
		“data”: [{
		“prog”: ”KAREL program name 1”,
		“vars”: [“pos_x”, “pos_y”, “pos_z”]
		},{
		“prog”: ” KAREL program name 2”,
		“vars”: [“pallet”]
		}]
		}

Return value		
Name	**Type**	**Description**
promise	Promise	Returns respObj as PromiseResult. See table [Details: Return value (respObj)] for details. PromiseState is the following. • resolve: Successful getting KAREL variable values. • reject: Error occurred during getting values.
Details: Return value (respObj)		
Name	**Type**	**Description**
status	String	Returns the results status. See table [Details: Status list] in this section for details.
errorDescription	String	Message explaining reject.

		Returns “null” when resolved. See table [Details: Status list] in this section for details on message.
data	Array	Array of objects with results. Returns “null” when rejected.
Details: respObj.data[x]		
Name	**Type**	**Description**
prog	String	Specified KAREL program name.
vars	Object	Pair of retrieved KAREL variable names and values.
Details: respObj.data[x].vars		
Name	**Type**	**Description**
KAREL variable	-(any)	KAREL variable values
Example: data		

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

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

Details: Status list		
status	**errorDescription**	**Description**
0	Null	Success
1	Invalid JSON	Invalid JSON
2	Invalid post in data	Invalid data key
3	Invalid post in program	Invalid prog key
4	Invalid post in variable	Invalid vars key
5	Unknown variable name	Non-existent KAREL variable
6	Unknown program name	Non-existent KAREL program
7	Bad variable or array index	Invalid variable or array index
8	Invalid value	Invalid variable value
9	Unknown Error	Unknown error

⚠ Caution
top.frpl_utl.getMultiKarelVars and top.frpl_utl.setMultiKarelVars are available on the robot controller software V9.40P/68 or after. If a plugin uses these APIs, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/67 or older.

7.5.20 top.frpl_utl.setMultiKarelVars

top.frpl_utl.setMultiKarelVars

Overview
Set multiple KAREL variables.

Function name		
frpl_utl.setMultiKarelVars		
Parameters		
Name	**Type**	**Description**
jsonData	Object	Json formatted data containing the following: • KAREL program name • KAREL variable names
Details: jsonData		
Name	**Type**	**Description**
data	Object / Array	Object or array containing KAREL program name and KAREL variable names
Details: data		
Name	**Type**	**Description**
prog	String	Name of the KAREL program where the variable is defined
vars	Object	Object containing KAREL variable names and values. Refer to table [Examples of KAREL Array and Structure] for specifying KAREL structures and arrays.
Details: vars		
Name	**Type**	**Description**
KAREL variable	-(any)	KAREL variable values to be set.
Example: data		

data = Object
{
“data”: {
“prog”: ”KAREL program name 1”,
“vars”: {“pos_x”: 111,
“pos_y”: 222,
“pos_z”: 333 }
}
}
		data = Array
		{
		“data”: [{
		“prog”: ”KAREL program name 1”,
		“vars”: {“pos_x”: 111,
		“pos_y”: 222,
		“pos_z”: 333 }
		},{
		“prog”: ”KAREL program name 2”
		“vars”: { “pallet”: [“seq1”, seq2, seq3] }
		}]
		}

Return value		
Name	**Type**	**Description**
promise	Promise	Returns respObj as PromiseResult. See table [Details: Return value (respObj)] for details. PromiseState is the following. • resolve: Successful setting KAREL variable values. • reject: Error occurred during setting values.
Details: Return value (respObj)		
Name	**Type**	**Description**
status	String	Returns the results status. See table [Details: Status list] in the previous section for details.
errorDescription	String	Message explaining reject. Returns “null” when resolved. See table [Details: Status list] in the previous section for details on message.

[Examples of KAREL Array and Structure]

KAREL Array

```
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
```

jsonData object example to set KAREL array

```
{
 “data”: {
  “prog”: ”KAREL program name 1”,
  “vars”: {“num_ary”: [[23,6,7],[12,5,78]] }
 }
}
```

jsonData object example to get KAREL array

```
{
 “data”: {
  “prog”: ”KAREL program name 1”,
  “vars”: { “num_ary[1,1]” }
 }
}
```

KAREL Structure

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

jsonData object example to set KAREL structure

```
{
 “data”: {
  “prog”: ”KAREL program name 1”,
  “vars”: {“rectangle1” : { “pt1”: {“x”: 0, “y”: 0},“pt2”: {“x”: 0, “y”: 0} }
 }
}
```

jsonData object example to get KAREL structure

```
{
 “data”: {
  “prog”: ”KAREL program name 1”,
  “vars”: [“rectangle1.pt1”]
 }
}
```

⚠ Caution
top.frpl_utl.getMultiKarelVars and top.frpl_utl.setMultiKarelVars are available on the robot controller software V9.40P/68 or after. If a plugin uses these APIs, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/67 or older.

7.5.21 top.frpl_utl.getRegister

top.frpl_utl.getRegister

Overview		
Get the specified register value.		
Function name		
frpl_utl.getRegister		
Parameters		
Name	**Type**	**Description**
type	String / Number	Specify register type to get 1: Numeric register 2: Position register 3: String register
index	String / Number	Specify register index number to get
Return value		
Name	**Type**	**Description**
promise	Promise	Returns respObj as PromiseResult. See table [Details: Return value (respObj)] described in this section for details. PromiseState is the following. • resolve: Successful getting register value. • reject: Error occurred during getting value.
Details: Return value (respObj)		
Name	**Type**	**Description**
status	String	Returns the results status. See table [Details: Status list] in this section for details.
errorDescription	String	Message explaining reject. Returns “null” when resolved. See table [Details: Status list] in this section for details on message.
regObj	Object	Returns objects with results. Returns “null” when rejected.

Details: Return value (respObj.regObj)		
Name	**Type**	**Description**
type	Number	Register type
index	Number	Register index number
comment	String	Comments of the register
value	String	Numeric value of the register
	String	String value of the string register
	Object	Position data of the position register • Returns position data for all motion group. • Returns Uninitialized when the position data is not recorded. • See table [Details: Position register data (value.gX)] in this section for details on how it returns the position data.
Details: Position register data (value)		
Name	**Type**	**Description**
totalgroup	Number	Total number of motion groups
Details: Position register data (value.gX) (where X indicates the motion group number)		
Name	**Type**	**Description**
group	Number	Motion group number of the position register
rep	Number	Position data representation 1: Cartesian 2: Joint
axis	Number	Number of robot axis (returns 0 if it is Cartesian representation)
extaxis	Number	Number of extended axis
config	String	• Configuration data for Cartesian position (Example) “N U T, 0, 0, 0” • Returns no data for Joint position
x,y,z,w,p,r	String	Obtained Cartesian position data (X, Y, Z, W, P, R)
j1 to j6 ext1 to ext3	String	• Joint position for robot axis (j1 to j6) and extended axis (ext1 to ext3) • ext1 to ext3 does not exist if there is no extended axis loaded.

Example: Position register data

```
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",
   },
}
```

Details: Status list		
status	**errorDescription**	**Description**
0	Null	Success
1	ValueError	Invalid parameters specified (Non identifiable value found)
2	WriteError	Failed to write data to the robot controller
3	TypeError: Argument (Parameter number (Parameter name)) is not of type (Correct data type).	Type error of parameters
4	TypeError: Argument (Parameter number (Parameter name)) is not 1-3.	Non-exist register type specified
5	IndexError: Argument (Parameter number (Parameter name)) does not exist.	Non-exist register index number specified (only when setting register)
6	ValueError: Argument (Parameter number (Parameter name)) cannot use (' &).	Found illegal characters such as (') or (&) in the register comment.
7	ValueError: Argument (Parameter number (Parameter name)) is not a number.	Found illegal value set in numeric/position register.
8	ValueError: Argument (Parameter number (Parameter name)) is incorrect configuration.	Found illegal configuration data specified.
9	ValueError: Argument (Parameter number (Parameter name)) is incorrect rep.	Found illegal position data representation specified.
10	ValueError: Argument (Parameter number (Parameter name)) is incorrect number of axes.	Found illegal number of axis specified.

⚠ Caution

top.frpl_utl.getRegister and top.frpl_utl.setRegister are available on the robot controller software V9.40P/61 or after. If a plugin uses these APIs, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/60 or older.

7.5.22 top.frpl_utl.setRegister

top.frpl_utl.setRegister

Overview		
Set the specified register value.		
Function name		
frpl_utl.setRegister		
Parameters		
Name	**Type**	**Description**
regObj	String / Number	Object containing register information to set. See table [Details: Parameters (regObj)] described in this section for details.

Return value		
Name	**Type**	**Description**
promise	Promise	Returns respObj as PromiseResult. See table [Details: Return value (respObj)] described in this section for details. PromiseState is the following. • resolve: Successful setting register value. • reject: Error occurred during setting value.
Details: Parameters (regObj)		
Name	**Type**	**Description**
type	Number	Register type
index	Number	Register index number
comment	String	Comments of the register to set
value	String	Numeric value to set the register
	String	String value to set the register
	Object	Position data to set the register • You can configure position data for number of motion groups you want to set. • See table [Details: Position register data (value.gX)] in this section for details on how to it returns the position data.
Details: Return value (respObj)		
Name	**Type**	**Description**
status	String	Returns the results status. See table [Details: Status list] in the previous section for details.
errorDescription	String	Message explaining reject. Returns “null” when resolved. See table [Details: Status list] in the previous section for details on message.
Details: Position register data (value.gX) (where X indicates the motion group number)		
Name	**Type**	**Description**
rep	Number	Position data representation to set 1: Cartesian 2: Joint
config	String	• Specify configuration for Cartesian position (Example) “N U T, 0, 0, 0” • No need to specify this parameter for Joint position
x,y,z,w,p,r	String/Number	Cartesian position data (X, Y, Z, W, P, R) to set
j1-j6 ext1-ext3	String/Number	Joint position data including extended axis to set • No need ext1 to ext3 if there is no extended axis loaded. • No unit required. • Setting position data fails if you specify position data for number of axis different from actual configuration loaded in the robot controller. Make sure to get the number of robot axis (regObj.value.gX.axes) and extended axis (regObj.value.gX.extaxes) by calling top.frpl_utl.getRegister() in prior to creating position register data.
Example: Position register data		

```
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",
  },
}
```

⚠ Caution

top.frpl_utl.getRegister and top.frpl_utl.setRegister are available on the robot controller software V9.40P/61 or after. If a plugin uses these APIs, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/60 or older.

7.5.23 top.frpl.getIO

top.frpl.getIO

Overview		
Get the value of a specified IO type from the robot controller.		
Function name		
frpl.getIO(ioType, point)		
Parameters		
Name	**Type**	**Description**
ioType	IOType	Type of robot controller I/O to get
point	Number	The I/O point number to retrieve based on the I/O type
Return values		
Name	**Type**	**Description**
promise	IOPoint	Returns a promise of type IOPoint with the I/O data
Example: API call		
Get the value of Digital Output I/O # 5 • let a1 = await top.frpl.getIO(IOType.DO, 5); More information about “IOType.DO” can be found in the section 7.6.2 below		
Example: Return data		

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

⚠ Caution

top.frpl.getIO is available on the robot controller software V9.40P/65 or after. If a plugin uses these APIs, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/64 or older.

7.5.24 top.frpl.setIO

top.frpl.setIO

Overview		
Set the value of a specified IO type from the robot controller.		
Function name		
frpl.setIO(ioType, point, value)		
Parameters		
Name	**Type**	**Description**
ioType	IOType	Type of robot controller I/O to set
point	Number	The I/O point number to set based on the I/O type
Value	Number	The value to set for the specified robot controller I/O point
Return values		
Name	**Type**	**Description**
promise	Status	Returns a promise of type Status with the status of the request

Example: API call

Set the value of I/O Digital Out with index 5 to “ON” state

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

Set the I/O Analog Out with index 7 to “33” state

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

More information about “IOType.DO” and “IOType.AO” can be found in the section 7.6.2 below

Example: Return data

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

> ⚠ **Caution**
> top.frpl.setIO is available on the robot controller software V9.40P/65 or after. If a plugin uses these APIs, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/64 or older.

7.5.25 top.frpl.startIOMonitor

top.frpl.startIOMonitor

Overview
Start an I/O monitor for the specified I/O point. When controller detects change with I/O value, it calls the callback function to notify about the change in value.
Function name
frpl.startIOMonitor(ioType, point, clientData, callback)

Parameters		
Name	**Type**	**Description**
ioType	IOType	The I/O type to observe
point	Number	The point number of the specified I/O type to observe for changes
clientData	Any	Data that is to be passed back to the client when the observed I/O point changes
callback	IIOHandler	A callback handler the subject will call when the I/O point changes. The callback function will receive a parameter of type OnIOArgs
Return values		
Name	**Type**	**Description**
promise	Status	Returns a promise of type Status with the status of the request
Parameters (callback)		
Name	**Type**	**Description**
args	OnIOArgs	The callback function will receive a parameter of type OnIOArgs.

Example: API call

```
// * See the last row of this table about how the callback functions work
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);
```

More information about “IOType.DO” can be found in the section 7.6.2 below

Note: The same callback function must be used for both starting and stopping the monitors

Example: Return Data

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

Example: Callback Data

After IO monitor is started, whenever the I/O Digital Out of index 3 changes to “ON” state the following output will be observed for “a1”.

foo = “Hello there”

And whenever the I/O Digital Out of index 3 changes to “OFF” state the following output will be observed for “a1”.
foo = “Hello there”

After IO monitor is started, whenever the I/O Digital Out of index 5 changes to “ON” state the following output will be observed for “a2”.

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

And whenever the I/O Digital Out of index 5 changes to “OFF” state the following output will be observed for “a2”.

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

⚠ Caution
top.frpl.startIOMonitor is available on the robot controller software V9.40P/65 or after. If a plugin uses these APIs, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/64 or older.

7.5.26 top.frpl.stopIOMonitor

top.frpl.stopIOMonitor

Overview		
Stop the I/O monitor for the specified I/O point that was previously started. The monitor is stopped by removing the previously registered client callback.		
Function name		
frpl.stopIOMonitor(ioType, point, callback)		
Parameters		
Name	**Type**	**Description**
ioType	IOType	The I/O type to observe
point	Number	The point number of the specified I/O type to observe for changes
callback	IIOHandler	A callback handler the subject will call when the I/O point changes. The callback function will receive a parameter of type OnIOArgs
Return values		
Name	**Type**	**Description**
promise	Status	Returns a promise of type Status with the status of the request
Example: API call		

let a1 = await top.frpl.stopIOMonitor(IOType.DO, 3, callback);
let a2 = await top.frpl.stopIOMonitor(IOType.DO, 5, callback);
 	
More information about “IOType.DO” can be found in the section 7.6.2 below
Note: The same callback function must be used for both starting and stopping the monitors		

Example: Return Data		

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

```
    errorDescription: "",
    state: true
}
```

⚠ Caution
top.frpl.stopIOMonitor is available on the robot controller software V9.40P/65 or after. If a plugin uses these APIs, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/64 or older.

7.5.27 top.frpl.getJointPos

top.frpl.getJointPos

Overview		
Get the specified robot group's joint angles.		
Function name		
frpl.getJointPos(groupNum)		
Parameters		
Name	**Type**	**Description**
groupNum	Number	Motion group number to get the joint angles
Return values		
Name	**Type**	**Description**
promise	RobotJointAngles	Returns a promise of type RobotJointAngles with joint angle data
Example: API call		
let a1 = await top.frpl.getJointPos(2); // position of group-2		
Example: Return Data		

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

⚠ Caution
top.frpl.getJointPos is available on the robot controller software V9.40P/65 or after. If a plugin uses these APIs, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/64 or older.

7.5.28 top.frpl.startJointPosMonitor

top.frpl.startJointPosMonitor

Overview
Start monitoring for robot controller joint angle changes. When controller detects change with joint positions, it calls the callback function to notify about the change in value.
Function name
frpl.startJointPosMonitor(clientData, callback)

Parameters		
Name	**Type**	**Description**
clientData	Any	Data the client wants passed back to when the callback function is invoked
callback	IJointPosHandler	The function to call when the robot's joint angles change
Return values		
Name	**Type**	**Description**
promise	Status	Returns a promise of type Status with the status of the request
Parameters (callback)		
Name	**Type**	**Description**
args	OnJointPosChange	The callback function will receive a parameter of type OnJointPosChange.

Example: API call

```
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);
```

Note: The same callback function must be used for both starting and stopping the monitors

Example: Return Data

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

Example: Callback Data

```
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";
```

⚠ Caution

top.frpl.startJointPosMonitor is available on the robot controller software V9.40P/65 or after. If a plugin uses these APIs, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/64 or older.

7.5.29 top.frpl.stopJointPosMonitor

top.frpl.stopJointPosMonitor

Overview
Stop the joint angle monitor that was previously started. The monitor is stopped by removing the previously registered client callback.
Function name
frpl.stopJointPosMonitor(callback)

Parameters (callback)		
Name	**Type**	**Description**
callback	IJointPosHandler	The callback handler that was previously registered for monitoring
Return values		
Name	**Type**	**Description**
promise	Status	Returns a promise of type Status with the status of the request
Example: API call		
let a1 = await top.frpl.stopJointPosMonitor(callback); **Note:** The same callback function must be used for both starting and stopping the monitors		
Example: Return Data		

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

> ⚠ **Caution**
> top.frpl.stopJointPosMonitor is available on the robot controller software V9.40P/65 or after. If a plugin uses these APIs, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/64 or older.

7.5.30 top.frpl.getWorldPos

top.frpl.getWorldPos

Overview		
Get the specified robot group's world position.		
Function name		
frpl.getWorldPos(groupNum)		
Parameters		
Name	**Type**	**Description**
groupNum	Number	Robot group number whose world position to return
Return values		
Name	**Type**	**Description**
promise	RobotXyzWprPos	Returns a promise of type RobotXyzWprPos with the world position data
Example: API call		
let a1 = await top.frpl.getWorldPos(1); // position of group-1		
Example: Return Data		

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';		

⚠ Caution
top.frpl.getWorldPos is available on the robot controller software V9.40P/65 or after. If a plugin uses these APIs, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/64 or older.

7.5.31 top.frpl.getWorldPosWithUF

top.frpl.getWorldPos

Overview		
Get the specified robot group's cartesian position with respect to the active user frame.		
Function name		
frpl.getWorldPosWithUF(groupNum)		
Parameters		
Name	**Type**	**Description**
groupNum	Number	Robot group number whose world position to return
Return values		
Name	**Type**	**Description**
promise	RobotXyzWprPos	Returns a promise of type RobotXyzWprPos with the world position data
Example: API call		
let a1 = await top.frpl.getWorldPosWithUF(1); // position of group-1		
Example: Return Data		

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		

⚠ Caution
top.frpl.getWorldPosWithUF is available on the robot controller software V9.40P/65 or after. If a plugin uses these APIs, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/64 or older.

7.5.32 top.frpl.startWorldPosMonitor

top.frpl.startWorldPosMonitor

Overview		
Start monitoring for robot controller tool center point (TCP) relative to the robot's world frame. When controller detects change with world position data, it calls the callback function to notify about the change in value.		
Function name		
frpl.startWorldPosMonitor(clientData, callback)		
Parameters		
Name	**Type**	**Description**
clientData	Any	Data the client wants passed back to when the callback function is invoked

callback	IWorldPosHandler	The function to call when the robot's world position changes
Return values		
Name	**Type**	**Description**
promise	Status	Returns a promise of type Status with the status of the request
Parameters (callback)		
Name	**Type**	**Description**
args	OnWorldPosChange	The callback function will receive a parameter of type OnWorldPosChange.
Example: API call		

```
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);
```

Note: The same callback function must be used for both starting and stopping the monitors

Example: Return Data

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

Example: Callback Data

```
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"
```

⚠ Caution

top.frpl.startWorldPosMonitor is available on the robot controller software V9.40P/65 or after. If a plugin uses these APIs, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/64 or older.

7.5.33 top.frpl.stopWorldPosMonitor

top.frpl.stopWorldPosMonitor

Overview		
Stop monitoring for robot controller tool center point (TCP) relative to the robot's world frame. The monitor is stopped by removing the previously registered client callback.		
Function name		
frpl.stopWorldPosMonitor(callback)		
Parameters		
Name	**Type**	**Description**
callback	IWorldPosHandler	Remove a previously registered client callback, observer, from the collection of world position change observers

Return values		
Name	**Type**	**Description**
promise	Status	Returns a promise of type Status with the status of the request
Example: API call		
let a1 = await top.frpl.stopWorldPosMonitor(callback); **Note:** The same callback function must be used for both starting and stopping the monitors		
Example: Return Data		
a1 = {state: true, errorCode: '', errorDescription: ''}		

⚠ Caution
top.frpl.stopWorldPosMonitor is available on the robot controller software V9.40P/65 or after. If a plugin uses these APIs, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/64 or older.

7.5.34 top.frpl.getVariable

top.frpl.getVariable

Overview		
Get a KAREL variable.		
Function name		
frpl.getVariable(programName, varName)		
Parameters		
Name	**Type**	**Description**
programName	String	The name of the KAREL program in which the variable is a part of
varName	String	Name of the KAREL variable data to get
Return values		
Name	**Type**	**Description**
promise	RobotVarInfo	Returns a promise of type RobotVarInfo containing the variable data
Example: API call		
let a1 = await top.frpl.getVariable(“prog_1”, “my_var”);		
Example: Return Data		
a1 = {variableName: 'my_var', programName: 'prog_1', value: 'hello', tag: 'prog_1+my_var'}		

⚠ Caution
top.frpl.getVariable is available on the robot controller software V9.40P/65 or after. If a plugin uses these APIs, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/64 or older.

7.5.35 top.frpl.setVariable

top.frpl.setVariable

Overview
Set the value of a KAREL variable on the robot controller.

Function name		
frpl.setVariable(programName, varName, value)		
Parameters		
Name	**Type**	**Description**
programName	String	The name of the KAREL program in which the variable is a part of
varName	String	Name of the KAREL variable data to set
value	Any	The value to be set
Return values		
Name	**Type**	**Description**
promise	Status	Returns a promise of type Status with the status of the request
Example: API call		
let a1 = await top.frpl.setVariable(“prog_1”, “my_var”, “test_string”);		
Example: Return Data		
a1 = {state: true, errorCode: ' ', errorDescription: ' '}		

> ⚠ **Caution**
> top.frpl.setVariable is available on the robot controller software V9.40P/65 or after. If a plugin uses these APIs, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/64 or older.

7.5.36 top.frpl.startVarMonitor

top.frpl.startVarMonitor

Overview		
Start a KAREL variable monitor. When controller detects change with variable value, it calls the callback function to notify about the change in value.		
Function name		
frpl.startVarMonitor(programName, varName, clientData, callback)		
Parameters		
Name	**Type**	**Description**
programName	String	The name of the KAREL program in which the variable is present
varName	String	The name of the variable to be observed for changes
clientData	Any	Data that is to be passed back to the client when the observed variable changes
callback	IVarHandler<RobotVarInfo>	A callback handler the subject will call when the variable data changes
Return values		
Name	**Type**	**Description**
promise	Status	Returns a promise of type Status with the status of the request
Parameters (callback)		
Name	**Type**	**Description**
args	OnVarArgs<RobotVarInfo>	The callback function will receive a parameter of type OnVarArgs<RobotVarInfo>
Example: API call		

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);
```

Note: The same callback function must be used for both starting and stopping the monitors

Example: Return Data

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

Example: Callback Data

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

⚠ Caution

top.frpl.startVarMonitor is available on the robot controller software V9.40P/65 or after. If a plugin uses these APIs, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/64 or older.

7.5.37 top.frpl.stopVarMonitor

top.frpl.stopVarMonitor

Overview

Stop the KAREL variable monitor that was previously started.
The monitor is stopped by removing the previously registered client callback.

Function name

frpl.stopVarMonitor(programName, varName, callback)

Parameters

Name	Type	Description
programName	String	The name of the KAREL program in which the variable is present
varName	String	The name of the variable to stop monitoring
callback	IVarHandler<RobotVarInfo>	The callback handler that was previously registered for monitoring

Return values

Name	Type	Description
promise	Status	Returns a promise of type Status with the status of the request

Example: API call

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

Note: The same callback function must be used for both starting and stopping the monitors

Example: Return Data

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

> ⚠ **Caution**
> top.frpl.stopVarMonitor is available on the robot controller software V9.40P/65 or after. If a plugin uses these APIs, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/64 or older.

7.5.38 top.frpl.runProgram

top.frpl.runProgram

Overview		
Run a TP program.		
Function name		
frpl.runProgram(programName, startAtLine)		
Parameters		
Name	**Type**	**Description**
programName	String	Name of the TP program to Run
startAtLine	Number	Line number to run the program from. Defaults to 1.
Return values		
Name	**Type**	**Description**
promise	Status	Returns a promise of type Status with the status of the request.
Example: API call		
let a1 = await top.frpl.runProgram(“myprog”); let failCase = await top.frpl.runProgram(“myprog”);		
Example: Return Data		
a1 = {state: true, errorCode: '', errorDescription: ''} failCase = {state: false, errorCode: '0x30007', errorDescription: 'PROG-007 Program is already running'}		

> ⚠ **Caution**
> top.frpl.runProgram is available on the robot controller software V9.40P/65 or after. If a plugin uses these APIs, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/64 or older.

7.5.39 top.frpl.pauseProgram

top.frpl.pauseProgram

Overview		
Pause a TP program.		
Function name		
frpl.pauseProgram(programName)		
Parameters		
Name	**Type**	**Description**
programName	String	Name of the TP program to pause
Return values		
Name	**Type**	**Description**
promise	Status	Returns a promise of type Status with the status of the request
Example: API call		
let a1 = await top.frpl.pauseProgram("myprog");		
Example: Return Data		

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

⚠ Caution
top.frpl.pauseProgram is available on the robot controller software V9.40P/65 or after. If a plugin uses these APIs, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/64 or older.

7.5.40 top.frpl.resumeProgram

top.frpl.resumeProgram

Overview		
Resume a TP program.		
Function name		
frpl.resumeProgram(programName)		
Parameters		
Name	**Type**	**Description**
programName	String	Name of the TP program to resume execution
Return values		
Name	**Type**	**Description**
promise	Status	Returns a promise of type Status with the status of the request
Example: API call		
let a1 = await top.frpl.resumeProgram("myprog");		
Example: Return Data		
a1 = {state: true, errorCode: '', errorDescription: ''}		

⚠ Caution
top.frpl.resumeProgram is available on the robot controller software V9.40P/65 or after. If a plugin uses these APIs, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/64 or older.

7.5.41 top.frpl.abortProgram

top.frpl.abortProgram

Overview		
Abort a TP program.		
Function name		
frpl.abortProgram(programName)		
Parameters		
Name	**Type**	**Description**
programName	String	Name of the TP program to abort
Return values		
Name	**Type**	**Description**
promise	Status	Returns a promise of type Status with the status of the request
Example: API call		
let a1 = await top.frpl.abortProgram("myprog");		
Example: Return Data		
a1 = {state: true, errorCode: ' ', errorDescription: ' '}		

⚠ Caution
top.frpl.abortProgram is available on the robot controller software V9.40P/65 or after. If a plugin uses these APIs, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/64 or older.

7.5.42 top.frpl.abortAllPrograms

top.frpl.abortAllPrograms

Overview		
Abort all TP programs.		
Function name		
frpl.abortAllPrograms()		
Parameters		
Name	**Type**	**Description**
N/A	N/A	N/A
Return values		
Name	**Type**	**Description**
promise	Status	Returns a promise of type Status with the status of the request
Example: API call		
let a1 = await top.frpl.abortAllPrograms();		
Example: Return Data		
a1 = {state: true, errorCode: ' ', errorDescription: ' '}		

⚠ Caution
top.frpl.abortAllPrograms is available on the robot controller software V9.40P/65 or after. If a plugin uses these APIs, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/64 or older.

7.5.43 top.frpl.readProgramToApi

top.frpl.readProgramToApi

Overview		
Read a TP program to API.		
Function name		
frpl.readProgramToApi(programName)		
Parameters		
Name	**Type**	**Description**
programName	String	Name of the TP program to read
Return values		
Name	**Type**	**Description**

promise	{
	status:Status,
	data:String
	}
		Returns a promise of type JSON object with keys 'status' and 'data'. The key 'status' will have the pass/fail status information along with an error description, if any. Contents of JSON key 'data' will have a string containing all the text from the requested program.

Example: API call		
let a1 = await top.frpl.readProgramToAPI(“myprog”);		

Example: Return Data

```
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: ''}
}
```

⚠ Caution

top.frpl.readProgramToAPI is available on the robot controller software V9.40P/65 or after. If a plugin uses these APIs, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/64 or older.

7.5.44 top.frpl.listTpPrograms

top.frpl.listTpPrograms

Overview
List names of all TP programs in the robot controller.

Function name		
frpl.listTpPrograms()		
Parameters		
Name	**Type**	**Description**
N/A	N/A	N/A
Return values		
Name	**Type**	**Description**

promise	{
	status:Status,
	data:String[ ]
	}
		Returns a promise of type JSON object with keys 'status' and 'data'. The key 'status' will have the pass/fail status information along with an error description, if any. Contents of JSON key 'data' will have a list/arrary of all the TP program names found in the controller

Example: API call		
let a1 = await top.frpl.listTpPrograms();		
Example: Return Data		
a1 = { data: ['DEFAULT', 'MOTN_BKWD', 'MOTN_FWD', 'SRI', 'TEST', 'TEST_1', 'TRANSPORT'], status: {state: true, errorCode: '', errorDescription: ''} }		

⚠ Caution
top.frpl.listTpPrograms is available on the robot controller software V9.40P/65 or after. If a plugin uses these APIs, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/64 or older.

7.5.45 top.frpl.deleteProgram

top.frpl.deleteProgram

Overview		
Delete a TP program from the controller.		
Function name		
frpl.deleteProgram(programName)		
Parameters		
Name	**Type**	**Description**
programName	String	Name of the TP program to be deleted
Return values		
Name	**Type**	**Description**
promise	Status	Returns a promise of type Status with the status of the request
Example: API call		
let a1 = await top.frpl.deleteProgram();		
Example: Return Data		
a1 = {state: true, errorCode: '', errorDescription: ''};		

⚠ Caution
top.frpl.deleteProgram is available on the robot controller software V9.40P/65 or after. If a plugin uses these APIs, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/64 or older.

7.5.46 top.frpl.uploadProgram

top.frpl.uploadProgram

Overview		
Upload a TP program to the controller. Set the parameter 'programData' to empty string(' ') to pass the optional parameters.		
Function name		
frpl.uploadProgram(programName, programData, programCommands, positionData, clientHeaders)		
Parameters		
Name	**Type**	**Description**
programName	String	Name of the TP program to upload
programData	String	Data of the program to be uploaded
programCommands	String	(Optional parameter) Commands/code lines of the program to be passed.
positionData	String	(Optional parameter) Position data from the program to be passed.
clientHeaders	JSON object of type { key:String, value:String }	(Optional parameter) Header data to be passed as an object { key:String, value:String } pairs.
Return values		
Name	**Type**	**Description**
promise	Status	Returns a promise of type Status with the status of the request

Example: API call

To demonstrate the API, we are considering the sample LS program below.

```
/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
```

The two examples below show a couple of variations to upload the LS program above to the controller using the Upload Program API.

Example - 1: Upload program by passing all the data

This example shows the basic method to upload a TP program. For this method, two variables are used. First variable (prog_name) has the program name as string. The second variable (prog_data) has the entire contents of the LS file stored as a string.

```
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);
```

Example - 2: Upload program with position data and TP code (program commands)

This example shows a method to upload a TP program where user can choose the contents of the program. For this method, only TP code and position data are used. All the data is stored as strings in the appropriate variables.

Note: User must set the second parameter of the API call “programData” as an empty string to use the optional parameters of the API.

```
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);
```

Example: Return Data

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

⚠ Caution

top.frpl.uploadProgram is available on the robot controller software V9.40P/65 or after. If a plugin uses these APIs, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/64 or older.

7.5.47 top.frpl.getCurrentProgramName

top.frpl.getCurrentProgramName

Overview		
Method to get the name of current/active program.		
Function name		
frpl.getCurrentProgramName()		
Parameters		
Name	**Type**	**Description**
N/A	N/A	N/A
Return values		
Name	**Type**	**Description**
promise	String	Returns a promise of type string
Example: API call		
let a1 = await top.frpl.getCurrentProgramName();		
Example: Return Data		
a1 = "myProgram1";		

> ⚠ **Caution**
> top.frpl.getCurrentProgramName is available on the robot controller software V9.40P/65 or after. If a plugin uses these APIs, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/64 or older.

7.5.48 top.frpl.getJointLimits

top.frpl.getJointLimits

Overview		
Method to get the robot joint limits.		
Function name		
frpl.getJointLimits()		
Parameters		
Name	**Type**	**Description**
N/A	N/A	N/A
Return values		
Name	**Type**	**Description**
promise	RobotJointLimits	Returns an promise of type RobotJointLimits with joint limits data
Example: API call		
let a1 = await top.frpl.getJointLimits();		
Example: Return Data		

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]
}		

> **⚠ Caution**
> top.frpl.getJointLimits is available on the robot controller software V9.40P/65 or after. If a plugin uses these APIs, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/64 or older.

7.5.49 top.frpl.getActiveUserFrameNumber

top.frpl.getActiveUserFrameNumber

Overview		
Method to get the active user frame (UF) number of the specified group.		
Function name		
frpl.getActiveUserFrameNumber(groupNumber)		
Parameters		
Name	**Type**	**Description**
groupNumber	Number	Group number to get the UF number from
Return values		
Name	**Type**	**Description**
promise	Number	Returns a promise of type number
Example: API call		
let a1 = await top.frpl.getActiveUserFrameNumber(1);		
Example: Return Data		
a1 = 2		

> **⚠ Caution**
> top.frpl.getActiveUserFrameNumber is available on the robot controller software V9.40P/65 or after. If a plugin uses these APIs, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/64 or older.

7.5.50 top.frpl.getActiveUserToolNumber

top.frpl.getActiveUserToolNumber

Overview		
Method to get the active user tool (UT) number of the specified group.		
Function name		
frpl.getActiveUserToolNumber(groupNumber)		
Parameters		
Name	**Type**	**Description**
groupNumber	Number	Group number to get the UT number from
Return values		
Name	**Type**	**Description**
promise	Number	Returns a promise of a type number
Example: API call		
let a1 = top.frpl.getActiveUserToolNumber(1);		
Example: Return Data		
a1 = 2		

⚠ Caution
top.frpl.getActiveUserToolNumber is available on the robot controller software V9.40P/65 or after. If a plugin uses these APIs, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/64 or older.

7.5.51 top.frpl.startActiveProgramMonitor

top.frpl.startActiveProgramMonitor

Overview		
Method to start monitoring active program data.		
Function name		
frpl.startActiveProgramMonitor(clientData, callback)		
Parameters		
Name	**Type**	**Description**
clientData	Any	Data that is to be passed back to the client when the state of active program is changed
callback	IActiveProgMonHandler	Client callback function
Return values		
Name	**Type**	**Description**
promise	Status	Returns a promise of type Status with the status of the request
Parameters (callback)		
Name	**Type**	**Description**
args	ProgramState	The callback function will receive a parameter of type ProgramState
Example: API call		

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

Example: Return Data		
a1 = {state: true, errorCode: '', errorDescription: ''}		
Example: Output Data		
foo = {clientData: 'prog_mon_data_exchange', programName: 'MOTN_FWD', taskName: '', executionState: 3, activeLineNumber: 0} **Note:** executionState = 3, is an ENUM stating that the program is aborted. More information can be found in section:7.6.17		

⚠ Caution
top.frpl.startActiveProgramMonitor is available on the robot controller software V9.40P/65 or after. If a plugin uses these APIs, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/64 or older.

Note
The rate of data return for this API is very fast and recurring (around 250 milliseconds). Meaning the callback function passed to this API will get executed every 250 milliseconds. This also means that all the operations defined inside the callback function will also be executed every 250 milliseconds.

7.5.52 top.frpl.stopActiveProgramMonitor

top.frpl.stopActiveProgramMonitor

Overview		
Method to stop monitoring active program data.		
Function name		
frpl.stopActiveProgramMonitor(callback)		
Parameters		
Name	**Type**	**Description**
callback	IActiveProgMonHandler	Client callback function
Return values		
Name	**Type**	**Description**
promise	Status	Returns a promise of type Status with the status of the request
Example: API call		

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

Example: Return Data		
a1 = {state: true, errorCode: '', errorDescription: ''}		

> ⚠ **Caution**
> top.frpl.stopActiveProgramMonitor is available on the robot controller software V9.40P/65 or after. If a plugin uses these APIs, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/64 or older.

7.5.53 top.frpl.stopAllMonitors

top.frpl.stopAllMonitors

Overview		
Clear the monitors initialized by the robot controller.		
Function name		
frpl.stopAllMonitors()		
Parameters		
Name	**Type**	**Description**
N/A	N/A	N/A
Return values		
Name	**Type**	**Description**
promise	Status	Returns a promise of type Status with the status of the request
Example: API call		
let a1 = await top.frpl.stopAllMonitors();		
Example: Return Data		
a1 = {state: true, errorCode: '', errorDescription: ''}		

> ⚠ **Caution**
> top.frpl.stopAllMonitors is available on the robot controller software V9.40P/65 or after. If a plugin uses these APIs, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/64 or older.

7.5.54 top.frpl.isEnabled

top.frpl.isEnabled

Overview		
Function to check if the API is ready to be used.		
Function name		
frpl.isEnabled ()		
Parameters		
Name	**Type**	**Description**
N/A	N/A	N/A
Return values		
Name	**Type**	**Description**
promise	Boolean	Returns a promise of type Boolean with the status of the request.
Example: API call		
let a1 = await top.frpl.isEnabled();		
Example: Return Data		
a1 = true		

> ⚠ **Caution**
> top.frpl.isEnabled is available on the robot controller software V9.40P/65 or after. If a plugin uses these APIs, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/64 or older.

7.6 DATA TYPES IN top.frpl FUNCTIONS

7.6.1 Status

Data Type name		
Status		
Properties		
Name	**Type**	**Description**
state	boolean	Status of the request from robot controller
errorCode	string	Error code of the request from robot controller
errorDescription	string	Description of the error code from robot controller

7.6.2 IOType

Data Type name
IOType (Enumeration)

Properties			
Name	**Type**	**Value**	**Description**
DI	enum	1	Digital Input
DO	enum	2	Digital Output
AI	enum	3	Analog Input
AO	enum	4	Analog Output
RI	enum	8	Robot Input
RO	enum	9	Robot Output
WI	enum	16	Weld Input
WO	enum	17	Weld Output
GI	enum	18	Group Input
GO	enum	19	Group Output

7.6.3 IOPoint

Data Type name		
IOPoint		
Properties		
Name	**Type**	**Description**
type	IOType	The type of the IO this point is a member
point	number	The logical number of the robot controller's IOPoint
value	number	The last updated value of the IOPoint

7.6.4 IIOHandler

Data Type name		
IIOHandler (Interface Type)		
Properties		
Name	**Type**	**Description**
args	OnIOArgs	I/O information

7.6.5 OnIOArgs

Data Type name		
OnIOArgs		
Properties		
Name	**Type**	**Description**
ioPoint	IOPoint	The IOPoint that changed
clientData	any	Return the client's data originally supplied during the start monitoring call

7.6.6 RobotXyzWprPos

Data Type name		
RobotXyzWprPos		
Properties		
Name	**Type**	**Description**
config	string	The configuration, posture, of the robot controller for the specified robot world position
groupNum	number	The robot group number for this world position
x	number	The x distance from the world position's origin
y	number	The y distance from the world position's origin

z	number	The z distance from the world position's origin
w	number	The angle of rotation about the x-axis for this group's world position
p	number	The angle of rotation about the y-axis for this group's world position
r	number	The angle of rotation about the z-axis for this group's world position

7.6.7 AxisTypes

Data Type name			
AxisTypes (Enumeration)			
Properties			
Name	**Type**	**Value**	**Description**
linear	enum	1	Linear Axis
rotary	enum	2	Rotary Axis

7.6.8 RobotJointLimits

Data Type name		
RobotJointLimits		
Properties		
Name	**Type**	**Description**
axisTypes	AxisTypes[ ] [ ]	A 2-dimensional array of enum type AxisTypes. Array # 1 will contain the data from robot group # 1, array # 2 will contain the data from robot group # 2 and so on.
lowerLimits	Number[ ] [ ]	A 2-dimensional number array. Array # 1 will contain the data from robot group # 1, array # 2 will contain the data from robot group # 2 and so on.
upperLimits	Number[ ] [ ]	A 2-dimensional number array. Array # 1 will contain the data from robot group # 1, array # 2 will contain the data from robot group # 2 and so on.

7.6.9 RobotJointAngles

Data Type name		
RobotJointAngles		
Properties		
Name	**Type**	**Description**
groupNum	Number	The requested group number of the robot
joints	Number [ ]	An array containing the joint angles of the requested robot group

7.6.10 OnJointPosChange

Data Type name		
OnJointPosChange		
Properties		
Name	**Type**	**Description**
clientData	any	Data sent by the client, during the start monitoring call, that is simply being passed back to the client
Functions		
Defination	**returns**	**Description**
get (groupNum: Number)	Array (Map<string, string>)	The function takes the group number as input from the user, and fetches the joint position data for that group. The return value is an array containing map as shown below,

		[{ "J1": "24.015", "J1_units": "deg" }, { "J2": "7.101", "J2_units": "deg" }, : : { "J6": "-4.101", "J6_units": "deg" },]

7.6.11 IJointPosHandler

Data Type name		
IJointPosHandler		
Properties		
Name	**Type**	**Description**
args	OnJointPosChange	Wrapper class for robot controller joint position change events

7.6.12 OnWorldPosChange

Data Type name		
OnWorldPosChange		
Properties		
Name	**Type**	**Description**
clientData	any	Data sent by the client, during the start monitoring call, that is simply being passed back to the client
Functions		
Defination	**returns**	**Description**
get (groupNum: Number)	Array (Map<string, string>)	The function takes the group number as input from the user, and fetches the joint position data for that group. The return value is an array containing map as shown below, [{ "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

Data Type name		
IWorldPosHandler		
Properties		
Name	**Type**	**Description**
args	OnWorldPosChange	Wrapper class for robot controller world position change events

7.6.14 RobotVarInfo

Data Type name		
RobotVarInfo		
Properties		
Name	**Type**	**Description**
programName	String	The program name in the robot controller

variableName	String	Name of the variable in the program
value	any	The last updated value of the Variable data

7.6.15 onVarArgs

Data Type name		
onVarArgs		
Properties		
Name	**Type**	**Description**
varInfo	RobotVarInfo	Variable information
clientData	any	Data sent by the client, during the start monitoring call, that is simply being passed back to the client

7.6.16 IVarHandler

Data Type name		
IVarHandler		
Properties		
Name	**Type**	**Description**
args	onVarArgs	Wrapper class for robot controller's KAREL variable change events

7.6.17 ExecutionState

Data Type name		
ExecutionState		
Properties		
Name	**Type**	**Description**
na	0	not active
running	1	Program is running
paused	2	Program is paused
aborted	3	Program is aborted

7.6.18 ProgramState

Data Type name		
ProgramState		
Properties		
Name	**Type**	**Description**
clientData	any	data passed by the client
programName	string	Program name (main)
taskName	string	Task (sub-program) name
executionState	ExecutionState	An enumeration defining the program state types
activeLineNumber	number	Line number that is running

7.6.19 IActiveProgMonHandler

Data Type name		
IActiveProgMonHandler		
Properties		
Name	**Type**	**Description**
args	ProgramState	Wrapper class defining robot program's current state

7.7 ADD MENUS

This section describes how to add plugin screens to the menu.
On the Tablet TP, the menu can be opened from the menu button at the top left of the screen. Screen developers can add the menu of the created screen to the displayed plugin menu.

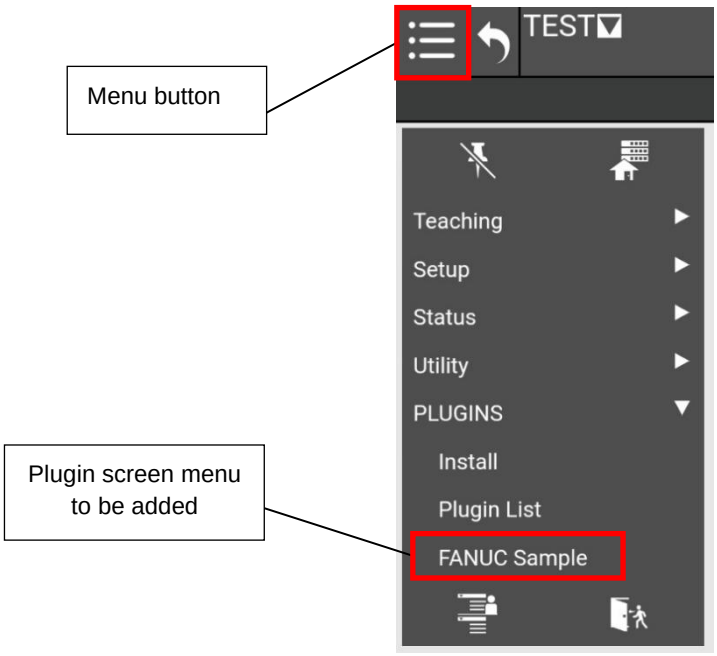


Fig. 7.7 (a) Menu of plugin screen

Screen developers need to create an XML file that describes menu configuration to add created plugin screen to the menu.

7.7.1 Creation of XML File

Screen developers can generate language-specific XML files by building an LXML file that describes the menu configuration. Building an LXML file will generate the XML files for each language. For details on how to build an LXML file, please see section 7.2 "Build Tools".

The LXML file that describes the menu structure must be named PLMENU.LXML.
PLMENU.LXML consists of the <FanucMenu> tag that describes the name of the plugin software and the <FanucMenuItem> tag that describes the menu items.

The following is a description example when adding the item "Sample" to the menu as shown in Figure 7.7 (a).

Description example of PLMENU.LXML

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

Note
- Always make sure to name the LXML file as PLMENU.LXML. If it has a different name, the menu cannot be added.
- It is allowed to describe only 1 <FanucMenu> tag in PLMENU.LXML file.

The parameters required for each tag are as follows.

<FanucMenu> tag
- Name: Company name_Plugin software name

<FanucMenuItem> tag
- Header: Menu item name
- Screen: URL of created special screen (Company name_Plugin software name / HTML file name)

Note
The screen URL described in the Screen parameter must be within 58 characters (including symbols). If it exceeds 58 characters, the screen may not be displayed correctly.

The components of the dictionary file (utx) created in "7.2 Build Tool" can be used for the Name parameter and Header parameter. When an LXML file is created using the dictionary components, the parts described in the dictionary components are replaced with the corresponding language during the building process.

Below is an example where the Header parameter is described in a dictionary component.

Dictionary component example
English dictionary: sampleeg.utx
Dictionary component: $-, menu_item_c ‘Sample’

Description example of PLMENU.LXML when using the dictionary components

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

Example of the generated XML file (English)

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

The XML file generated by the building process will be automatically included in the package. By installing the plugin software, the described menu configuration is added to the plugin menu.

Note

If the dictionary components are not used, the character string described in the LXML file will be written as is in the XML file. Please create LXML files using the dictionary components to change the text language according to the language selection on the robot controller.

7.7.2 How to add multiple screens to the menu

To add multiple screens to the menu, it is necessary to describe multiple <FanucMenuItem> tags.

Example of the PLMENU.LXML description when adding multiple screens

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



Fig 7.7.2 (a) Addition of multiple menus

It is also possible to divide the sections by nesting the <FanucMenuItem> tags.

Example of the PLMENU.LXML description for dividing sections

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

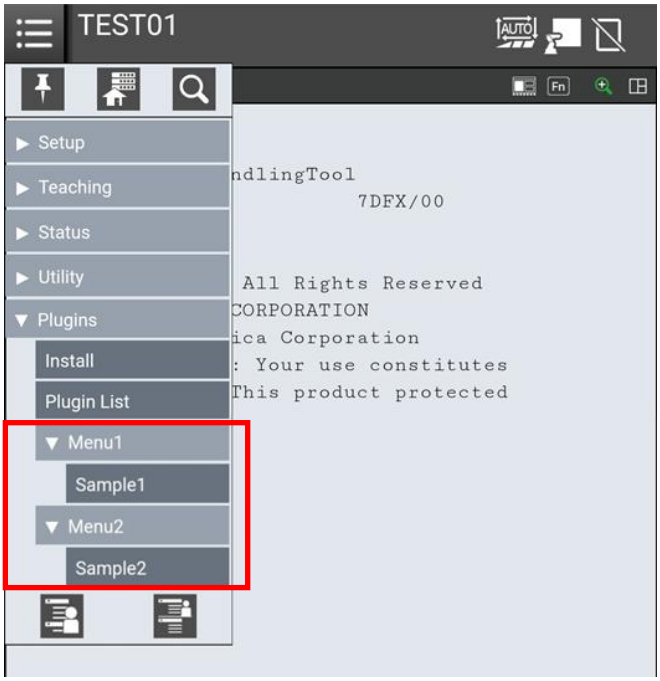


Fig 7.7.2 (b) Adding a menu with divided sections

7.8 VERIFICATION OF CREATED SCREEN USING ROBOGUIDE

The providers of plugin software can verify plugin screens specifically for the equipment and detail screens of plugin instruction, which is described later, on the Tablet TP window on ROBOGUIDE.

7.8.1 Preparations

Please use ROBOGUIDE Version 9 Rev.L or later.
When create a workcell, please select “V9.40” for the robot software version.

7.8.2 How to Use Tablet TP Window in ROBOGUIDE

Please open the ROBOGUIDE workcell and click [Tablet TP] button. Tablet TP window will be shown and then you will be able to perform the same operation as a real Tablet TP. Please install the plugin package file (*.ipl) you created and verify plugin screens specifically for the equipment and detail screen of plugin instruction. Please see chapter 3 of this document for the procedure to install a plugin package file.

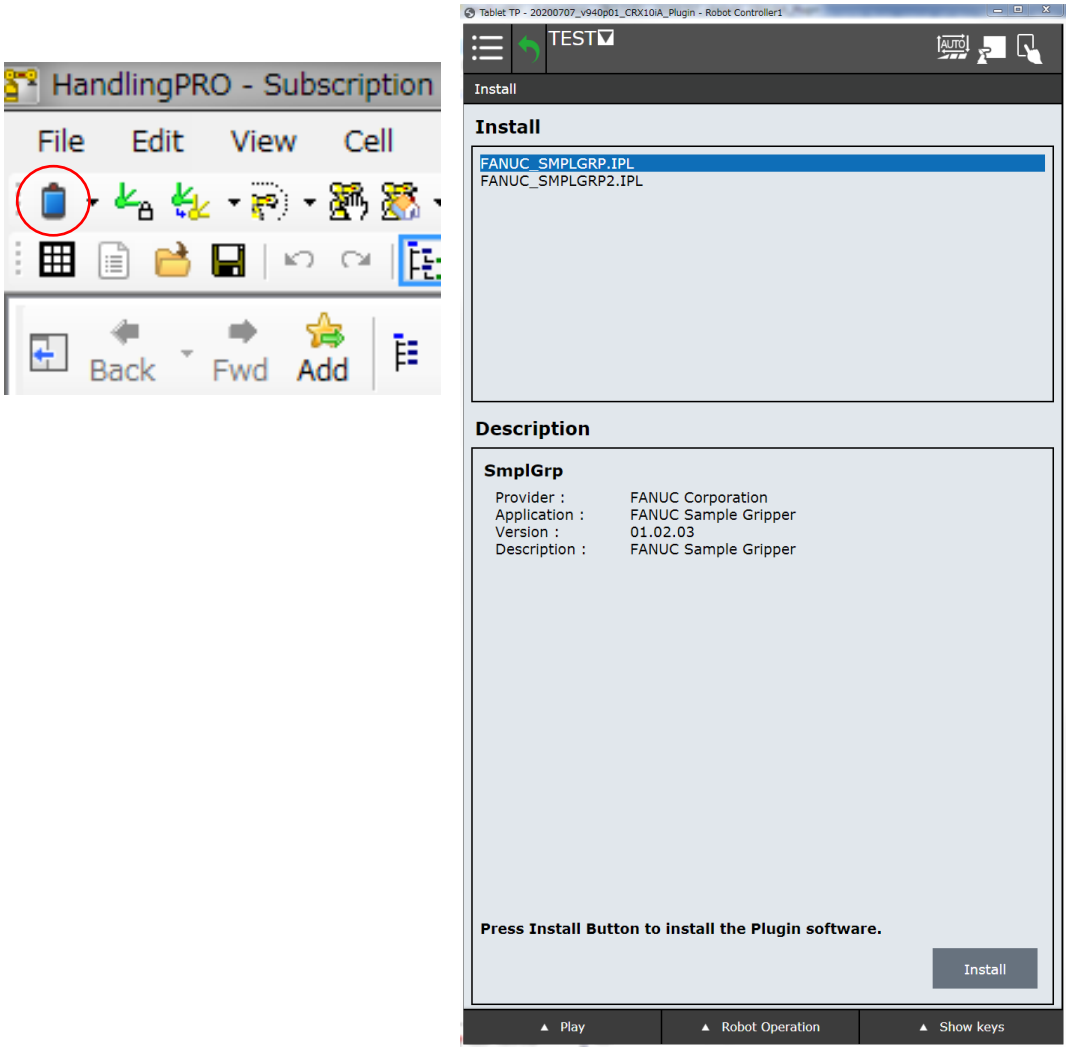


Fig 7.8.2 (a) Tablet TP window on ROBOGUIDE

Note

The Tablet TP window on the ROBOGUIDE workcell does not allow you to open the developer tools on a browser. If you want to use the developer tools, launch iRProgrammer in the robot home page with a PC browser. Refer to chapter 13 “OPERATIONS WITHOUT A TEACH PENDANT” in the R-30iB Plus OPERATOR’S MANUAL (Basic Function) (B-83284EN) for more details how to launch the iRProgrammer.

8 CREATION OF PLUGIN INSTRUCTIONS

New plugin instructions for controlling the tools can be added. This chapter describes how to create plugin instructions.

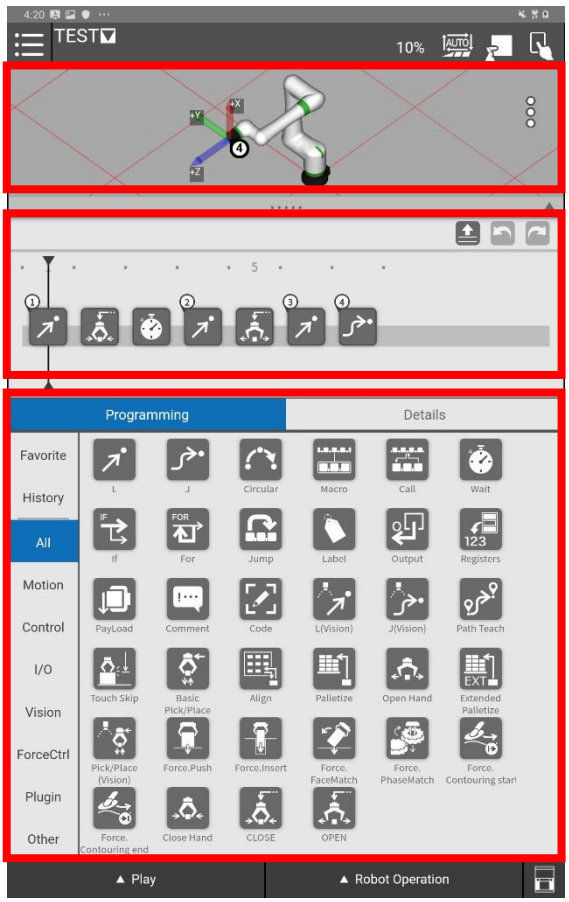


Fig. 8 (a) TP Program Editor Screen

8.1 CONSTRUCTION FILES

Create the following files to add plugin instructions.

Table 8.1 (a) Construction Files

Content	File name	Contents of Files
Plugin instruction definition file	(Arbitrary Name).lxml	File to define the following items. • Icon type of plugin instruction • Icon category of plugin instruction • Icon image data of plugin instruction • Icon name text of plugin instruction to display on icon pallet • Program that runs when plugin instruction is executed • Details screen of plugin instruction • Specify an initial value when plugin instruction icon is dropped on program line • Callback function name for details screen display

		"(Arbitrary Name)+(suffix for each language).xml" are created by building. Only half-width alphanumeric is applicable for file name to put in. The file name is arbitrary.
Program that runs when plugin instruction is executed	IPL_(Plugin ID)_(Arbitrary Program Name).tp IPL_(Plugin ID)_(Arbitrary Program Name).pc	This is a program that runs when plugin instruction is executed. Describe the behavior when plugin instruction is executed. You can specify TP program (*.tp) or KARLE program (*.pc) for this item.
Icon image of plugin instruction	(Arbitrary Name).png	Icon image file for plugin instruction.
Details screen data of plugin instruction	(Arbitrary Name).ishtm	Details screen data of plugin instruction (HTML).
	(Arbitrary name).js	JavsScript files for details screen of plugin instruction to check the value range of argument parameters, operating the related elements and describing callback functions.
	(Arbitrary Name).css	CSS files for details screen of plugin instruction to apply own style for details screen display of plugin instruction. It is unnecessary to create this file for iHMI component.

Note

- Plugin instruction definition file name (*.lxml, *.xml), icon image file name (*.png), and detail screen data file name (*ishtm, *.stm, *.js, *.css) are all arbitrary. Alphanumeric and underscore "_" are allowed and up to 42 char or less.
- Plugin ID must be defined as "(Company Name)_(Plugin Name)", and the file name of TP program and KAREL program must be "IPL_(Plugin ID)_(Arbitrary program name)". For example, if you define the Plugin ID as **"FANUC_SmplGrp"**, the program name must be "**IPL_FANUC_SmplGrp_*****.TP" and "**IPL_FANUC_SmplGrp_*****.PC". Here "***" indicates arbitrary program name.

8.2 PLUGIN INSTRUCTION DEFINITION FILE

This section describes the procedure to create a plugin instruction. The following figure shows an example of plugin instruction.

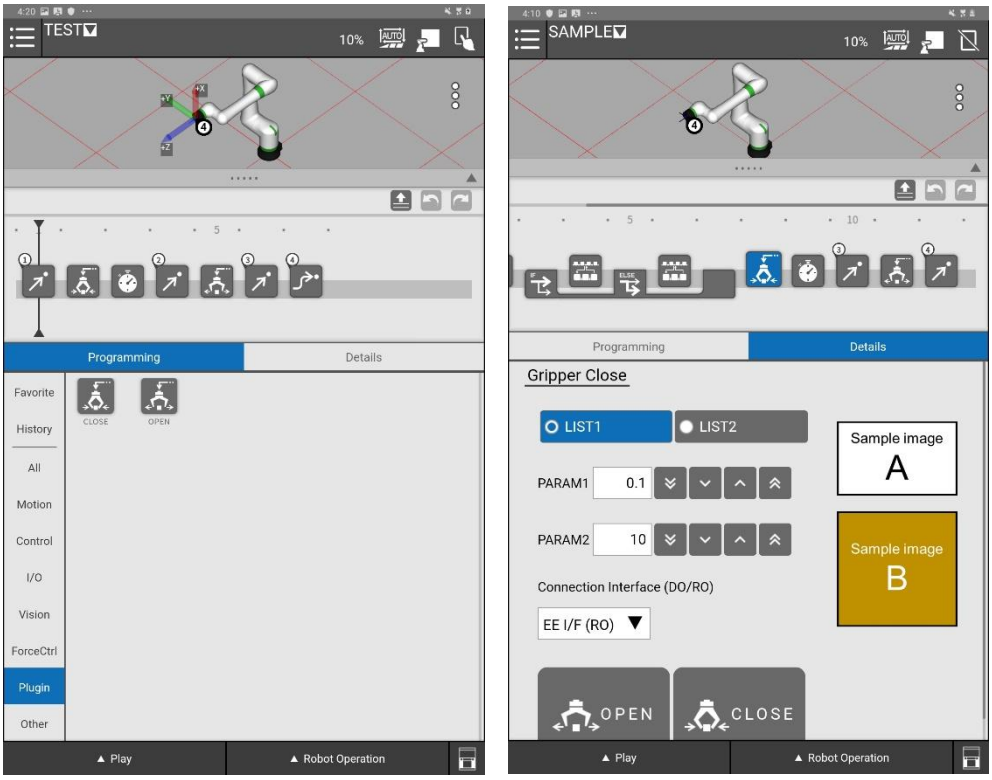


Fig. 8.2 (a) Example of plugin instruction

To create a plugin instruction, you need to define plugin instruction in XML format.

Dictionary elements defined in dictionary files (*.utx) are available in plugin instruction definition file (*.lxml) to support multiple languages. Plugin instruction definition file must have the extension *.lxml and build to embed dictionary elements. Dictionary files must be created for each language respectively.

By building using the build tool, the contents of each dictionary are embedded in XML file, and XML file of each language is generated. For the building method, please refer to section 7.2 "Build tool". Archive the generated XML files for each language in the plugin installation package file (*.ipl) for plugin installation.

Content of sampleapp.lxml (base file before build process)

```
<?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>
```

Specifying encoding="{{ienc.ienc_xml_encoding}}" at the 1st line of a base file (LXML file), the encoding attribute will be automatically replaced with a character code corresponding to each language in XML files generated by building using the build tool.

Specifies the dictionary element

Specifies the default parameter to attribution.

The content of sampleappeg.utx (English dictionary file)

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

The content of sampleappeg.utx (Japanese dictionary file)

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

The content of sampleappeg.xml (original file after build)

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

The content of sampleappkn.xml (Japanese file after build)

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

NOTE

The encoding attribute at the 1st line of a plugin instruction definition file (*.lxml) must be encoding="{{ienc.ienc_xml_encoding}}". By specifying this encoding, the encoding attribute at the 1st line of XML files generated by building using the build tool will be automatically replaced with an appropriate character code corresponding to each language.

The following contents can be described in the XML file.

- Icon type of plugin instruction
- Icon category of plugin instruction
- Icon image data of plugin instruction
- Icon name text of plugin instruction to display on icon pallet
- Program that runs when plugin instruction is executed
- Details screen of plugin instruction
- Specify an initial value when plugin instruction icon is dropped on program line
- Callback function name for details screen display

The following table shows the format of xml file.

Name of Tag		
instruction		
Attribution		
type	string	Choose the type of icon. Specify type="normal" for plugin icons. “normal" : normal icon 
section	string	Choose where to display among the instruction category tabs. “Cat_plugin” is used when it is unspecified. • cat_plugin : Plugin • cat_motion : Motion • cat_ ctrl : Control • cat_io : I/O • cat_other : Other
Required/Any		
Required		

⚠ Caution

- Specify “cat_plugin” for plugin instruction.
- The instruction category of “cat_plugin” is available on the robot controller software V9.40P/09 or after. Make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the Plugin software if the current software version is V9.40P/08 or older.

Name of Tag
Iconfile
Contents
Describing Icon Image Data File Name by Relative Path. Since it is assumed to be placed in the same folder on the robot controller, only the file name is used.
Required/Any
Required

Name of Tag
icontext
Contents
Describe instruction name, which are displayed under icon, within 16 bytes.
Required/Any
Required

Name of Tag
program
Contents
Describe the Name of TP Program (*.tp) or KAREL program (*.pc)
Required/Any
Required

Name of Tag
detail
Contents
Having detailed data on slave node, which is required for processing detailed screen display in advanced instruction. Omitted when there is no detailed screen display.
Required/Any
Any

Name of Tag
title
Contents
Describing Character String in order to Display on Title Line
Required/Any
Any

Name of Tag
src
Contents
describe stm name or him name on detailed screen display
Required/Any
Required when the attribute of the init tag is func

Name of Tag
init
Contents
• Specifies the default argument string when dragging and dropping the icon instruction to the program line. • Specifies the default argument string of the icon instruction using either (1) “param” attribute to specify a default argument string, or (2) “func” attribute to specify the name of the initialization function to call when dragging and dropping the icon instruction to the program line. (1) The “param” attribute is used as followings; - The “param” attribute specifies an argument string that is saved as a default argument value of the icon instruction. - When specifying the “param” attribute, the “func” attribute does not need to be specified. - Used as an argument of the icon instruction in an occasion of saving and executing the line in by the editor UI of the Tablet TP when dragging the icon on the program line by user. - Used as default values on the details panel input form of the icon instruction when dragging the icon icon to the program line and clicking icon by user. (2) The “func” attribute is used as followings. - The “func” attribute specifies the name of the initialization function to execute when dragging and dropping the icon instruction to the program line, and the return value is saved as the default argument string of the icon instruction. - When specifying the “func” attribute, the “param” attribute also needs to be specified in the tag. - An error “I117” is displayed when dragging and dropping the icon instruction that the “func” attribute is specified but the “param” attribute is missing in the init tag.

- Note that the return value of the initialization fuction that is specified with the “func” attribute is used when specifying both the “func” attribute and the “param” attribute in one init tag. - The initialization function (JavaScript function) that is specified with the “func” attribute must have setInstructionParam to specify a default argument string for the icon instruction.

Attribution		
param	string	Argument String
func	string	Function Name (Default Setting in Function Call)

Required/Any
Any

Name of Tag
Disp
Contents
Specify a callback function that runs when the icon on the program line is selected.
Required/Any
Required when there are setting values on the detail screen.

8.3 PROGRAM CALLED WHEN PLUGIN INSTRUCTION IS EXECUTED

Describe the behavior when of plugin instructions is executed in a TP program or KAREL program. Parameters that change the behavior of plugin instruction are set as arguments of TP program or KAREL program. Set the parameters on the details screen of plugin instruction. For details on creating the details screens, please refer to Section 8.5 "Creating Details Screen of Plugin Instruction". The parameter values can be read as Parameter[..] (AR [..]) in the TP program. For details of TP program creation, please refer to chapter 4 "Program Structure" in the B-83284EN R-30iB Plus Operator’s Manual (Basic Function).

Note

- TP program that is called when executing the plugin instruction must not have motion group by setting group mask [*,*,*,*,*,*,*,*] unless it has robot motion instruction and/or payload instruction which requires motion mask.
- Only if TP program has an instruction that requires motion mask such as robot motion instruction or payload instruction, please set only motion group #1 by setting group mask [1,*,*,*,*,*,*,*].
- TP program that is called when executing the plugin instruction must not have motion group #2 or more.
- “%NOLOCKGROUP” or “%LOCKGROUP=1” attribute must be set in all KAREL programs that are included in the plugin software.

8.3.1 Creation of a New TP Program

TP programs are created in ROBOGUIDE or robot controller environment.

The file name when creating the TP program is set to be an alphanumeric character that starts with an alphabet. The Plugin ID, which is defined in atuocfg.csv, must be defined as “(Company Name)_(Plugin Name)”, and the file name of TP program and KAREL program must be “IPL_(Plugin ID)_(Arbitrary program name)”. For example, if you define the Plugin ID as “FANUC_SmplGrp”, the program name must be “IPL_FANUC_SmplGrp_***.TP” and “IPL_FANUC_SmplGrp_***.PC”. Here “***” indicates arbitrary program name. Please refer to section “5.1 Format of Installation Definition File”. It is recommended that the program name are set to be things like “IPL_FANUC_SmplGrp_OPEN.TP” or “IPL_FANUC_SmplGrp_CLOSE.TP”. The total length of the file name specified as “IPL_(Company Name)_(Plugin Name)_(Program Name).TP” must be within 36 characters.

If there are no operation groups to be controlled in the program, move the cursor to the item of “Group mask” in the program detail screen and set "*" during TP program generation, as shown below. For example, if TP program only have logic instructions like opening or closing a gripper and does not have robot operation instructions, set all group mask to "*" in the motion group item. For details, refer to section 4.1.4 "Motion Group" in the R-30iB Plus Operator’s Manual (Basic Function) (B-83284EN).

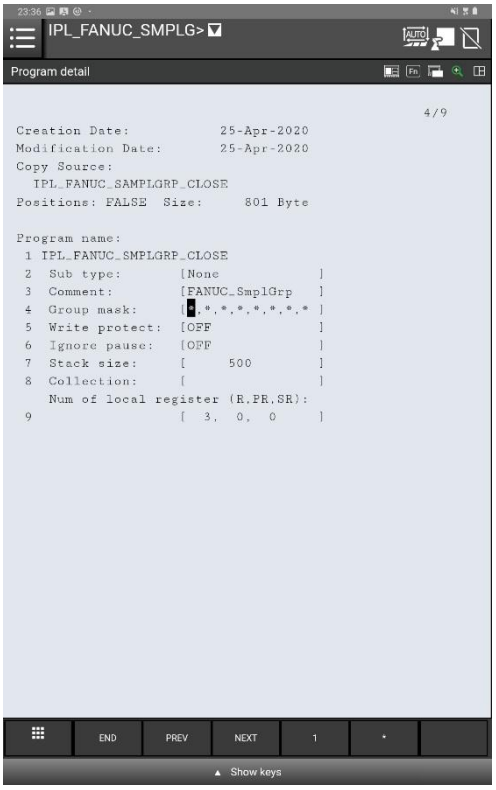


Fig. 8.3.1 (a) Group Mask Setting on the TP Program Details Screen

Note

TP program which includes Payload instruction requires to set the group mask regardless of whether it has robot operation instruction. In this case, please set the group mask to [1,*,*,*,*,*,*,*].

8.3.2 Local Registers in TP Program

The local registers (registers, character string registers, position registers) can be used as the local variables in the TP program. When using the local registers, the number of local registers to be used must be set in advance when creating the TP program.

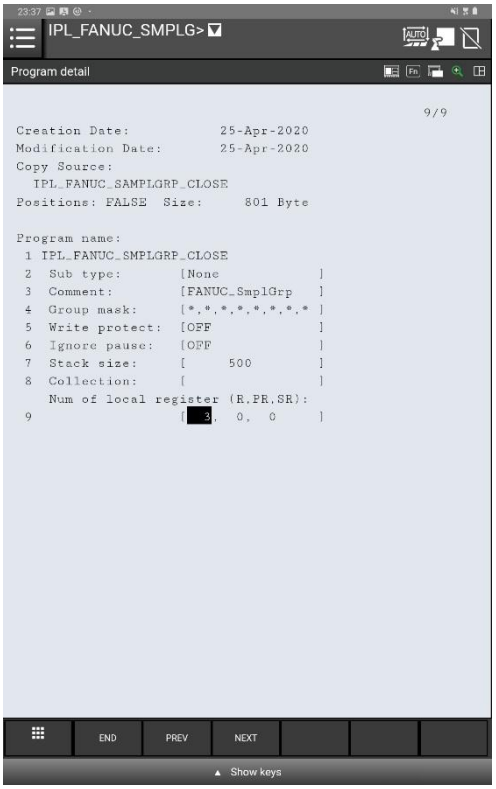


Fig. 8.3.2 (a) Number of Local Registers Setting on the TP Program Details Screen

Note

The internal memory is reserved for the number of local registers set in the program detail screen when the program is executed. Please set the minimal number for local registers and do not set more than necessary.

The format of each local register is as follows. Registers numbering from 1 to 9999 are reserved as register numbers for global registers. Since global registers are the register areas used by end-users, use local registers in TP programs for plugin instructions. The register number of the local register must be set in the range of 10001 onward (10000 + the number of set registers). For details of register specification, please refer to section 4.1.4 "Motion Group" in the R-30iB Plus Operator's Manual (Basic Function) (B-83284EN).

Table 8.3.2 (a) Local Registers

Local register name	Format in the TP program	Application
Local register	R [10001 onward (10000 + the number of set registers)]	The local registers store specific integer or decimal values. They can be used for arithmetic operation and indirect specification of the registers.
Local string registers	SR [10001 onward (10000 + the number of set registers)]	The local string register stores alphanumeric, katakana, and kanji character strings. Each register can hold up to 254 characters.
Local position register	PR [10001 onward (10000 + the number of set registers)] PR [10001 onward (10000 + the number of set registers), position register component number]	The position register is a variable that stores the position data (x, y, z, w, p, r). In the case of writing [a, b], "a" indicates the position register number, and "b" indicates the element number of the position register.

Please see the following table for TP instructions which are allowed to be used with local registers, and TP instructions which are not allowed to be used with local registers.

Table 8.3.2 (b) TP instructions which are allowed to be used with local registers

Instruction	TP instructions which are allowed to be used with local registers
Control Instruction	Registers, I/O, IF/SELECT, WAIT, JMP/LBL, CALL, Miscellaneous, Skip, String, Payload, FOR/ENDFOR, UFRAM_NUM, UTOOL_NUM, UFRAME[], UTOOL[]
Motion Instruction	L PR[], R[] mm/sec
Additional Motion Instruction	ACC, Offset,PR[], Incremental, Tool offset,PR[], DO[ ]=⋯, RO[ ]=⋯, GO[ ]=⋯, AO[ ]=⋯, TB (Time Before), TA (Time After), DB (Distance Before)

表 8.3.2 (c) TP instructions which are not allowed to be used with local registers

Instruction	TP instructions which are not allowed to be used with local registers
Control Instruction	Offset, Tool offset
Motion Instruction	CNT R[]
Additional Motion Instruction	Offset, Tool offset

Note

The above table is the specification as of the software version V9.40P/01. TP instructions which are allowed to be used with local registers may be increased in future.

8.3.3 Plugin Specific Alarm Message

Predefined alarm message specific to the Plugin may be posted by a TP program or a KAREL program. Alarm messages may be defined in multiple languages, that the alarm text in the appropriate language depending on the currently selected language may be posted. Alarm messages are defined specific to the Plugin, and therefore conflicts may be avoided with other plugins.

A plugin specific alarm text is displayed in the following format with “PLUG-001”.

```
PLUG-001 "Predefined alarm message text specific to the plugin"
```

An arbitrary number of alarm texts may be defined with alarm severity (warning, pause, abort) for each alarm. The displayed alarm messages are stored in the alarm history.

Please refer to the following sections for detail explanations.

⚠ Caution

- Plugin specific alarm feature is available on the robot controller software V9.40P/06 or after. If a plugin uses this feature, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/05 or older.
- Because the individual user defined alarms are not described in FANUC manuals, and therefore, they should be described in the plugin manual that is provided by the plugin manufacturer.

8.3.3.1 Data file that defines plugin specific alarm texts

An ASCII file defines the alarm texts. It has a fixed file name, “plgalm.dat”, and is to be enclosed in the plugin software installation file (*.IPL file).

The plgalm.dat file defines the English message texts as the default language. To support multiple language, a new file for each language should be created, and included in the IPL file, with the file name appended by the two letters indicating the language (e.g. KN for Japanese). For example, “plgalmkn.dat” for Japanese language.

The data format of the plgalm.dat file should be as follows.

```
NAME,”Plugin Name for alarm display”
$1,PAUSE,”Message text #1”
$2,WARN,”Message text #2”
$3,ABORT,”Message text #3”
...
END
```

The first line must always start with “NAME”. It defines the Plugin Name that is displayed in the cases “format” parameter is either 1 or 2. (see the next section)

The next lines after the first line define the alarm severity and the alarm texts corresponding to the alarm number.

- The number after the $ (dollar) sign indicates the alarm number.
- Delimited by “,” (comma), the next string defines the alarm severity.
  - WARN: Warning severity. The program execution will continue after the alarm text is displayed.
  - PAUSE: Pause severity. The program execution will be stopped after the alarm text is displayed.
  - ABORT: Abort severity. The program execution will be aborted after the alarm text is displayed.
- The string after the next delimiter “,” (comma) is the alarm text, that should be enclosed by “ (double quote). Adjust the text length to be about 30 letters (30 bytes). The two letters “%s”, if inserted within the text, will be replaced by the string specified as the fourth parameter for ipl_fanuc_sys_erpost or IPL_ERPOST sub-program.

The last line of the file should be always “END”.

8.3.3.2 Posting a plugin specific alarm message by a TP program

A TP program may CALL the following sub-program (KAREL program) to post a user defined alarm message.

```
CALL ipl_fanuc_sys_erpost(Plugin ID, Alarm_Number, Cause_code, Param, Format)
```

The meaning of each parameter is as follows.

- Plugin ID (string) : Indicates the Plugin ID defined in Autocfg.csv file in the IPL file
- Alarm_number (integer) : Alarm number as defined in the alarm text file (plgalm.dat). Please refer to the previous section for the format of the alarm text file.
- Cause_code (integer) : Specify Zero (0) for this parameter.
- param (string) : this string replaces “%s” in the message text.
- Format (integer) : Defines the format of alarm message displayed as follows.
  0: PLUG-001 “Message text #3”
  1: PLUG-001 “[PluginName-003] Message text #3”
  2: PLUG-001 “Message text #3 [PluginName-003]”

If no text is found for the given alarm number, it always shows the message as “[PluginName-nnn]”, with WARNING severity.

8.3.3.3 Posting a plugin specific alarm message by a KAREL program

It is also possible to post a Plugin specific alarm message from a KAREL program. The routine definition of IPL_ERPOST is as follows.

Purpose:

Posts a Plugin specific alarm message corresponding to the specified alarm number.

Syntax:

IPL_ERPOST (*Plugin_id, Alarm_number, Cause_code, Param, Format*) from IPL_FANUC_SYS_ERPOST
Input/Output Parameters :
[in] *Plugin_id* : STRING
[in] *Alarm_number* : INTEGER
[in] *Cause_code* : INTEGER
[in] *Param* : STRING
[in] *Format* : INTEGER

Details:

- *Plugin id* indicates the Plugin ID defined in Autocfg.csv in the IPL file.
- *Alarm number* specifies the alarm number as defined in the alarm text definition file (plgalm.dat). Please refer to the previous section for the format of plgalm.dat.
- Please specify zero (0) for *Cause_code*.
- *Param* specifies the string to replace "%s" in the message text.
- *Format* defines the format of alarm message displaed as follows.
  0: PLUG-001 "Message text #3"
  1: PLUG-001 "[PluginName-003] Message text #3"
  2: PLUG-001 "Message text #3 [PluginName-003]"

If no text is found for the given alarm number, it always shows the message as "[PluginName-nnn]", with WARNING severity.

8.3.4 Example of TP Program Creation

This section shows the device configuration for signal communication between the gripper control unit and the robot controller in controlling the gripper using DI / DO signals, and an example of mounting the TP program of a plugin instruction. The end-user controls the gripper open/close by calling the TP program of a plugin instruction (IPL_FANUC_SmplGrp_Ctrl.TP).

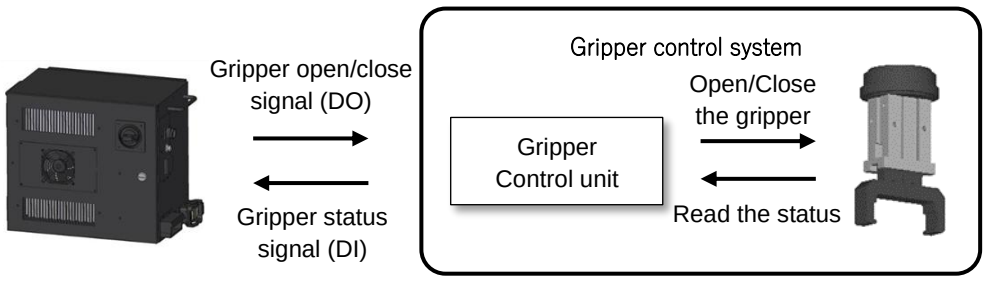


Fig. 8.3.4 (a) Outline of the plugin instruction program

I/O signal and alarm display

In this section's example, it is assumed that the DO signal (gripper open/close signal) is set by the argument parameter to control the opening and closing of the gripper.

- Argument parameter value 1 indicates to close the gripper by setting DO[101]=OFF / DO[102]=ON.

- Argument parameter value 0 indicates to open the gripper by setting DO[101]=ON / DO[102]=OFF.

By reading the DI signal (gripper status signal), you can check the status of the gripper.
- Gripper status DI[101]=ON indicates that the gripper is in an error state.
- Gripper status DI[1]=OFF indicates that the gripper can operate normally.

The index number of the DI / DO are set on the plugin screen, described as an implementation example in chapter 6 " Sharing the setting values between the plugin screen and instructions", and it is assumed that the set values are stored in the KAREL variables set in section 6.1 "Setting Items and KAREL Variable Definitions".

The behavior of the TP program

The gripper status is checked before the gripper is opened or closed the gripper, and if an error occurs, the following user alarm is displayed.

Displayed alarms:

'PLUG-001 The gripper has an error'

```
/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
```

Fig. 8.3.4 (b) Example of TP program implementation

Note

- Plugin ID must be defined as "(Company Name)_(Plugin Name)", and the file name of TP program and KAREL program must be "IPL_(Plugin ID)_(Arbitrary program name)". For example, if you define the Plugin ID as **"FANUC_SmplGrp"**, the program name must be "**IPL_FANUC_SmplGrp_*****.TP" and "**IPL_FANUC_SmplGrp_*****.PC". Here "***" indicates arbitrary program name.
- The parameter instruction in TP program allow the number of chars up to 30 or less for parameter name. In case to specify KAREL variable, KAREL program name and KAREL variable name must be specified in this instruction as below. Plugin ID, KAREL program name, and KAREL variable name must be defined in accord with this limitation.

```
Ex) R[10001]=$[IPL_FANUC_SmplGrp_Cfg]OpenDo
                1234567890123456789012345678 90
```

8.3.5 External Vision Interface

By using External Vision Interface (A05B-2670-R897), an external vision system other than FANUC can be executed from TP program. The vision system can be connected to the robot controller via Ethernet. Note that the vision system must support the External Vision Interface. For details, please refer to the External Vision Interface OPERATOR'S MANUAL (B-84244EN).

8.3.6 Setting User Socket Messaging by a KAREL Program

A KAREL program may call the KAREL routine to configure the User Socket Messaging function (A05B-2600-R648). Please refer to chapter 10 "SOCKET MESSAGING" of the Ethernet Function OPERATOR'S MANUAL (B-82974EN) for details of the User Socket Messaging function.

⚠ Caution

The KAREL routine "IPL_SETUPSM" is available on the robot controller software V9.40P/31 or after. If a plugin uses this feature, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the Plugin software if the current software version is V9.40P/30 or older.

The KAREL routine IPL_SETUPSM with the routine definition indicated below may be called in a KAREL program to setup User Socket Messaging.

Purpose:

Configures necessary settings for the User Socket Messaging function.

Syntax:

IPL_SETUPSM (*Mode, Tag_num, Port_num, Ip_addr, Comment, Use_udp, Status*) from IPL_FANUC_SYS_SETUPSM
Input/Output Parameters :
[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

Details:

- *Mode* indicates the Client tag or Server tag.
  1: Client tag
  2: Server tag
- *Tag_num* indicates the tag number. The valid range is 1 to 8.
- *Port_num* indicates the server port number to use for Socket Messaging. The valid range is 0 to 32767.
- *Ip_addr* specifies the server IP address or host name. This parameter only requires to be set when it specifies "1: Client tag" for the *mode*. When it specifies "2: Server tag" for the *mode*, please set null string for this parameter.
- *Comment* specifies comments. If you don't need to set the comment, please set null string for this parameter.
- *Use_udp* indicates whether to use UDP.
  FALSE: Not use UDP
  TRUE: Use UDP
- *Status* returns the status. 0 (zero) indicates the setup was done successfully. Non-zero value indicates an error occurs during the process. The following table shows alarm messages corresponding each possible status value.

Table 8.3.6 (a) Alarm messages corresponding each status value

Status	Alarm	Message	Remedy
169006	PLUG-006	Invalid argument(IPL_SETUPSM,1)	Check 1st argument
169007	PLUG-007	Invalid argument(IPL_SETUPSM,2)	Check 2nd argument
169008	PLUG-008	Invalid argument(IPL_SETUPSM,3)	Check 3rd argument
169009	PLUG-009	Invalid argument(IPL_SETUPSM,4)	Check 4th argument
169010	PLUG-010	Invalid argument(IPL_SETUPSM,5)	Check 5th argument
169011	PLUG-011	Invalid argument(IPL_SETUPSM,6)	Check 6th argument
169012	PLUG-012	R648 is required	Need to install R648 (User Socket Message)

8.3.7 Setting Payload Information by a KAREL Program

A KAREL program may call the KAREL routine "IPL_SETPLD" to setup the comment, weight, gravity center position of the load, and inertia about its gravity center. The routine definition of IPL_SETPLD is as follows. The IPL_SETPLD routine is only available to be called in a KAREL program.

> **⚠ Caution**
> The KAREL routine "IPL_SETPLD" is available on the robot controller software V9.40P/41 or after. If a plugin uses this feature, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the Plugin software if the current software version is V9.40P/40 or older.

Purpose:

Allows a KAREL program to setup payload information to the specified payload schedule.

Syntax:

IPL_SETPLD (*Gnum, Shc_num, Comment, Mass, Cog, Inertia, Unit_type, Flg_lim, Status*) from IPL_FANUC_SYS_PLGUTIL
Input/Output Parameters :
[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

Details:

- *Gnum* specifies the motion group number to setup the payload.
- *Sch_num* specifies the payload schedule number to setup.
- *Comment* specifies comments. If you specified uninitialized STRING variable for this parameter, it will be ignored.
- *Mass* specifies the weight of the load in [kg].
- *Cog* specifies the gravity center position of the load. *Cog[1]* for X direction, *Cog[2]* for Y direction, and *Cog[3]* for Z direction. The unit depends on the value specified in *Unit_type*.
- *Inertia* specifies the inertia about its gravity center. *Inertia[1]* about X direction, *Inertia[2]* about Y direction, and *Inertia[3]* about Z direction. The unit depends on the value specified in *Unit_type*.
- *Unit_type* specifies the unit to set *Cog* and *Inertia* as follows.
  - *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* returns the result of overload check. If it exceed the capacity, *Flg_lim* returns TRUE, otherwise it returns FALSE.
- *Status* returns the status. If Status returns non-zero value, you may use POST_ERR to show an appropriate alarm.
  - *Status*=0 indicates the setup was done successfully.
  - *Status*=15170 indicates the setup was done successfully, but the load is close to the capacity.
  - *Status*=15171 indicates the setup was done successfully, but it is overload.
  - *Status*=15671 indicates the setup was failed.
  - *Status*=169025 indicates the setup was failed due to incorrect parameters specified.
  - Other non-zero values indicates an internal error occurred.

8.3.8 Getting Payload Information by a KAREL Program

A KAREL program may call the KAREL routine “IPL_GETPLD” to get the comment, weight, gravity center position of the load, and inertia about its gravity center. The routine definition of IPL_GETPLD is as follows. The IPL_GETPLD routine is only available to be called in a KAREL program.

> ⚠ **Caution**
> The KAREL routine “IPL_GETPLD” is available on the robot controller software V9.40P/41 or after. If a plugin uses this feature, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the Plugin software if the current software version is V9.40P/40 or older.

Purpose:

Returns payload information in the specified payload schedule.

Syntax:

IPL_GETPLD (*Gnum, Shc_num, Comment, Mass, Cog, Inertia, Unit_type, Status*) from IPL_FANUC_SYS_PLGUTIL
Input/Output Parameters :
[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

Details:

- *Gnum* specifies the motion group number to get the payload.
- *Sch_num* specifies the payload schedule number to get.
- *Comment* returns comments in the specified payload schedule.
- *Mass* returns the weight of the load in [kg] in the specified payload schedule.
- *Cog* returns the gravity center position of the load in the specified payload schedule. *Cog[1]* for X direction, *Cog[2]* for Y direction, and *Cog[3]* for Z direction. The unit depends on the value specified in *Unit_type*.
- *Inertia* returns the inertia about its gravity center in the specified payload schedule. *Inertia[1]* about X direction, *Inertia[2]* about Y direction, and *Inertia[3]* about Z direction. The unit depends on the value specified in *Unit_type*.
- *Unit_type* specifies the unit to read *Cog* and *Inertia* as follows.
  - *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* returns the status. If Status returns non-zero value, you may use POST_ERR to show an appropriate alarm.
  - *Status*=0 indicates data acquisition of the payload information was successfully done.
  - *Status*=169026 indicates the data acquisition was failed due to incorrect parameters specified.
  - Other non-zero values indicates an internal error occurred.

8.3.9 Getting MAC Address by a KAREL Program

A KAREL program may call the KAREL routine “IPL_GETMACADDR” to get MAC address of the robot controller ethernet port #1. MAC address is a unique information to every single robot controller so you may use for individual identification. The routine definition of IPL_GETMACADDR is as follows.The IPL_GETMACADDR routine is only available to be called in a KAREL program.

> ⚠ **Caution**
> The KAREL routine “IPL_GETMACADDR” is available on the robot controller software V9.40P/44 or after. If a plugin uses this feature, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the Plugin software if the current software version is V9.40P/43 or older.

Purpose:

Returns MAC address of the robot controller Ethernet port #1.

Syntax:

IPL_GETMACADDR (*Mac_addr, Status*) from IPL_FANUC_SYS_PLGUTIL
Input/Output Parameters :
[out] *Mac_addr* : STRING
[out] *Status* : INTEGER

Details:

- *Mac_addr* returns MAC address of the robot controller Ethernet port #1 in STRING data, for example ”00:e0:e4:80:8f:2d”.
- STRING variable to be specified for *Mac_addr* must have 17 or more characters as MAC address has 17 characters.
- *Status* returns the status. If Status returns non-zero value, you may use POST_ERR to show an appropriate alarm.
  - *Status*=0 indicates data acquisition of MAC address was successfully done.

- *Status*=12337 indicates the data acquisition was failed because it was called in ROBOGUIDE environment. IPL_GETMACADDR routine does not work in ROBOGUIDE.
- Other non-zero values indicates an internal error occurred.

8.4 CREATION OF ICON FOR PLUGIN INSTRUCTION

Icons to be displayed on the editor screen must be created in PNG format with a transparent background. The following are the necessary contents for icons;



- Image Format: Transparent PNG format
- Size: 256px × 256px (Square Shape)
- Color: 1 to 2 colors
- Color of Background: Transparent
- File Name: Alphabets

8.5 CREATING DETAILS SCREEN OF PLUGIN INSTRUCTION

Display a details screen for the setting parameters of each plugin instruction. Details screen is displayed in the details area when a plugin insctruction icon is selected in the iHMI editor.
Create a details screen in HTML. Change the extension to ishtm. Please refer to section 6.3.2 "Using iHMI component" on how to create the file.
Dictionary elements defined in dictionary files (*.utx) are available in plugin instruction details screen to support multiple languages. Details screen data file must have the extension *.ishtml (*.ihtml) and build to embed dictionary elements. Dictionary files must be created for each language respectively.
By building using the build tool, the contents of each dictionary are embedded in *.stm (*.htm) file, and *.stm (*.htm) file of each language is generated. For further information about how to build the base file, please refer to section 7.2 "Build tool". Archive the generated *.stm (*.htm) files for each language in the plugin installation package file (*.ipl) for installation.

Content of sampleapp.ishtm (base file prior to build)

```
<!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="content">
        <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="content">
        <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>
```

Character code is retreated by SSI.

Loads iHMI Component CSS.

Loads CSS used in normal tags.

Load a JavaScript file for value range check, controls of the related elements, and Callback functions, etc.

The parameters set on the details screen are set as the arguments of the TP program.
The following is an example of the callback function for setting the initial values and the callback function for displaying the set values on the details screen for the setting. The following "dispAdvInstData" function is a callback function that displays the setting value on the Detail screen. This function's argStr contains the argument of the set subprogram.
The "callbackInputText1" and "callbackInputText2" functions are the examples when the textbox value on the Detail screen is changed. When the value of the text box on the Detail screen is changed, the "setInstructionParam" API is called to update the subprogram argument.

```
// Setting for the input from value
function dispAdvInstData(argStr) {//argStr="1,2"
      //Process for displaying

      if( argStr.length === 0 ){
          return false; //Return false in case of false for argument
      }
       var params = argStr.split(',');
       argText1 = params[0];
       argText2 = params[1];

      //Setting the input form
       document.getElementById('text1').refresh(argText1, '1', '200', false);
       document.getElementById('text2').refresh(argText2, '1', '200', false);

      //Registration for the call back function
       document.getElementById('text1').setCallback(callbackInputText1);
       document.getElementById('text2').setCallback(callbackInputText2);

       return true; // Completion of process
}
// Callback when inputting form
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 API for Creating a Detail Screen

Hereby, API is explained as it is required for acquiring program information while editing and callback when the completion of describing the screen display for iHMIeditor, when you create the detail screen display of instruction.

top.setInstructionParam

Overview		
It is the function in order to notify to ihmieditor for the completion of describing detailed screen display, by dragging and dropping on icon (calling an XML definition initialization function) or selecting an icon on the timeline (calling an XML definition drawing function).		
Function name		
setInstructionParam (argStr)		
Parameters		
Name	**Type**	**Description**
argStr	String	Specify parameters for TP program or KAREL program to be called in the plugin icon instruction.
Return value		
Name	**Type**	**Description**
N/A	N/A	N/A

Note

When passing a REAL value as an argument of plugin instruction using setInstructionParam, if the difference between the REAL value and INTEGER value is zero, the argument value is passed as an INTEGER value. In this case, please consider checking the type in Karel program of plugin instruction and then branch off based on whether it is a REAL or INTEGER.

8.6 ERROR LIST FOR PLUGIN INSTRUCTION

The error code is displayed so that the creator of plugin instruction can detect errors, such as input errors and file placement errors. Errors are detected simultaneously by multiple plugin instructions, and the errors of up to 3 plugin instructions are displayed.

Table 8.6 (a) List of Error Code

No	Error Code	Content of Error
I0xx : Plugin instruction/ file existence check		
1	I002	02 : PNG does not exist
2	I003	03 : An src tag is present, and STM does not exist
I1xx : Plugin instruction / errors on describing XML		
3	I102	02 : No iconfile tag
4	I103	03 : No icontext tag
5	I104	04 : No program tag
6	I110	10 : No type in instruction
7	I111	11 : There is no src tag, and init is func
8	I113	13 : Wrong value in type tab
9	I114	14 : Wrong value as section
10	I117	17 : An init func is present, and param settings does not exist
11	I119	19 : No instruction tag
12	I120	20 : Same TP program or KAREL program name exists
13	I123	23 : TP program name is 36 characters or more
14	I124	24 : TP program name starts with a number
I2xx : Plugin instruction/ errors on connection when acquiring XML		
15	I201	01 : Parse error when acquiring XML
16	I202	02 : Time-out error when acquiring XML
17	I203	03 : Other when acquiring XML
I5xx : Plugin instruction/ JavaScript errors		
18	I501	01 : Return of the callback function for disp is false
19	I502	02 : Return of the callback function for init is false
20	I503	03 : Function in func for init is missing

9 ERROR MESSAGES AT INSTALLATION

This chapter describes the error messages that may be posted during plugin software installation.

9.1 LIST OF ERROR MESSAGES

9.1.1 SDTL (ID = 123)

SDTL-084 Wrong data type(Line %s)

Cause: The installation definition file (Autocfg.csv) includes an illegal command or a command unavailable with the current software version of the robot controller.

Remedy: Please check the content of the installation definition file (Autocfg.csv). If it includes a command unavailable with the current software version of the robot controller, please upgrade the software to the latest version prior to installing the Plugin software.

SDTL-101 Too many Plugins defined

Cause: Number of plugins exceeded its maximum.

Remedy: Uninstall plugins that are not needed.

SDTL-102 Operation canceled

Cause: Current operation has been canceled by the operator.

Remedy: Do the operation again.

SDTL-103 Non-conforming Program Name

Cause: The Content of the Plugin file is invalid. (The name of the TP or KAREL program is not according to the rule, that it has to start with “IPL_(Plugin ID)_”.

Remedy: Please contact the provider of the plugin software.

SDTL-104 Invalid Plugin software (%s)

Cause: The Content of the Plugin file is invalid.

Remedy: Please contact the provider of the plugin software.

SDTL-105 Unsupported Robot type

Cause: This robot type is not supported for this plugin.

Remedy: This plugin is incompatible with the robot type, and cannot be loaded to this robot. Contact the provider of the plugin software to obtain the correct plugin software for the robot type.

SDTL-106 Option (%s) is required

Cause: This plugin requires the specified option.

Remedy: This plugin cannot function without the specified feature (option). Please load the specified option before installing the plugin.

SDTL-111 Error Getting menu data

Cause: The Content of the Plugin file is invalid. (Invalid file path of the menu info file)

Remedy: Please contact the provider of the plugin software.

SDTL-112 Menu data syntax Error

Cause: The Content of the Plugin file is invalid. (Invalid syntax of the menu info file)

Remedy: Please contact the provider of the plugin software.

SDTL-113 System Error

Cause: System error getting Screen ID system variable.

Remedy: Contact the FANUC representative.

SDTL-114 System Error

Cause: System error setting Screen ID system variable.
Remedy: Contact the FANUC representative.

10 TOOL I/F (CRX series only)

This chapter describes the specifications of Tool I/F equipped on the EE connector of FANUC Robot CRX series.

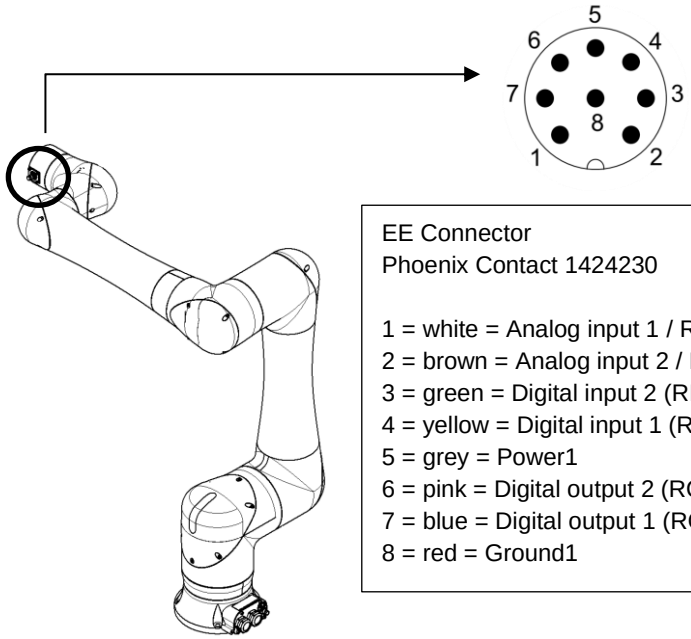


Fig. 10 (a) Tool I/F on EE Connector

⚠ CAUTION

- In case of a tool connects to the EE connector, the Tool I/F configuration must be completed before connecting the device to the EE connector. If you connect the device before the configuration is complete, the device may be damaged due to, for example, incorrect power supply voltage. The various settings of Tool I/F communication specified in the Autocfg.csv are set by the plugin software installation, and taken effect after cycling the power of the robot controller.
- FANUC Robot CRX series does not support hot swapping of the tool connections to the EE connector. For the device connects to the EE connecter, make the following precautions clear in the manual to be provided by a tool manufacturer.
  [Precautions]
  - The plugin software installation must be completed before connecting the device.
  - When detaching/attaching a tool connects to the EE connector, please follow any of these methods shown below, otherwise the device may be damaged.
    1. Connect/disconnect the device following the guidance shown in Manual Tool Change screen.
    2. Connect/disconnect the device executing Tool Detach/Attach icon instruction.
    3. Connect/disconnect the device while the robot controller is turned off.

10.1 TOOL I/F SPECIFICATIONS

Following are the specifications of the EE Connector for tools. For tool side, a generally available M8 connector (socket) can be used. Table 10.1 (a) shows some EE connector specifications but it does not necessarily have to be selected from this table. Please select proper length for your tool.

Table 10.1 (a) EE Connector specifications

Connector	Specification	Maker / dealer
Tool side	1404187 M8 socket, straight (1.5m) (Free end) 1404196 M8 socket, straight (0.3m) (Straight plug end) 1404198 M8 socket, straight (0.6m) (Straight plug end) 1404199 M8 socket, straight (1.5m) (Straight plug end) 1404191 M8 socket, angled (1.5m) (Free end) 1404201 M8 socket, angled (0.3m) (Straight plug end) 1404202 M8 socket, angled (0.6m) (Straight plug end) 1404203 M8 socket, angled (1.5m) (Straight plug end)	Phoenix Contact
Robot mechanical unit side	1424230	Phoenix Contact

Table 10.1 (b) Pin layout of EE Connector on the mechanical unit side

Pin number	Signal pin or Power pin name
1	Analog input 1 / RS485+
2	Analog input 2 / RS485-
3	Digital input 2 (RI[2])
4	Digital input 1 (RI[1])
5	Power1 (24V / 12V / Power off)
6	Digital output 2 (RO[2]) / Ground2 (*1) / Open (*2)
7	Digital output 1 (RO[1]) / Power2 (*1) (24V /12V / Power off) / Open (*2)
8	Ground1

(*1) Pin 7 can also be used as power supply pin (Power2). When using Pin 7 as Power 2, Pin 6 becomes Ground 2.
(*2) In 2A supply mode, Pin 6 and Pin 7 is Open, and the power is supplied from Pin 5 (Power1) and Pin 8 (Ground1).

Note
Power1 and Power2 are independent power supply. Power1 and Power2 must not be connected.

10.1.1 Internal Power Supply Specifications

Following are the specifications of internal power supply from Tool I/F to tools.

Set the internal power supply (Power1, Power2) to 0V, 12V or 24V.
Power1 and Power2 are independent power supply and can be set different power supply mode respectively.

The electrical specifications are shown below.

Table 10.1.1 (a) Electrical specifications of internal power supply

Parameter	Min	Typ	Max	Unit
Supply voltage in 24V mode	21.6	24	26.4	V
Supply voltage in 12V mode	10.8	12	13.2	V
Supply current in both modes (*2)	--	1000	2000 (*1)	mA
Supply current in 2A supply mode (*2)	--	2000	2000	mA

(*1) When 2A supply mode is not used, 2000 mA is for max 1 second. Duty cycle max: 10%. Average current must not exceed 1000 mA. When Power2 is used, total current of Power1 and Power2 must not exceed 2000 mA.
(*2) When connecting inductive loads, please use a protective diode.

Note
When 2A supply mode is used, Digital output1 (RO[1]) (Power2) and Digital output2 (RO[2]) (Ground2) are not available.
Power1 and Power2 are independent power supply. Power1 and Power2 must not be connected.

10.1.2 Digital Output Signal Specifications

Following are the specifications of digital output signals from Tool I/F to tools.

Digital output signals support three different modes.

Table 10.1.2 (a) Digital output signal modes

Mode	Active	Inactive
Sinking	Low	Open
Sourcing	High	Open
Push-Pull	High	Low

The electrical specifications are shown below.

Table 10.1.2 (b) Electrical specifications of digital output signals

Parameter	Min	Typ	Max	Unit
Voltage when open	-0.5	-	26	V
Current when sinking / pin	0	600	3000 (*1)	mA
Current when sourcing / pin	-	-	500 (*2)	mA

(*1) 3000 mA for max 1 second. Duty cycle max: 10%. Average current must not exceed 1000 mA.
(*2) Total current of power1 and source current of digital output signals must not exceed 2000 mA.

Note
When 2A supply mode is used, Digital output1 (RO[1]) (Power2) and Digital output2 (RO[2]) (Ground2) are not available.

Digital output signal is updated every 4ms.

10.1.3 Digital Input Signal Specifications

Following are the specifications of digital input signals from tools to Tool I/F.

Table 10.1.3 (a) Electrical specifications of digital intput signals

Parameter	Min	Typ	Max	Unit
Input voltage	-0.5	-	26	V
Logical low voltage	-	-	2.0	V
logical high voltage	5.5	-	-	V
Input resistance	-	48k		Ω

Digital input signal is updated every 4ms.

10.1.4 Analog Input Signal Specifications

Following are the specifications of analog input signals from tools to Tool I/F.

Analog input signals are non-differential and can be set to either voltage (0-10V) or current (4-20mA) The electrical specifications are shown below.

Table 10.1.4 (a) Electrical specifications of analog input signals

Parameter	Min	Typ	Max	Unit
Input voltage in voltage mode	-0.5	-	12.5	V
Input resistance (Range 0V to 10V)	-	10.6k		Ω
Resolution	-	12	-	bit
Input voltage in current mode	-0.45	-	4.5	V
Input current in current mode	-2.5	-	24	mA
Input resistance (Range 4mA to 20mA)	-	183	187	Ω
Resolution	-	12	-	bit

Note
When Analog input 1 and/or Analog input 2 is used, RS485 communication is not available.

Analog input signal is updated every 4ms.

10.1.5 RS485 Communication Specifications

Following are the specifications of RS485 communication between Tool I/F and tools.

It is selectable from the following about Baud Rates, Stop Bits and Parity.

Baud Rates	9.6k, 19.2k, 38.4k, 57.6k, 115.2k, 1M, 2M, 5M
Stop Bits	1, 2
Parity	None, Odd, Even

Termination resistor	120Ω (*1)

(*1) Please insert termination resistor of 120 ohm in the RS485 communication line of the tool.

Note
When RS485 communication is used, Analog input 1 and Analog input 2 are not available.

11 TOOL I/F COMMUNICATION (CRX series only)

This chapter describes the specifications of Tool I/F Communication Function which enables communication between the tool connected to the EE (EE1) connector of FANUC Robot CRX series, and the robot controller. This function provides the following communication between the tool connected to the EE connector and the robot controller.

- Digital input signal: 2 (RI[1], RI[2])
- Digital output signal: 2 (RO[1], RO[2])
- Analog input signal: 2 (AI)
- RS485 Serial Communication

This chapter explains the necessary settings to configure Tool I/F communication for the primary device in case the plugin software has settings of one or more devices that use Tool I/F.
For settings of Tool I/F communication for the secondary and subsequent devices in case the plugin software has settings of multiple devices that use Tool I/F, please refer to section 12.5 “Information for Plug-in Software Developers”.

Note
When RS485 communication is used, Analog input 1 and Analog input 2 are not available.

⚠ CAUTION
- In case of a tool connects to the EE connector, the Tool I/F configuration must be completed before connecting the device to the EE connector. If you connect the device before the configuration is complete, the device may be damaged due to, for example, incorrect power supply voltage. The various settings of Tool I/F communication specified in the Autocfg.csv are set by the plugin software installation, and taken effect after cycling the power of the robot controller.
- FANUC Robot CRX series does not support hot swapping of the tool connections to the EE connector. For the device connects to the EE connecter, make the following precautions clear in the manual to be provided by a tool manufacturer.
  [Precautions]
  - The plugin software installation must be completed before connecting the device.
  - When detaching/attaching a tool connects to the EE connector, please follow any of these methods shown below, otherwise the device may be damaged.
    1. Connect/disconnect the device following the guidance shown in Manual Tool Change screen.
    2. Connect/disconnect the device executing Tool Detach/Attach icon instruction.
    3. Connect/disconnect the device while the robot controller is turned off.

11.1 COMMON SETTINGS FOR TOOL I/F COMMUNICATION

This section describes the common settings necessary to activate the Tool I/F communication. You can configure the settings using SSET,TLIF_COMMON command in Autocfg.csv. Please refer to chapter 5 “Installation definition file” for details of Autocfg.csv, and section 5.2.6 “SSET command” for details of SSET command. It requires the cycle power to take effect the settings. The following settings are unnecessary if the plugin does not use the Tool I/F communication.

> **⚠ Caution**
> SSET,TLIF_COMMON command described in this section is available on the robot controller software V9.40P/73 or after. If a plugin uses this command, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/72 or older.

Syntax:

```
SSET,TLIF_COMMON,Tool I/F Enable/Disable,Power supply mode,Power1
supply voltage,Power2 supply voltage,Power Voltage for 2A Supply
mode,!!!!
```

Example:

```
SSET,TLIF_COMMON,1,1,2,0,0,!!!!
```

Table 11.1 (a) SSET command parameters for common settings of Tool I/F

Items	Description
Tool I/F Enable/Disable	0: Disables the Tool I/F communication 1: Enables the Tool I/F communication (default)
Power supply mode	Sets the power supply mode. 1: Digital output mode (default) 2: 2A supply mode 3: 2 pin power supply mode
Power1 supply voltage	Sets power supply voltage [V] for Power1 (pin #5). 0: OFF (No power supply) (default) 1: 12V 2: 24V Any one of the above must be specified. If other value is set, it becomes 0 [V].
Power2 supply voltage	Sets power supply voltage [V] for Power2 (pin #7). 0: OFF (No power supply) (default) 1: 12V 2: 24V Any one of the above must be specified. If other value is set, it becomes 0 [V].
Power Voltage for 2A Supply mode	Sets power supply voltage when 2A supply mode is selected for Power supply mode. 0: OFF (No power supply) (default) 1: 12V 2: 24V Any one of the above must be specified. If other value is set, it becomes 0 [V].

Note
When 2A supply mode or 2 pin power supply mode is used, Digital output 1, 2 (RO[1], RO[2]) are not available.

11.2 SETTINGS FOR DIGITAL OUTPUT SIGNAL

This section describes the settings necessary to use Digital output 1,2 (RO[1], RO[2]) available in the Tool I/F communication function. You can configure the settings using SSET,TLIF_DOCFG command in Autocfg.csv. Please refer to chapter 5 “Installation definition file” for details of Autocfg.csv, and section 5.2.6 “SSET command” for details of SSET command. It requires the cycle power to take effect the settings. The following settings are unnecessary if the plugin does not use the Digital Output signals.

⚠ Caution
SSET,TLIF_DOCFG command described in this section is available on the robot controller software V9.40P/73 or after. If a plugin uses this command, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/72 or older.

```
Syntax:
   SSET,TLIF_DOCFG,Digital output 1 mode,Digital output 2
   mode,Digital output voltage,!!!!

Example:
   SSET,TLIF_DOCFG,2,2,2,!!!!
```

Table 11.2 (a) SSET command parameters for Digital output signals of Tool I/F

Parameters	Description
Digital output 1 mode	Specifies the mode for Digital output 1 (RO[1]) (pin #7) when Digital output mode is selected for Power supply mode. 0: Digital output is not used. (default) 1: Sinking 2: Sourcing 3: Push-Pull
Digital output 2 mode	Specifies the mode for Digital output 2 (RO[2]) (pin #6) when Digital output mode is selected for Power supply mode. 0: Digital output is not used. (default) 1: Sinking 2: Sourcing 3: Push-Pull
Digital output voltage	Sets the voltage of Digital output1 (RO[1]) (pin #7) and Digital output2 (RO[2]) (pin #6). 0: OFF (default) 1: 12V 2: 24V

Note
- When “2” (2A supply mode) or “3” (2 pin power supply mode) is specified for the Power supply mode, Digital output 1 (RO[1]) and Digital output 2 (RO[2]) are unavailable.
- When at least either of Digital output mode 1 or 2 is “1” (Sinking) or “3” (Push-Pull), the Digital output voltage cannot be set “1” (12V).
- Possible values for the Digital output voltage depends on the combination of values of Digital output 1, 2 mode. Please see Table 11.2 (c).

Table 11.2 (b) Necessary parameters for each Power supply mode

Pin number	Digital output mode	2A supply mode	2 pin power supply mode
Power1 (pin #5)	Power1 supply voltage	Power Voltage for 2A Supply mode	Power1 supply voltage
Power2 (pin #7)	-	-	Power2 supply voltage
Digital output 1 (RO[1]) (pin #7)	Digital output 1 mode Digital output voltage	-	-
Digital output 2 (RO[2]) (pin #6)	Digital output 2 mode Digital output voltage	-	-

Table 11.2 (c) Possible values for Digital output voltage

		Digital output 2 mode			
		0 (Not used)	**1 (Sinking)**	**2 (Sourcing)**	**3 (Push-Pull)**
Digital output 1 mode	**0 (Not used)**	0 (OFF)	0 (OFF)	1 (12V), 2 (24V)	2 (24V)
	1 (Sinking)	0 (OFF)	0 (OFF)	2 (24V)	2 (24V)
	2 (Sourcing)	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 SETTINGS FOR ANALOG INPUT SIGNAL

This section describes the settings necessary to use Analog input 1, 2 available in the Tool I/F communication function. You can configure the settings using SSET,TLIF_AICFG command in Autocfg.csv. Please refer to chapter 5 “Installation definition file” for details of Autocfg.csv, and section 5.2.6 “SSET command” for details of SSET command. It requires the cycle power to take effect the settings. The following settings are unnecessary if the plugin does not use the Analog Input signals.

⚠ Caution
SSET,TLIF_AICFG command described in this section is available on the robot controller software V9.40P/73 or after. If a plugin uses this command, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/72 or older.

Syntax:
```
SSET,TLIF_AICFG,Purpose of pin #1/#2,Analog input 1 mode,Analog
input 2 mode,!!!!
```

Example:
```
SSET,TLIF_AICFG,2,1,1,!!!!
```

Table 11.3 (a) SSET command parameters for Analog Input signals of Tool I/F

Parameters	Description
Purpose of pin #1 / #2	Selects the use of the pin #1 and #2 on the EE connector (select between Analog Input and RS485 serial communication). 0: OFF 1: RS485 (default) 2: Analog input
Analog input 1 mode	Specifies the mode for Analog input1 (pin #1). 0: OFF (default) 1: 0-10V (Voltage) 2 4-20mA (Current)
Analog input 2 mode	Specifies the mode for Analog input2 (pin #2). 0: OFF (default) 1: 0-10V (Voltage) 2 4-20mA (Current)

Either RS485 Serial Communication or Analog Input is available by setting the above. If Analog input is enabled, the two Analog input signals are automatically assigned as Lack:107 and Slot:1. You can use GET_PORT_ASG built-in function to identify the index number of AI signals. It is also possible to change the index number of AI signals by using SET_PORT_ASG built-in function. The resolution of Analog input hardware is 12 bit.

Note

- When RS485 communication is used, analog input signal is unavailable.
- When using the Analog input, the Purpose of pin #1 / #2 must be set “2” (Analog input), and the corresponding Analog input 1,2 mode must be set “1” (0-10V) or “2” (4-20mA).
- The Analog input signal of the Tool I/F communication function is automatically assigned to AI signal upon power-on, but the AI index number varies depending on the system. If the plug-in uses the Analog input signal of the Tool I/F communication function, make sure to identify the AI index number upon power-on using the GET_PORT_ASG Build-in procedure described later in this section. You can run a KAREL program at every start-up by adding the COLD option to the LOAD command in the Autocfg.csv. See section 5.2.2 "LOAD Command" for details.

11.3.1 GET_PORT_ASG Built-in Procedure

Purpose:

Allows a KAREL program to determine the physical port(s) to which a specified logical port is assigned.

Syntax:

GET_PORT_ASG(*Log_port_type, Log_port_no, Rack_no, Slot_no, Phy_port_type, Phy_port_no, N_ports, Status*)
Input/Output Parameters :
[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

Details:

- *Log_port_type* specifies the code for the type of port whose assignment is being accessed. For AI (Analog Input) signals, this is 3.
- *Log_port_no* specifies the number of the port whose assignment is being accessed. Set the index number of AI signal to determine the physical port.
- *Rack_no* is returned with the rack containing the port module. For Analog Input signals on Tool I/F, this is 107.
- *Slot_no* is returned with the slot containing the port module. For Analog Input signals on Tool I/F, this is 1.
- *Phy_port_type* is returned with the type of port assigned to. Often this will be the same as *Log_port_type*. Exceptions are if *Log_port_type* is a group type (io_gpin or io_gpout) or a port is assigned to memory-image or dummy ports.
- *Phy_port_no* is returned with the number of the port assigned to. For Analog Input1 (pin #1) on Tool I/F, this is 1, and for Analog Input2 (pin #2), this is 2.
- *N_ports* is returned with the number of physical ports assigned to the logical port.
- *Status* is returned with zero if the parameters are valid and the specified port is assigned. Otherwise, it is returned with an error code.

11.3.2 SET_PORT_ASG Built-in Procedure

Purpose:

Allows a KAREL program to assign one or more logical ports to specified physical port(s).

Syntax:

SET_PORT_ASG(*Log_port_type, Log_port_no, Rack_no, Slot_no, Phy_port_type, Phy_port_no, N_ports, Status*)
Input/Output Parameters :
[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

Details:

- *Log_port_type* specifies the code for the type of port to be assigned. For AI (Analog Input) signals, this is 3.
- *Log_port_no* specifies the number of the port to be assigned.
- *Rack_no* is the rack containing the port module. For Analog Input signals on Tool I/F, this is 107.
- *Slot_no* is the slot containing the port module. For Analog Inputs signal on Tool I/F, this is 1.
- *Phy_port_type* is the type of port to be assigned to. Often this will be the same as *Log_port_type*. Exceptions are if *Log_port_type* is a group type (io_gpin or io_gpout) or a port is assigned to memory-image or dummy ports. For AI (Analog Input) signals, this is 3.
- *Phy_port_no* is the number of the port to be assigned to. For Analog Input1 (pin #1) on Tool I/F, this is 1, and for Analog Input2 (pin #2), this is 2.
- *N_ports* is the number of physical ports to be assigned to the logical port.
- *Status* is returned with zero if the parameters are valid. Otherwise, it is returned with an error code. The assignment is invalid if the specified port(s) do not exist or if the assignment of *Log_port_type* to *Phy_port_type* is not permitted. For example, GINs cannot be assigned to DOUTs. Neither *Log_port_type* nor *Phy_port_type* can be a system port type (SOPIN, for example).

Note
The assignment does not take effect until the next power-up.

11.4 SETTINGS FOR SERIAL COMMUNICATION

This section describes the settings necessary to use RS485 Serial Communication available in the Tool I/F communication function. You can define the setting using SSET,TLIF_RS485 command in Autocfg.csv. Please refer to chapter 5 “Installation definition file” for details of Autocfg.csv, and section 5.2.6 “SSET command” for details of SSET command. It requires the cycle power to take effect the settings. The following settings are unnecessary if the plugin does not use the RS485 Serial Communication.

⚠ Caution
SSET,TLIF_RS485 command described in this section is available on the robot controller software V9.40P/73 or after. If a plugin uses this command, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/72 or older.

Syntax:

```
SSET,TLIF_RS485,Purpose of pin #1/#2,Parity,Stop bit,Baud
rate,!!!!
```

Example:

```
SSET,TLIF_RS485,1,0,0,0,!!!!
```

Note
- If you use serial communication, the protocol must always be one response (500 bytes or less) from the tool for one request (492 bytes or less) from the robot controller.
- A response from a tool must be made within 10ms from the start of request.
- The I/O update interval becomes 12ms at the timing of serial communication.

Table 11.4 (a) SSET command parameters for Serial Communication of Tool I/F

Parameters	Description
Purpose of pin #1 / #2	Selects the use of the pin #1 and #2 on the EE connector (select between Analog Input and RS485 serial communication). 0: OFF 1: RS485 (default) 2: Analog input
Parity	Sets the Parity for Serial communication. 0: none (default) 1: odd 2: even
Stop bit	Sets Stop bit for Serial communication. 0:1-bit (default) 1:2-bit
Baud rate	Sets Baud rate for Serial communication. 0:5M (default), 1:2M, 2:1M, 3:115.2k, 4:57.6k, 5:38.4k, 6:19.2k, 7:9.6k [bps]

11.5 KAREL BUILT-IN FOR SERIAL COMMUNICATION

The following KAREL built-in functions are available for RS485 Serial Communication. These built-in functions are available in KAREL program.

If the built-in function of TLIF_WRITE, TLIF_READ, or TLIF_CLEAR_BUFFER is executed on ROBOGUIDE, the status returns 67427 which means “HOST-427 ROBOGUIDE doesn't support KAREL built-in for Tool I/F function”.
If the TLIF_BYTES_AHEAD built-in is executed on ROBOGUIDE, *N_bytes* always returns 0.

The following KAREL built-in functions are supported to be called in multi-tasking.

11.5.1 TLIF_WRITE

Purpose:

Send serial data.

Syntax:

TLIF_WRITE(*Length, Send_data, Status*)
[in] *Length* : INTEGER
[in] *Send_data* : ARRAY OF BYTE
[out] *Status* : INTEGER
%ENVIRONMENT Group :TLIF

Details:

- *Length* specifies the size of send data in bytes. This function can send up to 492 byte.
- *Length* bytes data is sent from the top of *Send_data*.
- *Status* indicates the status of the attempted operation. Non-zero value indicates an error occurred.
- After sending command using TLIF_WRITE once, make sure to read all response data using TLIF_READ written in section 11.5.3 before sending the next command data. It is not allowed to send commands by calling TLIF_WRITE multiple times in a row before reading the response data. Otherwise, communication may be stopped due to timeout and warning may happen.
- This function returns before actually sending.
- The data is sent without fail unless alarm happened or error returned. A function to check whether contents of the send buffer were sent is not provided.

11.5.2 TLIF_BYTES_AHEAD

Purpose:

Check the byte size of serial data received from the device connected via Tool I/F.

Syntax:

TLIF_BYTES_AHEAD(*N_bytes*)
[out] *N_bytes* : INTEGER
%ENVIRONMENT Group :TLIF

Details:

- If *N_bytes* is 0, it did not receive any data. If *N_bytes* is more than 0, *N_bytes* indicates the byte length of received data.

11.5.3 TLIF_READ

Purpose:

Read serial data received from the device connected to the EE connector.

Syntax:

TLIF_READ(*Timeout, Size, Read_data, Status*)

[in] *Timeout* : INTEGER
[in,out] *Size* : INTEGER
[out] *Read_data* : ARRAY OF BYTE
[out] *Status* : INTEGER
%ENVIRONMENT Group :TLIF

Details:

- *Timeout* [ms] specifies the time-out value. If *Timeout* is 0, this function reads size which already received, but equal or less than *Size* bytes. If *Timeout* is set to (-1), this function waits infinitely until receive the data for *Size* bytes. When timeout, received byte size is set to *Size* and the data is set to *Read_data*.
- This function reads *Size* bytes. Read byte is set to *Size*. This function returns if equal or more than *Size* bytes received. This function can receive a response data up to 500 byte.
- The data is read to *Read_data.*
- *Status* indicates the status of the attempted operation. Non-zero value indicates an error occurred.

11.5.4 TLIF_CLEAR_BUFFER

Purpose:

Clear the send buffer and receive buffer.

Syntax:

TLIF_CLEAR_BUFFER(*status*)
[out] *status* : INTEGER
%ENVIRONMENT Group :TLIF

Details:

- In order to prevent receiving the previously received data when running the program in a state where last received data accumulated in the receive buffer when the previous operation has malfunction or stopped halfway especially when developing the program, it is recommended calling this function first.
- If you cannot clear all by receiving data after clear, you can clear by calling this function after waiting for a while after the device sends data.
- *Status* indicates the status of the attempted operation. Non-zero value indicates an error occurred.

11.5.5 TLIF_CRC16

Purpose:

Calculate CRC used by Modbus etc.

Syntax:

TLIF_CRC16(*PuchMsg, UsDataLen, Crc, Status*)
[in] *PuchMsg* : ARRAY OF BYTE
[in] *UsDataLen* : INTEGER
[out] *Crc* : ARRAY OF BYTE
[out] *Status* : INTEGER
%ENVIRONMENT Group :TLIF

Details:

- This function sets CRC of Modbus etc. to *Crc*[1], *Crc*[2] by using *UsDataLen* bytes from the top of *PuchMsg*.
- *Status* indicates the status of the attempted operation. Non-zero value indicates an error occurred.

11.6 SAMPLE PROGRAM OF SERIAL COMMUNICATION

This section shows the program example of serial communication below.
This sample program uses Modbus protocol but it does not necessarily have to be Modbus.

```
PROGRAM IPL_FANUC_SmplGrp_SrlComm

%NOLOCKGROUP
%ENVIRONMENT TLIF
%ENVIRONMENT PC
%INCLUDE        etdccopt
%INCLUDE        klevccdf

VAR
   stat             : INTEGER
   send_data        : ARRAY[255] OF BYTE – Up to 492 bytes
   receive_data : ARRAY[255] OF BYTE – Up to 500 bytes
   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


------------------------------------------------------------
---------------- MAIN FUNCTION ---------------
------------------------------------------------------------
BEGIN
   -- TLIF_CLEAR_BUFFER prevents the program from receiving the previously received data
   -- when running the program while previous data still remains in the receive buffer due to that,
   -- for example, the last execution was paused before reading the data in buffer.
   TLIF_CLEAR_BUFFER(stat)
   IF stat <> 0 THEN
   POST_ERR(stat, '', 0, 2)
      ABORT
   ENDIF

   send_data[1] =    1 -- slave address
   send_data[2] = 16 -- function_code
   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 receive CRC ERROR ', CR)
   --                1234567890123456789212345678931234 5
   ABORT
ENDIF

IF receive_data[1] <> 1 THEN
   WRITE TPERROR (CHR(cc_clear_win)) -- Clear TP error window
   WRITE TPERROR('MODBUS receive address ERROR ', CR)
   --                1234567890123456789212345678931234 5
   ABORT
ENDIF

IF receive_data[2] <> 16 THEN
   WRITE TPERROR (CHR(cc_clear_win)) -- Clear TP error window
   WRITE TPERROR('MODBUS receive function code ERROR ', CR)
   --                1234567890123456789212345678931234 5
   ABORT
ENDIF

start_add = receive_data[3] * 256 + receive_data[4]
IF start_add <> 10000 THEN
   WRITE TPERROR (CHR(cc_clear_win)) -- Clear TP error window
   WRITE TPERROR('MODBUS receive start address ERROR ', CR)
   --                1234567890123456789212345678931234 5
   ABORT
ENDIF
```

If you do not use the TLIF_BYTES_AHEAD built-in, this part does not need to be implemented.

```
   register_num = receive_data[5] * 256 + receive_data[6]
   IF register_num <> 4 THEN
      WRITE TPERROR (CHR(cc_clear_win)) -- Clear TP error window
      WRITE TPERROR('MODBUS receive register num ERROR ', CR)
      ABORT
   ENDIF

   WRITE TPERROR (CHR(cc_clear_win)) -- Clear TP error window
   WRITE TPERROR('SMAPLE program SUCCESS', 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 ALARM LIST

HOST-410 TLIF_READ timeout

Cause: The execution of TLIF_READ built-in function for the Tool I/F function is timed out.

Remedy: This warning is just for information. Data received until timeout is read and the size is returned to the 2nd argument of TLIF_READ. You may have to make the timeout value specified in the 1st argument of TLIF_READ greater.

HOST-411 TLIF_READ request size error

Cause: The size requested in TLIF_READ built-in function for the Tool I/F function is greater than 500 byte.

Remedy: Decrease the request size, which is 2nd argument of TLIF_READ.

HOST-412 Internal error for Tool I/F function %s

Cause: When %s = 1, this is caused by HOST-425.

Remedy: When %s = 1, do the same remedy as HOST-425.

HOST-413 Receive buffer overflow

Cause: Receive buffer for TLIF_READ built-in function for the Tool I/F function is overflowed.

Remedy: Decrease the size of receive data equal or less than 500 byte.

HOST-414 1st argument of TLIF_WRITE is invalid

Cause: The 1st argument of TLIF_WRITE built-in function for the Tool I/F function is not initialized or less than 1.

Remedy: Fix the 1st argument of TLIF_WRITE that specifies the length of send data.

HOST-415 1st argument of TLIF_WRITE is greater than 2nd argument size

Cause: The 1st argument of TLIF_WRITE built-in function for the Tool I/F function is greater than 2nd argument size.

Remedy: Fix the 1st argument of TLIF_WRITE that specifies the length of send data, or fix the array length of the 2nd argument of TLIF_WRITE that is an array of BYTE for send data.

HOST-416 2nd argument of TLIF_READ is uninit or less than 1

Cause: The 2nd argument of TLIF_READ built-in function for the Tool I/F function is not initialized or less than 1.

Remedy: Fix the 2nd argument of TLIF_READ that specifies the size of read data.

HOST-417 2nd argument of TLIF_READ is greater than 3rd argument size

Cause: The 2nd argument of TLIF_READ built-in function for the Tool I/F function is greater than 3rd argument size.

Remedy: Fix the 2nd argument of TLIF_READ that specifies the size of read data, or fix the array length of the 3rd argument of TLIF_READ that is an array of BYTE where read data is stored.

HOST-418 1st argument of TLIF_READ is uninit or less than -1

Cause: The 1st argument of TLIF_READ built-in function for the Tool I/F function is not initialized or less than -1.

Remedy: Fix the 1st argument of TLIF_READ that specifies the timeout value for reading the response data.

HOST-419 2nd argument of TLIF_CRC16 is illegal

Cause: The 2nd argument of TLIF_CRC16 built-in function for the Tool I/F function is uninitialized, or the value is out of the range 1 to 32767.

Remedy: Fix the 2nd argument of TLIF_CRC16 that specifies the data length.

HOST-420 2nd argument of TLIF_CRC16 is greater than 1st argument size

Cause: The 2nd argument of TLIF_CRC16 built-in function for the Tool I/F function is greater than the array length of the 1st argument.

Remedy: Fix the 2nd argument of TLIF_CRC16 that specifies the data length, or fix the BYTE array length specified in the 1st argument of TLIF_CRC16.

HOST-421 3rd argument size of TLIF_CRC16 is smaller than 2

Cause: The 3rd argument size of TLIF_CRC16 built-in function for the Tool I/F function is smaller than 2.

Remedy: Increase the BYTE array size specified in the 3rd argument of TLIF_CRC16 greater than 1.

HOST-422 2nd argument of TLIF_READ is less than 1

Cause: The 2nd argument of TLIF_READ built-in function for the Tool I/F function is less than 1.

Remedy: Fix the 2nd argument of TLIF_READ that specifies the size to read data.

HOST-423 Send buffer overflow by TLIF_WRITE

Cause: The data buffer for sending data for TLIF_WRITE built-in function for the Tool I/F function is overflowed.

Remedy: The data size to send must be equal or less than 492 bytes. Once data equal or less than 492 bytes has been sent using TLIF_WRITE, make sure to read all response data using TLIF_READ described in section 11.5.3 before sending the next data. It is not allowed to call TLIF_WRITE multiple times before reading a response data using TLIF_READ, otherwise the communication may stop until it times out and a warning occurs.

HOST-424 Tool I/F communication error

Cause: A communication error occurred between the Tool I/F board and the robot controller.

Remedy:
1. If the rated current of the tool connected to the tool I/F exceeds 2A, replace the tool or take an external power supply.
2. Take the measures in case that FUSE11 on the main board is blown as described in section 3.6 “FUSE-BASED TROUBLESHOOTING" in the R-30iB Mini Plus CONTROLLER MAINTENANCE MANUAL (B-84175).

HOST-425 Tool I/F communication error (%d)

Cause: 1. A communication error occurred between the Tool I/F board and the robot controller.
2. No response data received from the tool in 10ms.

Remedy: When the number shown in the alarm message (%d) is 91,

- If there is no serial communication detected between the robot and tools, check the mechanical unit cable between the Tool I/F board and the robot controller, the robot connection cable, and each connector.
- If there is serial communication detected between the robot and tools, in addition to the above, make sure that the tool responds in 10ms and the wiring between the tool and the EE connector, and also check section 3.2 “TROUBLESHOOTING DUE TO FUSE” in the “CRX-10iA, CRC-10iA/L, CRX-20iA/L MECHANICAL UNIT MAINTENANCE MANUAL” (B-84195).

When the number shown in the alarm message (%d) is other than 91, contact FANUC service representative with the alarm message and the number shown in the alarm message.

HOST-426 Tool I/F board major version error. Board = %d, Controller = %d

Cause: The Tool I/F board version is too old.

Remedy: Contact FANUC service representative and tell this alarm message and version number shown in this alarm message.

HOST-427 ROBOGUIDE doesn't support KAREL built-in for Tool I/F

Cause: ROBOGUIDE does not support KAREL built-ins for serial communication of the Tool I/F Communication Function.

Remedy: This is just for notification.

HOST-428 Digital output voltage setting, mode setting error

Cause: When the Digital output voltage is “1 (12V)”, the Digital output 1,2 mode cannot be set to “1” (Sinking) or “3” (Push-Pull) in the Tool I/F communication function.

Remedy: The Digital output 1, 2 mode was automatically changed to “0” (Digital output is not used). Please set the correct value.

12 TOOL CHANGE FUNCTION (CRX series only)

This chapter describes the specifications of Tool Change function lets you detach/attach the tool connected to the EE (EE1) connector of FANUC Robot CRX series without turning off the robot controller, and its necessary implementation in the plug-in software to support this function.

To allow you to detach/attach the device with Tool Change function, it is necessary to set the wait time for detaching in Autocfg.csv besides settings of Tool I/F communication described in chapter 11 “TOOL I/F COMMUNICATION FUNCTION”. You can also set the wait time for attaching as necessary. In addition, you can also implement the settings of multiple devices that use Tool I/F in one Autocfg.csv.

For settings of Tool I/F communication and the wait time for detaching/attaching for the primary device in case the plugin software has settings of one or more devices that use Tool I/F, please refer to the following chapter/section.

- Chapter 11 “Tool I/F Communication Function”
- Section 12.5.2 “Setting wait time for detaching/attaching for primary device”

For settings of Tool I/F communication and the wait time for detaching/attaching for the secondary and subsequent devices in case the plugin software has settings of multiple devices that use Tool I/F, please refer to the following chapter/section.

- Section 12.5.3 “Setting Sub-ID for secondary device”
- Section 12.5.4 “Setting Tool I/F communication for secondary device”
- Section 12.5.5 “Setting wait time for detaching/attaching for secondary device”

⚠ CAUTION

- In case of a tool connects to the EE connector, the Tool I/F configuration must be completed before connecting the device to the EE connector. If you connect the device before the configuration is complete, the device may be damaged due to, for example, incorrect power supply voltage. The various settings of Tool I/F communication specified in the Autocfg.csv are set by the plugin software installation, and taken effect after cycling the power of the robot controller.
- FANUC Robot CRX series does not support hot swapping of the tool connections to the EE connector. For the device connects to the EE connecter, make the following precautions clear in the manual to be provided by a tool manufacturer.
  [Precautions]
  - The plugin software installation must be completed before connecting the device.
  - When detaching/attaching a tool connects to the EE connector, please follow any of these methods shown below, otherwise the device may be damaged.
    1. Connect/disconnect the device following the guidance shown in Manual Tool Change screen.
    2. Connect/disconnect the device executing Tool Detach/Attach icon instruction.
    3. Connect/disconnect the device while the robot controller is turned off.

12.1 MANUAL TOOL CHANGE SCREEN

Manual Tool Change screen lets you manually detach/attach the tool without turning off the robot controller. This screen is available as a standard feature of FANUC Robot CRX series with a Tablet TP.

To show Manual Tool Change screen, tap the menu icon on the Tablet TP, select “Setup” - “Tool I/F”, tap the “Manual Tool Change” tab at the top of the screen.

⚠ CAUTION

- To detach/attach tools using Manual Tool Change screen, the robot controller software must be V9.40P/38 or later, and the “Tool I/F Version” shown at the top right corner of the Manual Tool Change screen must be V1.4 or later. If the “Tool I/F Version” is V1.3 or older, please contact your FANUC representative.
- If the “Detach” or “Attach” button is pressed in Manual Tool Change screen with the “Tool I/F Version” V1.3 or older, “HOST-443 Tool Interface board version doesn't support tool change” alarm occurs.
- If “Enable Tool I/F” is set OFF in Tool I/F Setup screen, Manual Tool Change screen cannot be used. Please set “Enable Tool I/F” to ON and re-start the controller.

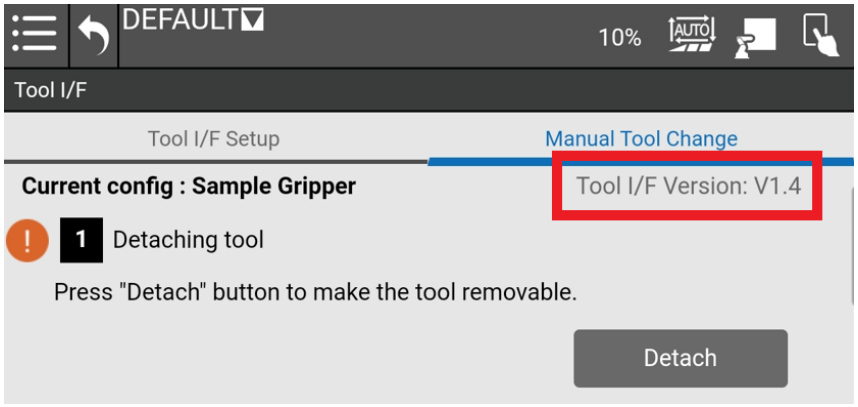


Fig. 12.1 (a) Tool I/F Version

12.1.1 Detaching Tool

Press “Detach” button at the upper part of Manual Tool Change screen to shut off the power on the EE connector, and to allow the user to safely disconnect the device. After the wait time specified in the plug-in software has passed, the button will show “Detached”. Make sure that the button label has been changed to “Detached” before removing the device.

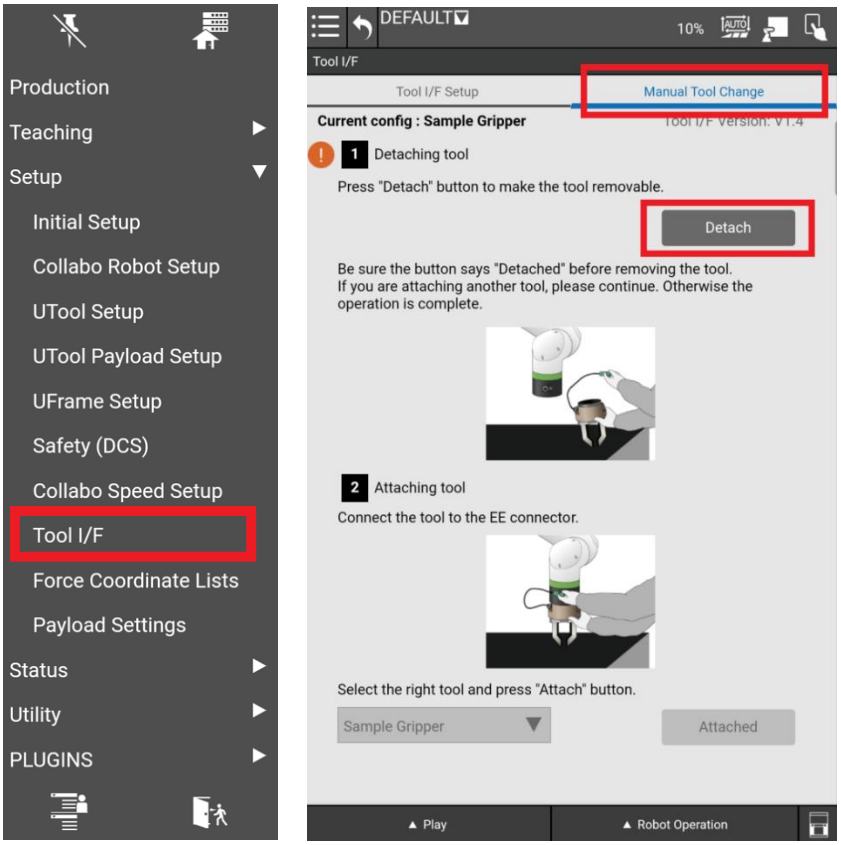


Fig. 12.1.1 (a) Manual Tool Change screen (Detach)

If appropriate wait time is not specified in the plug-in software for the device you are trying to detach, the following pop-up message will appear when “Detach” button is pressed. You may cancel the operation by pressing “CANCEL” button. Or proceed with the operation by pressing “OK”. Be sure to wait enough time according to the specification of the device before removing it.

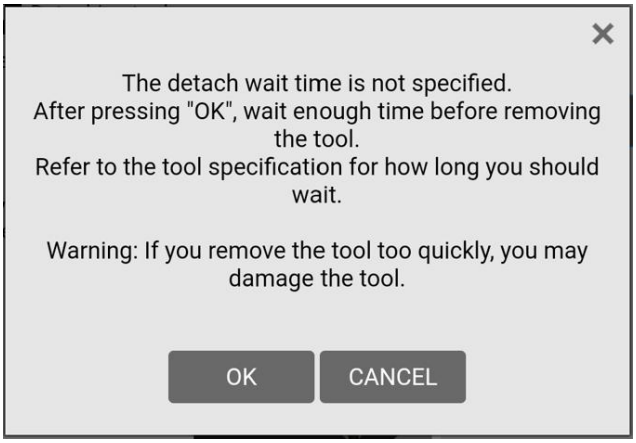


Fig. 12.1.1 (b) Pop-up message shown when “Detach” button is pressed

⚠ CAUTION
If you are using Manual Tool Change screen to detach a device, be sure to wait enough time for the device according to its specifications before removing it. If you disconnect the device prematurely, the device may be damaged.

12.1.2 Attaching Tool

When you attach a device, select the right device from the device list shown at the lower part of Manual Tool Change screen, then press “Attach” button to start supplying power and the communication with the device. After the wait time specified in the plug-in software has passed, the button label will be changed to “Attached”.

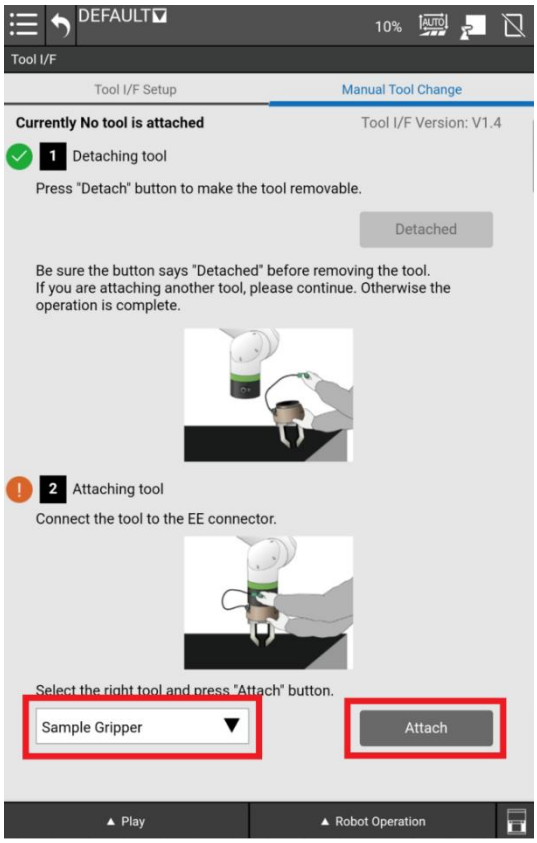


Fig. 12.1.2 (a) Manual Tool Change screen (Attach)

⚠ CAUTION

- The device list in Manual Tool Change screen only shows the devices whose plug-in software have been installed.
- Be sure to select the right device that is actually connected to the EE connector before pressing “Attach” button. If you press “Attach” button for a wrong device selected, the device may be damaged.
- The system is unable to perform Tool change between devices that requires different Power supply voltage setting. The robot controller has to be re-started to attach a device that requires different Power supply voltage setting from the device previously attached.
- The system is unable to perform Tool change between devices that requires different Digital output (RO[1], RO[2]) voltage setting. The robot controller has to be re-started to attach a device that requires different Digital output voltage setting from the device previously attached.

12.2 TOOL DETACH / ATTACH ICON INSTRUCTION

This section describes icon instructions available to detach / attach devices to connect to the EE connector.

- Tool Detach icon instruction
- Tool Attach icon instruction

⚠ CAUTION
- To use the Tool Detach / Attach icon instruction, the robot controller software must be V9.40P/38 or later, and the “Tool I/F Version” shown at the top right corner of Manual Tool Change screen must be V1.4 or later. If the “Tool I/F Version” is V1.3 or older, please contact your FANUC representative.
- If the Tool Detach / Attach icon instruction is executed with the “Tool I/F Version” V1.3 or older, “HOST-443 Tool Interface board version doesn't support tool change” alarm occurs.
- If “Enable Tool I/F” is set OFF in Tool I/F Setup screen, the Tool Detach / Attach icon instruction cannot be used. Please set “Enable Tool I/F” to ON and re-start the controller.

12.2.1 Tool Detach

Tool Detach icon instruction will shut off the power on the EE connector and allow user to safely disconnect the device. You can find this instruction in “Other” category.

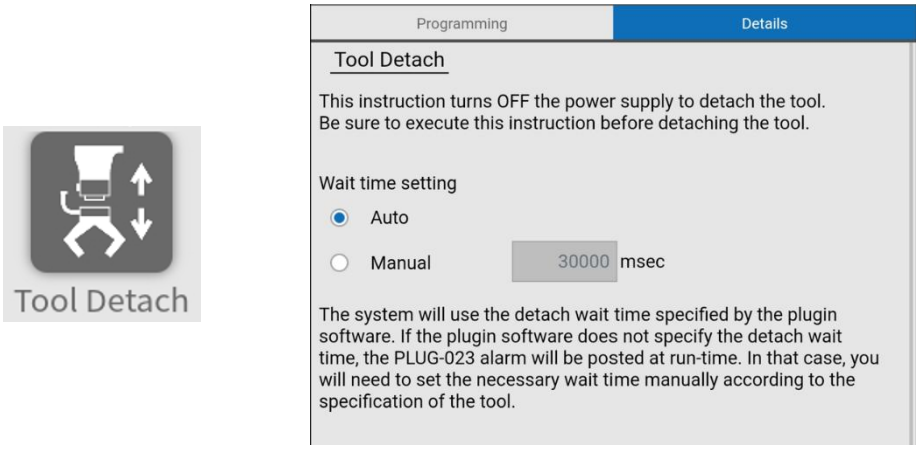


Fig 12.2.1 (a) Tool Detach Icon Instruction

In the detail settings screen, you can specify the wait time to safely detach the device after shutting off the power supply.

If you select “Auto” for the wait time setting, the execution of the instruction completes after the wait time that is specified in the plug-in software. If the wait time is not specified in the plug-in software, “PLUG-023 Plugin software does not have wait time for detaching” alarm is posted when the icon instruction is executed for the device. In that case please manually set the correct time according to the specification of the device.

⚠ CAUTION
When you select “Manual” to set the wait time in Tool Detach icon instruction, be sure to set the right time according to the specification of the device. If you disconnect the device prematurely, the device may be damaged.

12.2.2 Tool Attach

Tool Attach icon instruction will start supplying the power on the EE connector and the communication with the device. You can find this instruction in “Other” category.

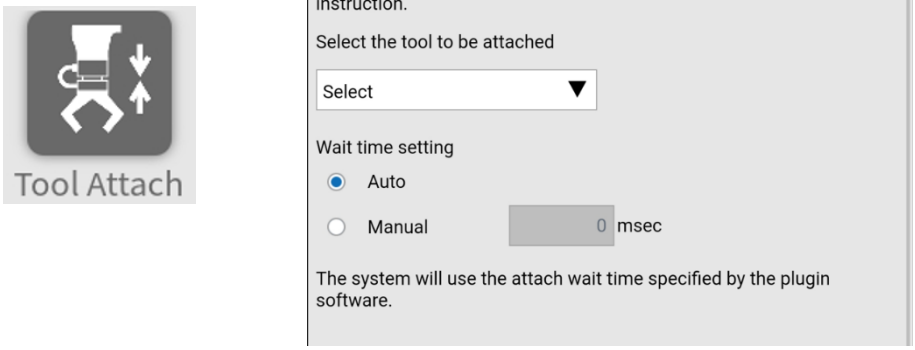


Fig 12.2.2 (a) Tool Attach Icon Instruction

The detail settings screen shows a list of devices whose plug-in software have been installed. Select the right device to be attached from the device list.

Some devices may require some time for its initialization after the power is turned on. The wait time may be specified in the detail settings of Tool Attach icon instruction.

If you select “Auto” for the wait time setting, the execution of the instruction completes after the wait time that is specified in the plug-in software. If you select “Manual”, the instruction waits for the specified wait time.

⚠ CAUTION

- The device list in the detail settings of Tool Attach icon instruction only shows the devices whose plug-in software have been installed.
- Be sure to select the device that is actually connected to the EE connector before executing the Tool Attach instruction. If you execute the Tool Attach instruction for a wrong device selected, the device may be damaged.
- The system is unable to perform Tool change between devices that requires different Power supply voltage setting. The robot controller has to be re-started to attach a device that requires different Power supply voltage setting from the device previously attached.
- The system is unable to perform Tool change between devices that requires different Digital output (RO[1], RO[2]) voltage setting. The robot controller has to be re-started to attach a device that requires different Digital output voltage setting from the device previously attached.

12.3 RESTRICTIONS OF TOOL CHANGE FUNCTION

Tool Change function has the following restrictions.

- To use Tool Change function, the robot controller software must be V9.40P/38 or later, and the “Tool I/F Version” shown at the top right corner of Manual Tool Change screen must be V1.4 or later.

- Tool Change function is only available on Tablet TP with CRX series.
- The device list in the detail settings of Tool Attach icon instruction and Manual Tool Change screen only shows the devices whose plug-in software have been installed.
- If "Enable Tool I/F" is set OFF in Tool I/F Setup screen, Tool Change function cannot be used. Please set "Enable Tool I/F" to ON and re-start the controller.
- The system is unable to perform Tool change between devices that requires different Power supply voltage setting. The robot controller has to be re-started to attach a device that requires different Power supply voltage setting from the device previously attached.
- The system is unable to perform Tool change between devices that requires different Digital output (RO[1], RO[2]) voltage setting. The robot controller has to be re-started to attach a device that requires different Digital output voltage setting from the device previously attached.

12.4 ALARM LIST FOR TOOL CHANGE FUNCTION

PLUG-013 Enable Tool I/F and cycle power

Cause: Tool I/F is disabled when attempting to detach/attach the tool with Manual Tool Change screen or Tool Detach/Attach icon instruction.

Remedy: Enable Tool I/F on Tool I/F Setup screen or Installation Definition file (Autocfg.csv) in the plug-in software to enable Tool I/F and cycle controller power.

PLUG-014 12V cannot be set when Sinking or Push-Pull

Cause: Invalid voltage settings are detected when attempting to install the plug-in software. 12V cannot be set when at least either of Digital output mode 1 or 2 is "1" (Sinking) or "3" (Push-Pull).

Remedy: Set 0V or 24V as Digital output voltage when the Digital output mode 1,2 is "1" (Sinking) or "3" (Push-Pull).

PLUG-015 Cycle power to change Tool I/F voltage

Cause: The voltage setting is different from the previous tool when attempting to attach the tool with Manual Tool Change screen or Tool Attach icon instruction. The robot controller has to be re-started to attach a device that requires different voltage setting from the device previously attached.

Remedy: Cycle power if the different voltage setting needs to be applied.

PLUG-016 1st argument is invalid (Tool Change)
PLUG-017 2nd argument is invalid (Tool Change)
PLUG-018 3rd argument is invalid (Tool Change)

Cause: The argument of Tool Detach/Attach icon instruction is invalid.

Remedy: Check arguments regarding Tool Detach/Attach program or functions.

PLUG-019 Detaching or Attaching is in process

Cause: Previous detaching or attaching process has not completed yet.

Remedy: Wait until completing the process.

PLUG-020 Execute Tool Detach instruction first

Cause: The tool needs to be detached before attaching another tool.

Remedy: Detach the tool before attaching another tool.

PLUG-021 No Tool I/F

Cause: IPL_START_SERIALCOMM function was executed with the robot model that Tool I/F is unavailable.

Remedy: IPL_START_SERIALCOMM function cannot be used with the robot model that Tool I/F is unavailable. Check the KAREL program or use robot which has Tool I/F.

PLUG-022 Select tool to attach

Cause: The selected device is invalid.

Remedy: Select the correct tool.

PLUG-023 Plugin software does not have wait time for detaching

Cause: Tool Detach icon instruction was executed for the device that does not have wait time for detaching.

Remedy: Set proper wait time in Installation Definition file (Autocfg.csv) in the plug-in software, or manually set proper wait time in the details screen of Tool Detach icon instruction according to the specification of the device.

PLUG-024 Failed to initialize Tool Change function

Cause: The initialization process of Tool Change function failed. In case the plugin software has settings of multiple devices that use Tool I/F, PLGSUB command for the secondary and subsequent devices may have not been specified in the Installation Definition file (Autocfg.csv).

Remedy: Verify the PLGSUB command for the secondary and subsequent devices is specified correctly in the Installation Definition file (Autocfg.csv).

12.5 INFORMATION FOR PLUGIN SOFTWARE DEVELOPERS

This section explains SSET command to configure Tool I/F communication settings and the wait time for detaching/attaching in Autocfg.csv for the devices that use Tool I/F. Please refer to chapter 5 “Installation definition file” for details of Autocfg.csv, and section 5.2.6 “SSET command” for details of SSET command. It requires the cycle power to take effect the settings in Autocfg.csv.

> ⚠ **Caution**
> If one plugin software has multiple devices that use the Tool I/F communication and at least one of the devices uses the analog input signal on the Tool I/F, the device that uses the analog input signal must be configured as a primary device.

12.5.1 Setting Tool I/F Communication for Primary Device

For settings of Tool I/F communication for the primary device in case the plugin software has settings of one or more devices that use Tool I/F, please configure using SSET command described in chapter 11 “Tool I/F Communication Function”.

12.5.2 Setting Wait Time for Detaching / Attaching for Primary Device

This section explains SSET,TLIF_DTCHTM and SSET,TLIF_ATCHTM command in Autocfg.csv to set the wait time for the primary device in case the plugin software has settings of one or more devices that use Tool I/F. Please refer to chapter 5 “Installation definition file” for details of Autocfg.csv, and section 5.2.6 “SSET command” for details of SSET command. It requires the cycle power to take effect the settings. The following settings are unnecessary if the device does not use the Tool I/F communication.

> ⚠ **Caution**
> SSET,TLIF_DTCHTM command and SSET,TLIF_ATCHTM command described in this section are available on the robot controller software V9.40P/38 or after. If a plugin uses this command, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/37 or older.

1. TLIF_DTCHTM: Sets the wait time [ms] to allow user to safely disconnect the device after shutting off the power on the EE connector for the primary device in case the plugin software has settings of one or more devices that use Tool I/F.

```
Syntax:
   SSET,TLIF_DTCHTM,Wait time for detaching[ms],!!!!

Example:
   SSET,TLIF_DTCHTM,1500,!!!!
```

⚠ CAUTION
- SSET,TLIF_DTCHTM sets the wait time to wait for the completion of discharging capacitor equipped on the device and safely detach the device after shutting off the power on the EE connector. If you disconnect the device before the time expires, the device may be damaged.
- If the wait time for detaching is not specified in Autocfg.csv, a pop-up message when pressed "Detach" button in Manual Tool Change screen (see section 12.1.1), and PLUG-023 alarm when executing Tool Detach icon instruction (see section 12.2.1) will be shown respectively. Please make sure to set the wait time for detaching in Autocfg.csv so that user can safely detach the device.

2. TLIF_ATCHTM: Sets the wait time [ms] to wait until it turns ready after the power is turned on for the primary device in case the plugin software has settings of one or more devices that use Tool I/F.

```
Syntax:
    SSET,TLIF_ATCHTM,Wait time for attaching[ms],!!!!

Example:
    SSET,TLIF_ATCHTM,800,!!!!
```

It is optional to set the wait time for attaching, but some devices may require some time for its initialization after the power is turned on. You can use SSET,TLIF_ATCHTM command for this purpose.

12.5.3 Setting Sub-ID for Secondary Devices

A unique Sub-ID must be specified for the secondary and subsequent devices respectively in case the plugin software has settings of multiple devices that use Tool I/F. This section explains PLGSUB command to set Sub-ID for the secondary and subsequent devices.

⚠ Caution
PLGSUB command described in this section is available on the robot controller software V9.40P/38 or after. If a plugin uses this command, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/37 or older.

```
Syntax:
    PLGSUB,Sub-ID,Name of Secondary device,!!!!

Example:
    PLGSUB,FANUC_SmplGrp2,FANUC Device,!!!!
```

Sub-ID	Identifies the Plugin from other devices. This ID must be unique. This ID must be "(Company Name)_(Device Name)" to make it unique. Alphanumeric and underscore "_" are allowed and up to 16 char or less. Ex) `FANUC_SmplGrp2` **`1234567890123456`**
Name of Secondary device	Specifies the name of secondary and subsequent devices that use Tool I/F. The name specified here will be shown in the device list next

	to “Attach” button of Manual Tool Change screen (see section 12.1.2) and the device list in the details screen of Tool Attach icon instruction (see section 12.2.2).

Note
- In case the plugin software has settings of multiple devices that use Tool I/F, Sub-ID and device name must be specified for the secondary and subsequent devices respectively using PLGSUB command described in this section. If the Autocfg.csv has settings of Tool I/F communication for the secondary and subsequent devices but Sub-ID and device name are missing, the initialization process of Tool Change function fails at start-up and “PLUG-024 Failed to initialize Tool Change function” appears.
- Sub-ID specified with PLGSUB command must be “(Company Name)_(Device Name)” as with Plugin ID for the primary device.

PLGSUB command may be also specified in plginfo**.dat file to switch languages of the name of secondary and subsequence devices according to the language setting, as well as PLGINFO command. For details of plginfo**.dat and PLGINFO command, please refer to section 5.1.2 “Multi-Language Support for PLGINFO/PLGSUB Command”.

Note
- PLGSUB command may be specified in plginfo**.dat file only if the plugin software has two or more devices that use Tool I/F.
- The first line of plginfo**.dat must always be PLGINFO command. PLGSUB command may be specified after the line of PLGINFO command.
- No blank line is allowed between PLGINFO and PLGSUB command.

12.5.4 Setting Tool I/F Communication for Secondary Devices

This section explains SSET command to configure Tool I/F communication settings for the secondary and subsequent devices in case the plugin software has settings of two or more devices that use Tool I/F. The command syntax is basically the same as the commands described in chapter 11 “Tool I/F Communication Function”, but the Sub-ID must be specified at the top of the commands to identify the device to setup.

⚠ Caution
SSET commands described in this section are available on the robot controller software V9.40P/73 or after. If a plugin uses this command, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/72 or older.

Note
- The Sub-ID specified with the SSET command described in this section must be “(Company Name)_(Device Name)” as with Plugin ID for the primary device. The Sub-ID must be unique.
- The Sub-ID specified with the SSET command described in this section must be the same as the ID specified with the PLGSUB command, which is described in the previous section.

1. TLIF_SUB_COMMON: Sets the common settings necessary to activate the Tool I/F communication for the secondary and subsequent device in case the plugin software has settings of two or more devices that use Tool I/F. For details of each parameter in SSET,TLIF_SUB_COMMON command, please refer to section 11.1 “Common Settings for Tool I/F Communication”.

   ```
   Syntax:
      SSET,TLIF_SUB_COMMON,Sub-ID,Tool I/F Enable/Disable,Power
      supply mode,Power1 supply voltage,Power2 supply voltage,Power
      Voltage for 2A Supply mode,!!!!

   Example:
      SSET,TLIF_SUB_COMMON,FANUC_SmplGrp2,1,1,2,0,0,!!!!
   ```

2. TLIF_SUB_DOCFG: Sets the settings necessary to use Digital output 1,2 (RO[1], RO[2]) available in the Tool I/F communication function for the secondary and subsequent device in case the plugin software has settings of two or more devices that use Tool I/F. For details of each parameter in SSET,TLIF_SUB_ DOCFG command, please refer to section 11.2 “Settings for Digital Output Signal”.

   ```
   Syntax:
      SSET,TLIF_SUB_DOCFG,Sub-ID,Digital output 1 mode,Digital
      output 2 mode,Digital output voltage,!!!!

   Example:
      SSET,TLIF_SUB_DOCFG,FANUC_SmplGrp2,2,2,2,!!!!
   ```

3. TLIF_SUB_AICFG: Sets the settings necessary to use Analog input 1,2 available in the Tool I/F communication function for the secondary and subsequent device in case the plugin software has settings of two or more devices that use Tool I/F. For details of each parameter in SSET,TLIF_SUB_AICFG command, please refer to section 11.3 “Settings for Analog Input Signal”.

   ```
   Syntax:
      SSET,TLIF_SUB_AICFG,Sub-ID,Purpose of pin #1/#2,Analog input 1
      mode,Analog input 2 mode,!!!!

   Example:
      SSET,TLIF_SUB_AICFG,FANUC_SmplGrp2,2,1,1,!!!!
   ```

4. TLIF_SUB_RS485: Sets the settings necessary to use RS485 Serial Communication available in the Tool I/F communication function for the secondary and subsequent device in case the plugin software has settings of two or more devices that use Tool I/F. For details of each parameter in SSET,TLIF_SUB_RS485 command, please refer to section 11.4 “Settings for Serial Communication”.

   ```
   Syntax:
      SSET,TLIF_SUB_RS485,Sub-ID,Purpose of pin #1/#2,Parity,Stop
      bit,Baud rate,!!!!

   Example:
      SSET,TLIF_SUB_RS485,FANUC_SmplGrp2,1,0,0,0,!!!!
   ```

12.5.5 Setting Wait Time for Detaching / Attaching for Secondary Devices

This section explains SSET,TLIF_SUB_DTCHTM and SSET,TLIF_SUB_ATCHTM command in Autocfg.csv to set the wait time for the secondary and subsequent device in case the plugin software has settings of two or more devices that use Tool I/F. The command syntax is basically the same as the commands described in section 12.5.2 “Setting wait time for detaching/attaching for primary device”, but the Sub-ID must be specified at the top of the commands to identify the device to setup.

⚠ Caution
SSET,TLIF_SUB_DTCHTM command and SSET,TLIF_SUB_ATCHTM command described in this section are available on the robot controller software V9.40P/38 or after. If a plugin uses this command, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the plugin software if the current software version is V9.40P/37 or older.

1. TLIF_SUB_DTCHTM: Sets the wait time [ms] to allow user to safely disconnect the device after shutting off the power on the EE connector for the secondary and subsequent device in case the plugin software has settings of two or more devices that use Tool I/F.

```
Syntax:
   SSET,TLIF_SUB_DTCHTM,Sub-ID,Wait time for detaching[ms],!!!!

Example:
   SSET,TLIF_SUB_DTCHTM,FANUC_SmplGrp2,1500,!!!!
```

⚠ CAUTION
- SSET,TLIF_SUB_DTCHTM sets the wait time to wait for the completion of discharging capacitor equipped on the device and safely detach the device after shutting off the power on the EE connector. If you disconnect the device before the time expires, the device may be damaged.
- If the wait time for detaching is not specified in Autocfg.csv, a pop-up message when pressed “Detach” button in Manual Tool Change screen (see section 12.1.1), and PLUG-023 alarm when executing Tool Detach icon instruction (see section 12.2.1) will be shown respectively. Please make sure to set the wait time for detaching in Autocfg.csv so that user can safely detach the device.

2. TLIF_SUB_ATCHTM: Sets the wait time [ms] to wait until it turns ready after the power is turned on for the secondary and subsequent device in case the plugin software has settings of two or more devices that use Tool I/F.

```
Syntax:
   SSET,TLIF_SUB_ATCHTM,Sub-ID,Wait time for attaching[ms],!!!!

Example:
   SSET,TLIF_SUB_ATCHTM,FANUC_SmplGrp2,800,!!!!
```

It is optional to set the wait time for attaching, but some devices may require some time for its initialization after the power is turned on. You can use SSET,TLIF_ATCHTM command for this purpose.

12.5.6 KAREL Functions to Determine Device Disconnection Status

The KAREL functions IPL_START_SERIALCOMM and IPL_END_SERIALCOMM with the function definition indicated below may be called in a KAREL program to determine the device's disconnection status.

Purpose:

Checks whether the device specified with the plug-in ID is electrically connected and in a communication-capable state. If the device is connected and ready to write/read serial data, transition it to an active communication mode and prevent it from being disconnected by the Tool Detach instruction. This state persists until the IPL_END_SERIALCOMM function described next is called.

Syntax:

IPL_START_SERIALCOMM (*Plugin_id*, *Ok_to_start*, *Status*) from IPL_FANUC_SYS_PLGUTIL
Input/Output Parameters :
[in] *Plugin_id* : STRING
[out] *Ok_to_start* : INTEGER
[out] *Status* : INTEGER

Details:

- *Plugin_id* indicates the plug-in ID (Plug-in ID specified with a PLGID command in Autocfg.csv) of the plug-in device to determine the tool's disconnection status.
- *Ok_to_start* returns a value indicating whether or not the device is connected the KAREL program can perform serial communication.
  0: Unable to communicate (disconnected)
  1: Able to communicate (connected)
- *Status* returns the status. 0 (zero) indicates the call was done successfully. Non-zero value indicates an error occurs during the process.

Purpose:

When the device specified with the plug-in ID is connected, transition it to a communication-completed state and enable it to be disconnected by the Tool Detach instruction.

Syntax:

IPL_END_SERIALCOMM (*Plugin_id*, *Status*) from IPL_FANUC_SYS_PLGUTIL
Input/Output Parameters :
[in] *Plugin_id* : STRING
[out] *Status* : INTEGER

Details:

- *Plugin_id* indicates the plug-in ID of the plug-in device to transition it to a communication-completed state to allow it to be disconnected.
- *Status* returns the status. 0 (zero) indicates the call was done successfully. Non-zero value indicates an error occurs during the process.

⚠ CAUTION

- The KAREL functions IPL_START_SERIALCOMM and IPL_END_SERIALCOMM are available on the robot controller software V9.40P/71 or after. If a plugin uses these functions, make it clear in the manual which is provided by a device supplier, that the software version of the robot controller must be upgraded to the latest version prior to installing the Plugin software if the current software version is V9.40P/70 or older.
- Once the IPL_START_SERIALCOMM function is called and the device enters the communicating state, the device cannot be disconnected again until the next IPL_END_SERIALCOMM function is called. Ensure that the IPL_END_SERIALCOMM is always called after the IPL_START_SERIALCOMM function.
- Following the invocation of the IPL_START_SERIALCOMM function, refrain from executing repeated TLIF_WRITE/TLIF_READ built-ins without subsequently invoking the IPL_END_SERIALCOMM function. Executing multiple TLIF_WRITE/TLIF_READ built-ins between IPL_START_SERIALCOMM and IPL_END_SERIALCOMM may lead to increased wait times for the Tool Detach instruction if it is triggered during that period, potentially blocking the TP program from proceeding.

12.5.7 Example of Installation Definition File

The following figure shows an example of an Autocfg.csv file including the secondary and subsequent devices.

```
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
```

Fig 12.5.7 (a) Example of an Autocfg.csv

13 PERIPHERAL DEVICE INTERFACE ON R-30iB MINI PLUS

This chapter describes the specifications of the peripheral device interface on a robot controller R-30*i*B Mini Plus. The following table shows the peripheral device interface of R-30*i*B Mini Plus.

Table 13 (a) Peripheral Device Interface Types

Name	Ordering specification	Number of I/O points		Remarks
		JRM18		
		DI	DO	
Main board A	A05B-2695-H001	20	16 (Source)	Vision I/F, Force sensor I/F, Line tracking I/F, HDI

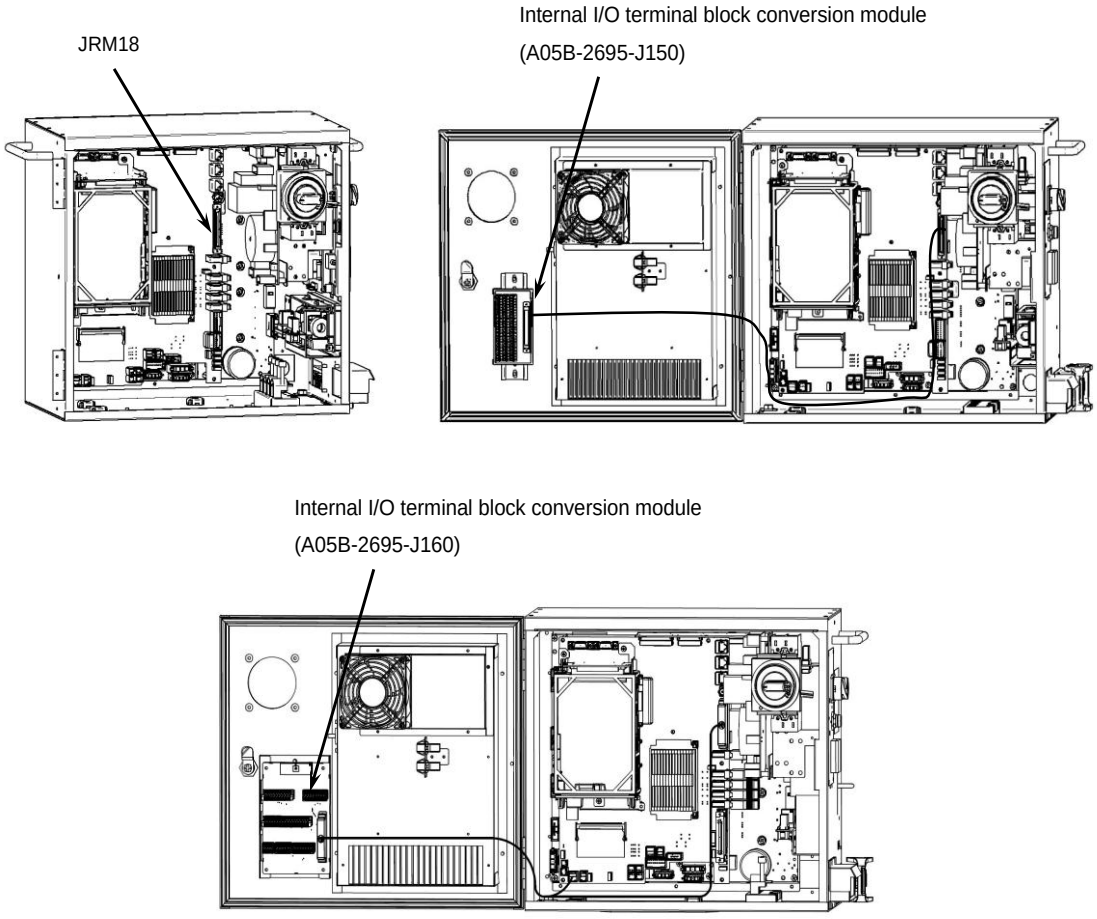


Fig.13 (a) Connecting the peripheral device cable (JRM18)
Upper left: No I/O terminal block conversion module
Upper right: Internal I/O terminal block conversion module (A05B-2695-J150)
Lower: Internal I/O terminal block conversion module (A05B-2695-J160)

> **⚠ Caution**
> R-30iB Mini Plus can be equipped to supply 24V power to an external devices only when the total current including for DI/DO and Safety DI/DO is less than 0.7A, Please refer to “12.5 24V power supply for external” for details. If the total current is 0.7A or more, please take 24V from external power supply.

13.1 PERIPHERAL DEVIDE INTERFACE (JRM18)

There are 20 data inputs (DI) and 16 data outputs (DO) available on peripheral device interface (JRM18) of the main board. The following tables show the DI/DO signals of peripheral device interface.

Table 13.1 (a) Input signals of peripheral device interface

Connector number	Signal name	Standard I/O assignment		Description	Remarks
		UOP auto asg. : Simple (JRM18)	**UOP auto asg. : None Full Full (Slave) Simple Simple (Slave)**		
JRM18-8	DI01	DI[101]	DI[101]	Peripheral device status	General signal
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]	Temporary stop	
JRM18-48	DI14	UI[5] RESET	DI[82]	External reset	
JRM18-15	DI15	UI[6] START	DI[83]	Start	
JRM18-49	DI16	UI[8] ENBL	DI[84]	Operation enabled	
JRM18-16	DI17	UI[9] PNS1	DI[85]	Program number	
JRM18-50	DI18	UI[10] PNS2	DI[86]	Program number	
JRM18-17	DI19	UI[11] PNS3	DI[87]	Program number	
JRM18-51	DI20	UI[12] PNS4	DI[88]	Program number	

Table 13.1 (b) Output signals of peripheral device interface

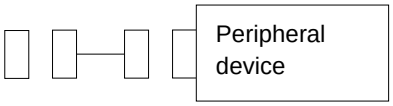
Connector number	Signal name	Standard I/O assignment		Description	Remarks
		UOP auto asg. : Simple (JRM18)	**UOP auto asg. : None Full Full (Slave) Simple Simple (Slave)**		
JRM18-18	DO01	DO[101]	DO[101]	Peripheral device status	General signal
JRM18-52	DO02	DO[102]	DO[102]		
JRM18-19	DO03	DO[103]	DO[103]		
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]	During automatic operation	
JRM18-58	DO14	UO[6] FAULT	DO[82]	Alarm	
JRM18-25	DO15	UO[9] BATALM	DO[83]	Battery voltage drop	
JRM18-59	DO16	UO[10] BUSY	DO[84]	During operation	

13.2 CONNECTION BETWEEN MAIN BOARD (JRM18) AND PERIPHERAL DEVICES

Peripheral device control interface (source type DO)

JRM18

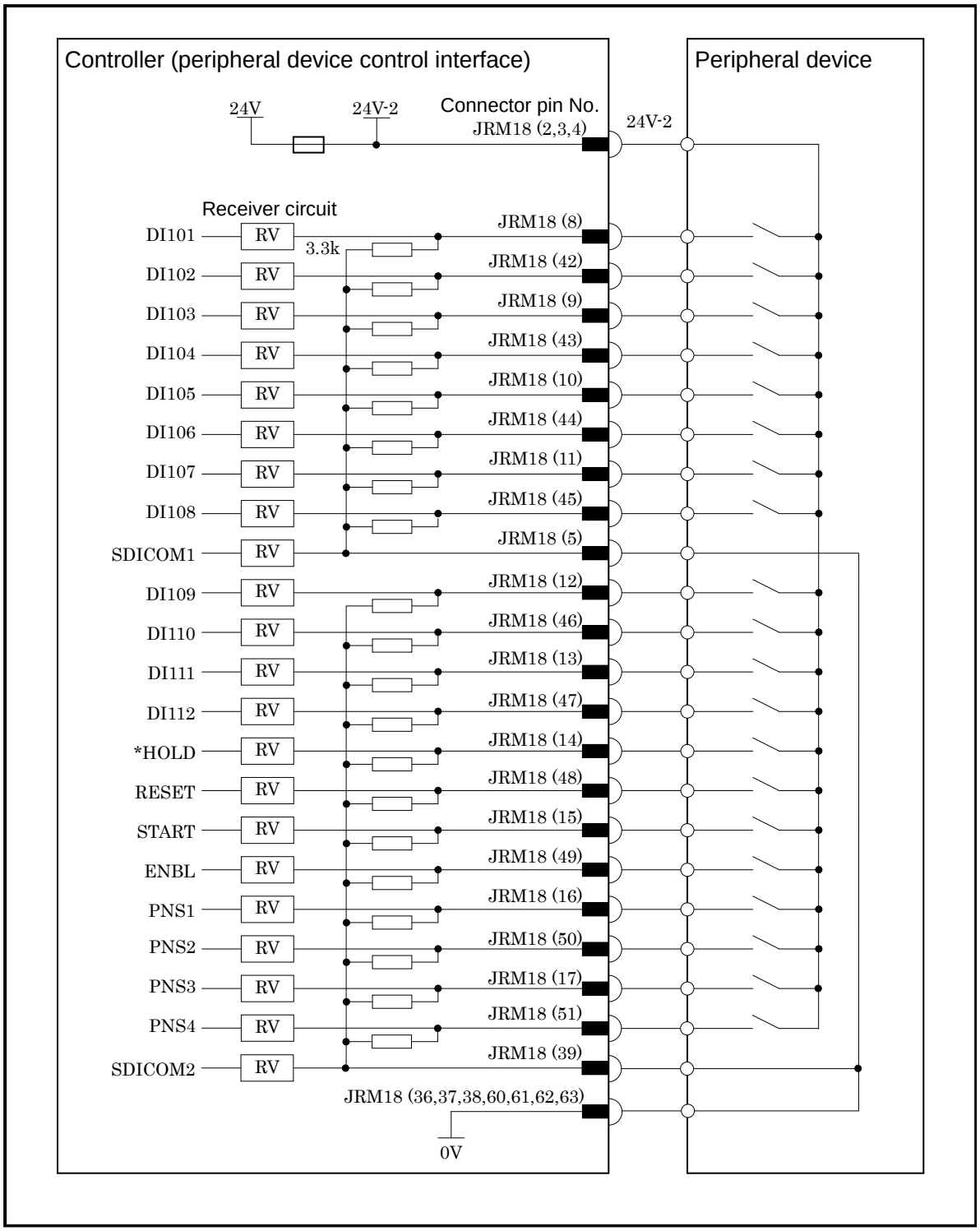
1	24VEXT_IN	35	0VEXT
2	24V-2	36	0V
3	24V-2	37	0V
4	24V-2	38	0V
5	SDICOM1	39	SDICOM2
6	EES1	40	EES2
7	EAS1	41	EAS2
8	DI101	42	DI102
9	DI103	43	DI104
10	DI105	44	DI106
11	DI107	45	DI108
12	DI109	46	DI110
13	DI111	47	DI112
14	*HOLD	48	RESET
15	START	49	ENBL
16	PNS1	50	PNS2
17	PNS3	51	PNS4
18	DO101	52	DO102
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	



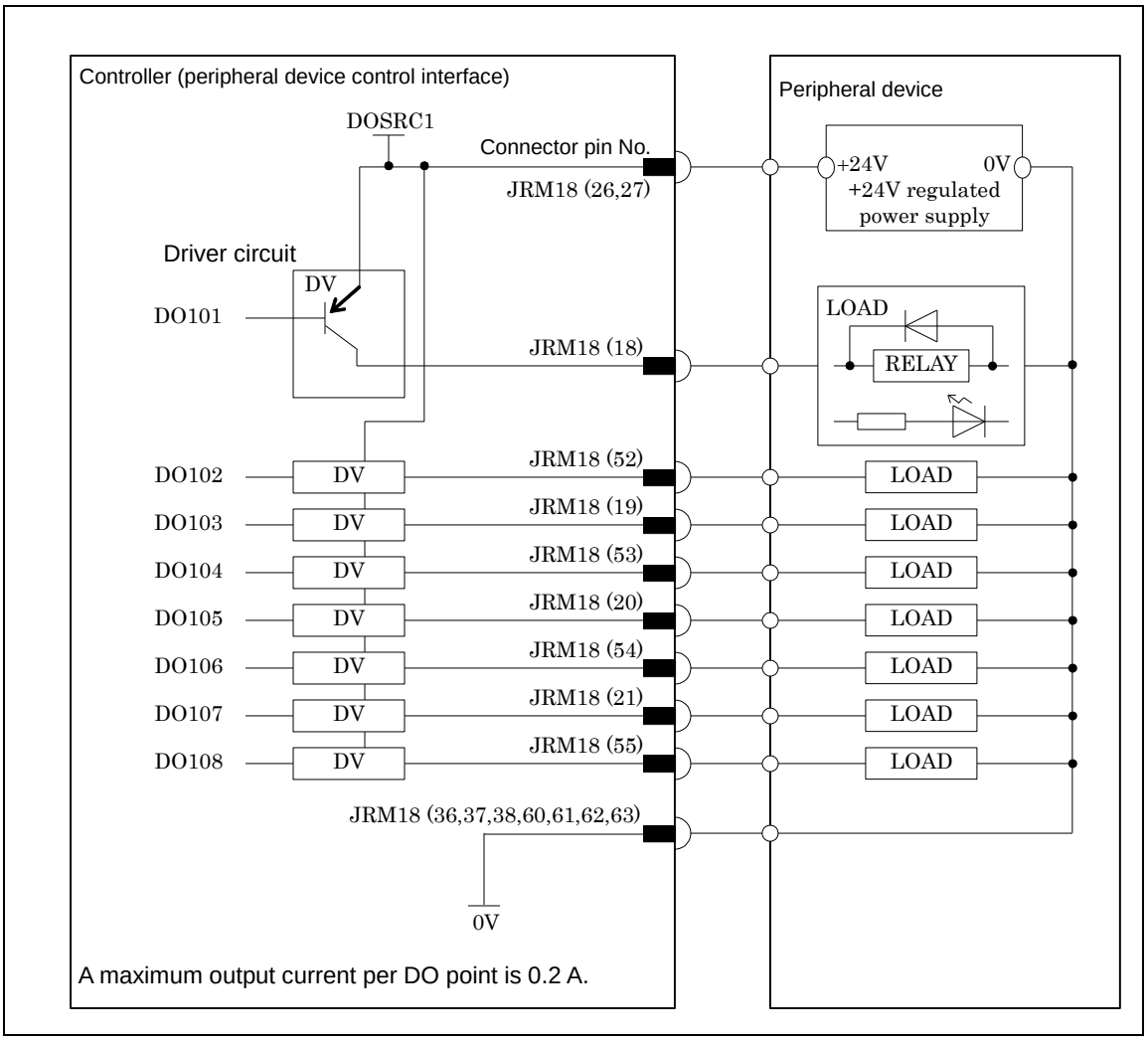
SDICOM1 to 2 signals are common selection signal for SDI.
When 24V-2 common is used, connect to 0V.
When 0V common is used, connect to 24V-2.

SDICOM1: Selects a common for DI101 to DI108.
SDICOM2: Selects a common for DI109 to DI112, *HOLD, RESET, START, ENBL, PNS1 to PNS4.

NOTE
1 The peripheral device connection cables are optional.
2 The DOSRC1 and DOSRC2 pins of the JRM18 are pins for supplying power to drivers. (None of these pins can be left open.)

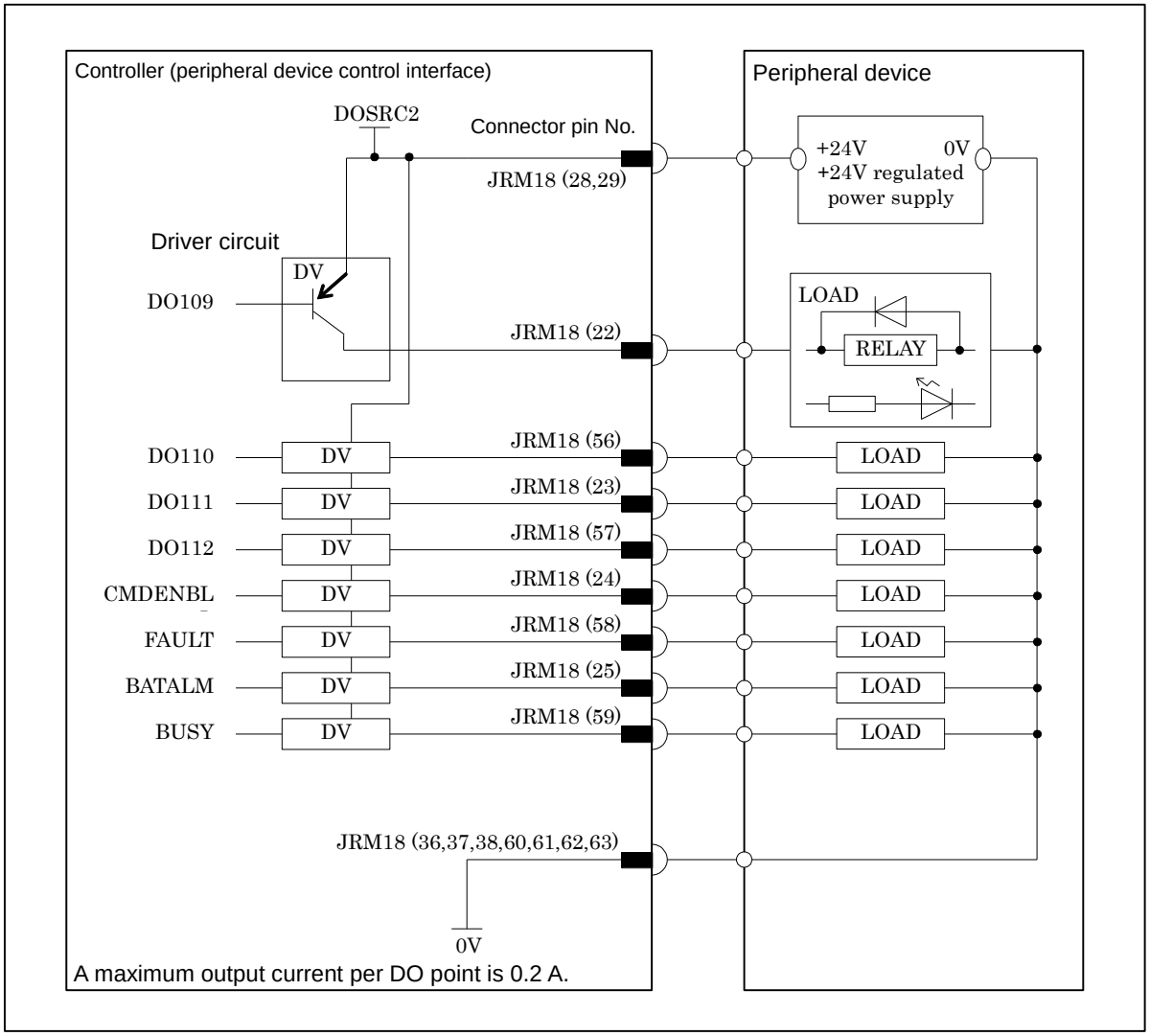


NOTE
In this diagram, common voltage of input devices is +24V.



NOTE

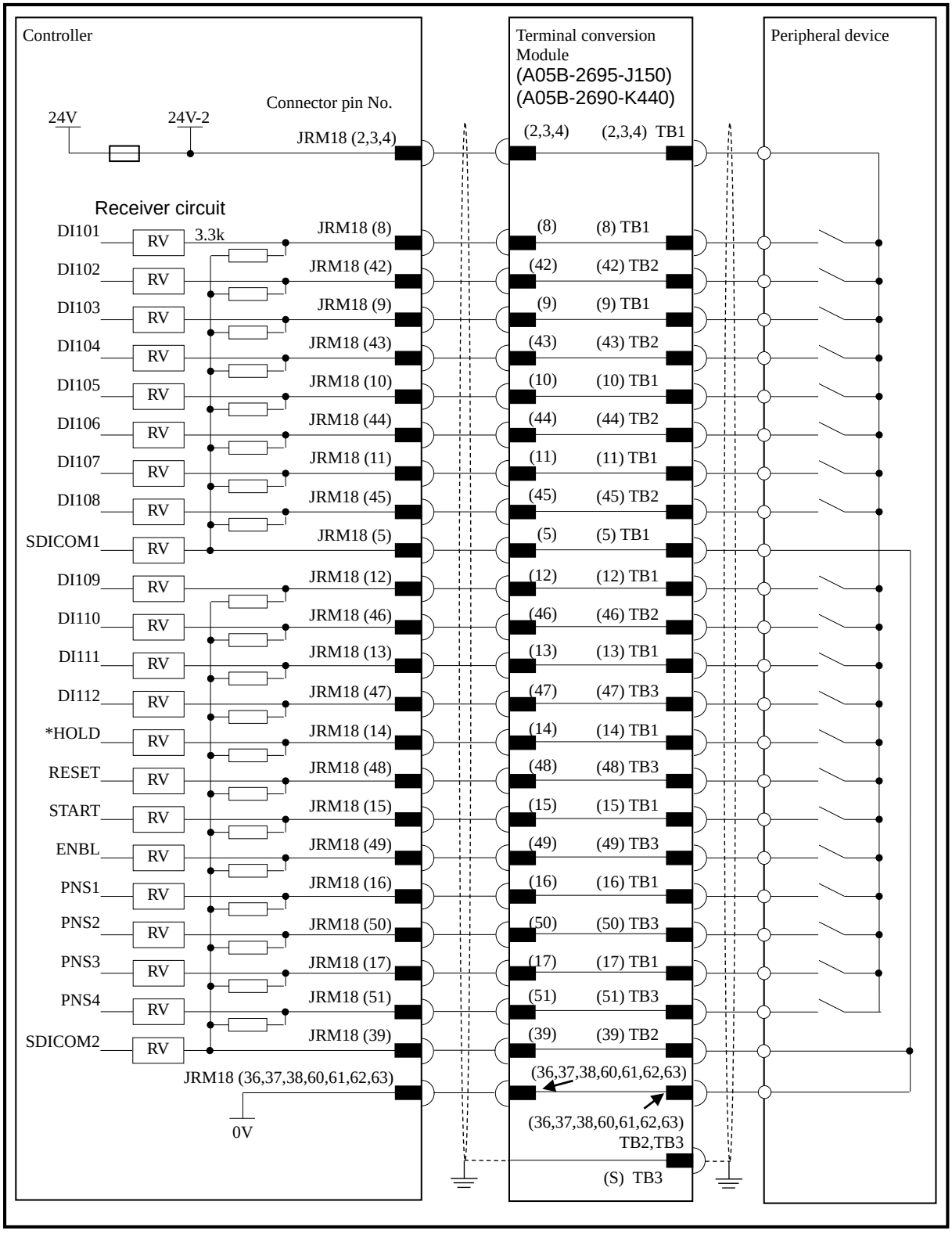
Connector pin No.36-38 and No.60-63 of JRM18 are for 0V. None of these pins can be left open.



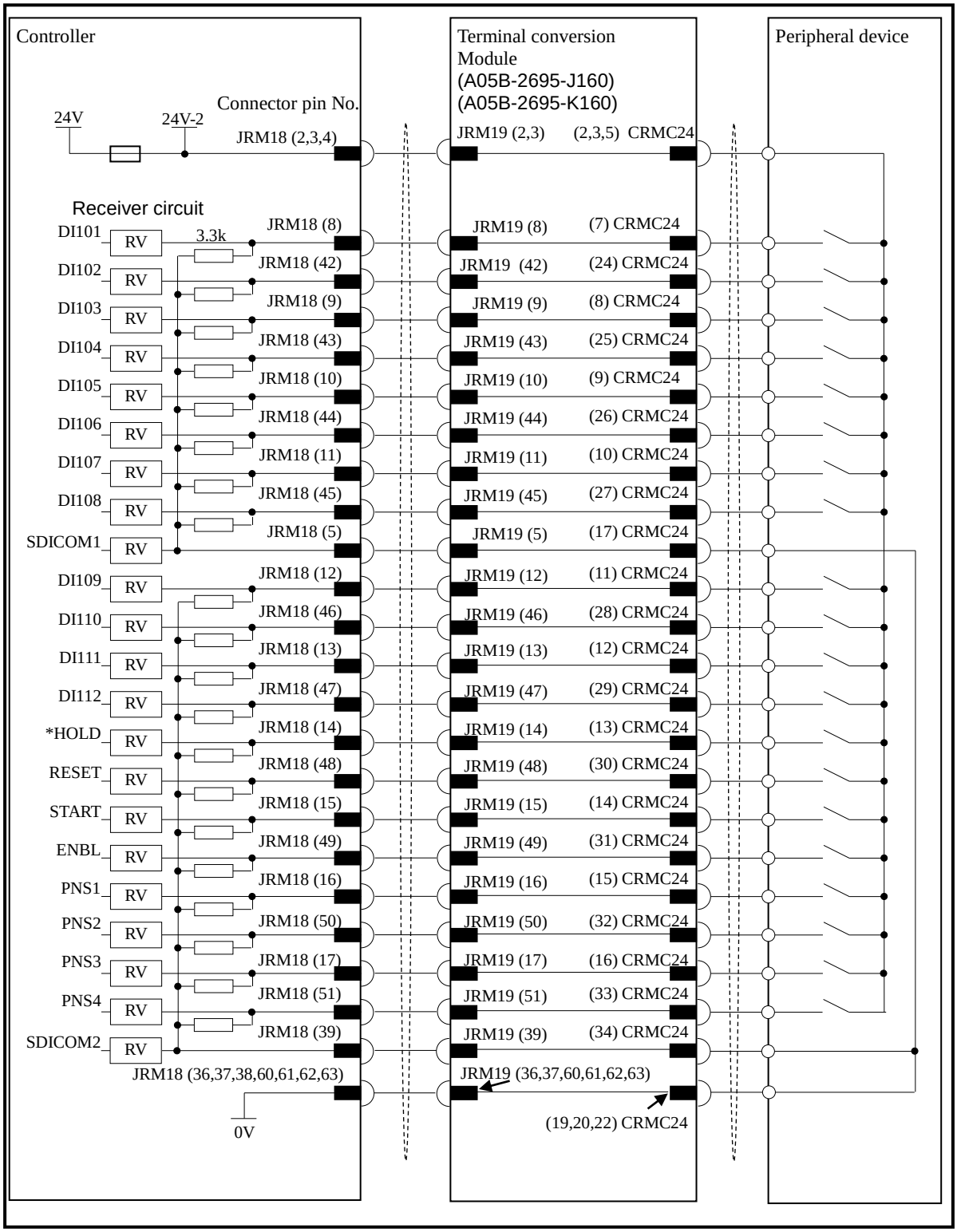
NOTE

Connector pin No.36-38 and No.60-63 of JRM18 are for 0V. None of these pins can be left open.

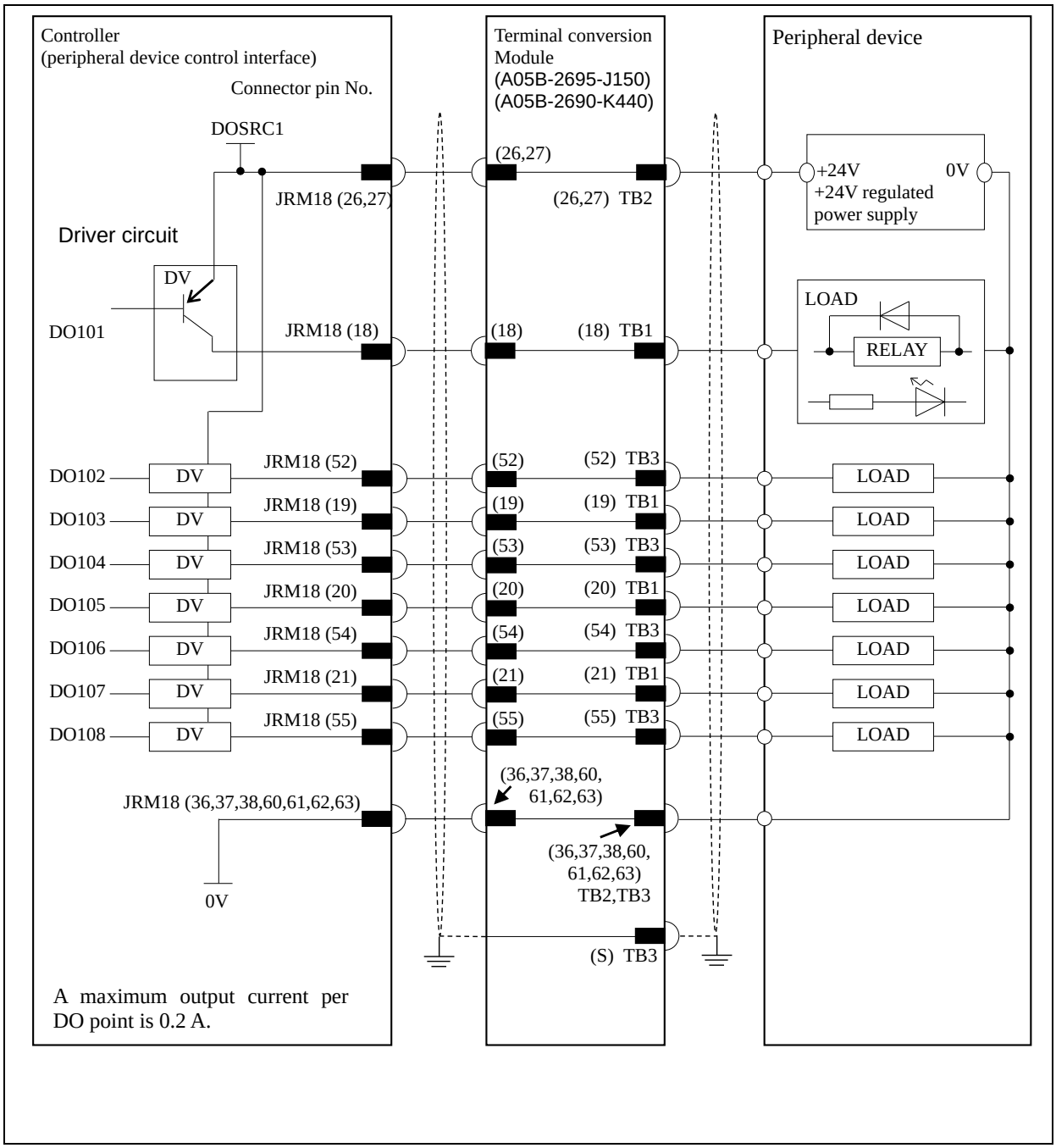
13.3 CONNECTION BETWEEN MAIN BOARD (JRM18) AND PERIPHERAL DEVICES (FOR TERMINAL CONVERSION MODULE)



NOTE
In this diagram, common voltage of input devices is +24V.

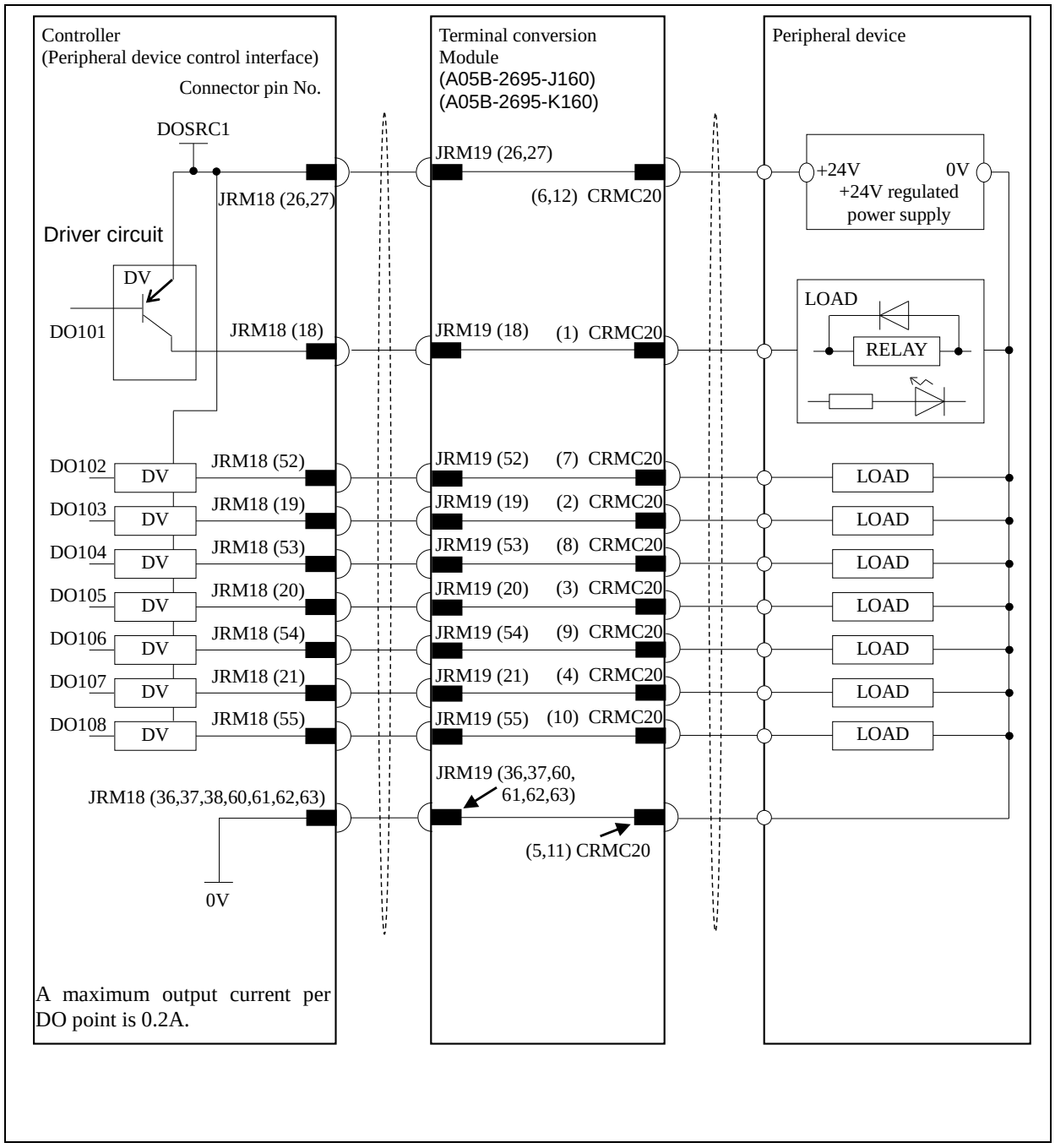


NOTE
In this diagram, common voltage of input devices is +24V.



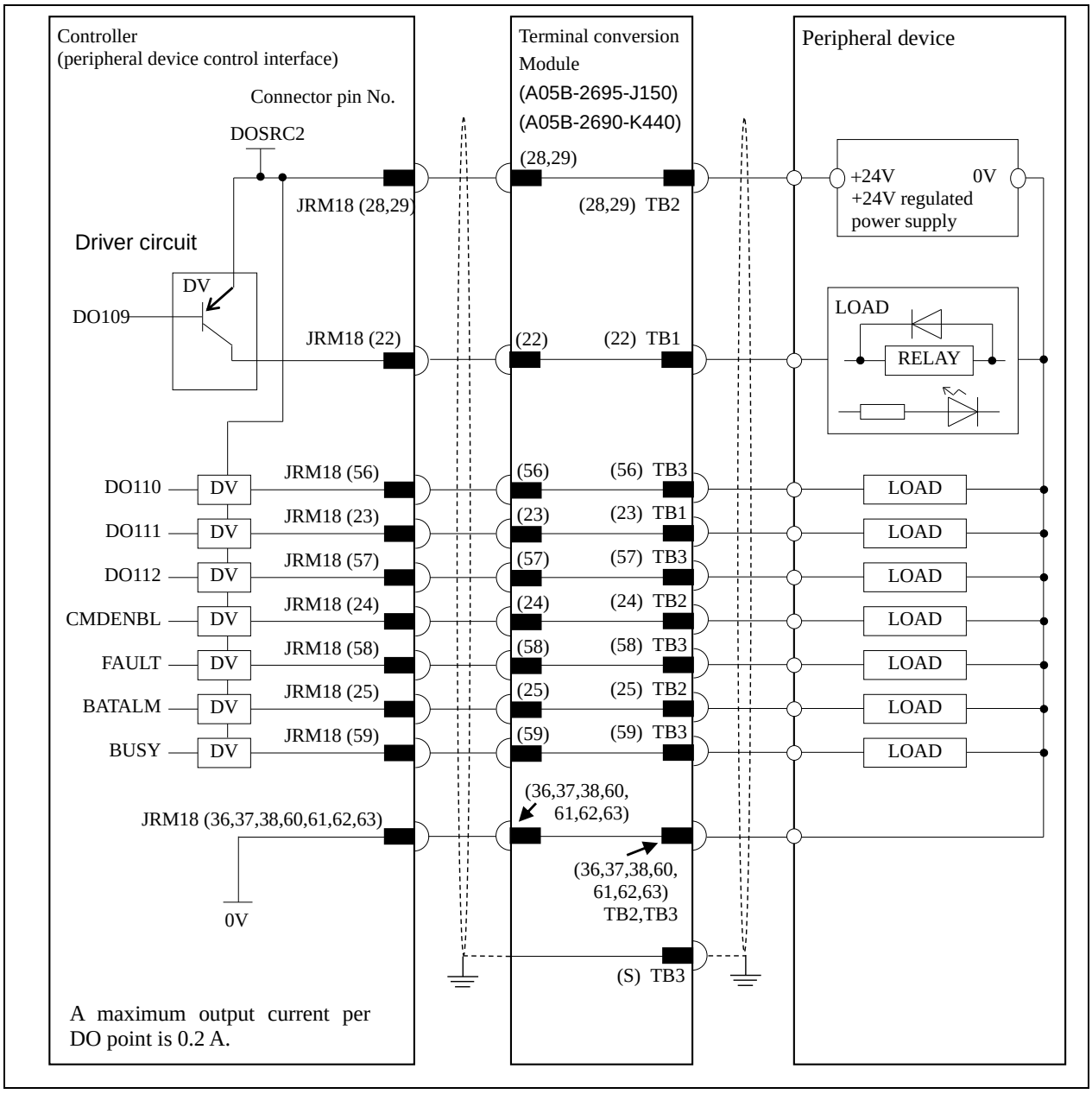
NOTE

Connector pin No.36-38 and No.60-63 of JRM18 are for 0V. None of these pins can be left open.



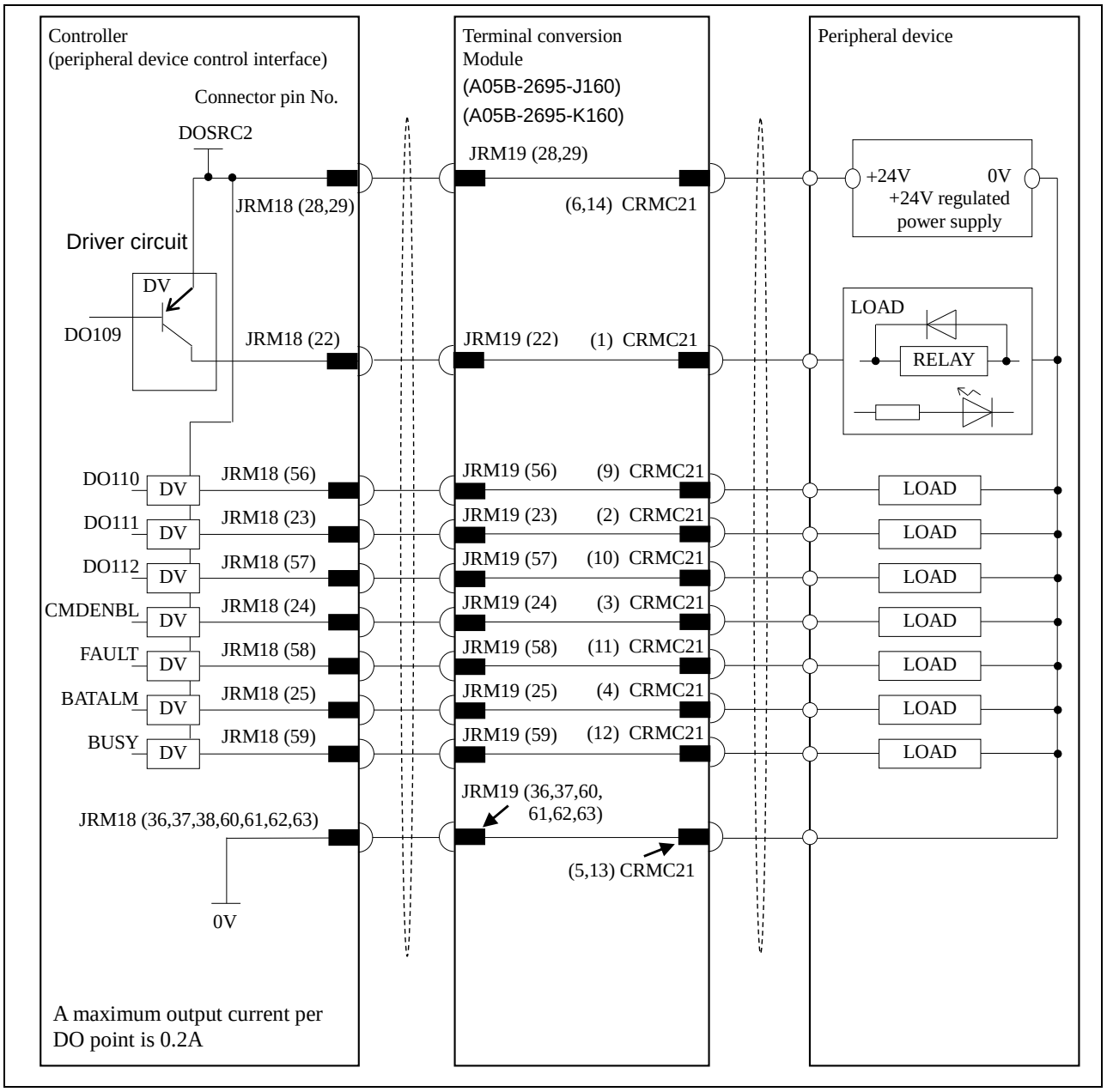
NOTE

Connector pin No.36-38 and No.60-63 of JRM18 are for 0V. None of these pins can be left open.



NOTE

Connector pin No.36-38 and No.60-63 of JRM18 are for 0V. None of these pins can be left open.

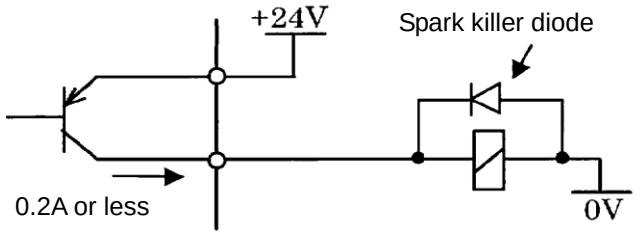


NOTE
Connector pin No.36-38 and No.60-63 of JRM18 are for 0V. None of these pins can be left open.

13.4 DIGITAL I/O SIGNAL SPECIFICATIONS OF PERIPHERAL DEVICES

This section describes the specifications of the digital I/O signals interfaced with the peripheral devices.

(1) Output signals in peripheral device interface (Source type DO)
(a) Example of connection



(b) Electrical specifications

Maximum load current when driver is on:	200mA (including momentary level)
Saturation voltage when driver is on:	1.0V max.
Dielectric strength:	24V ±20% (including momentary level)
Leakage current when driver is off:	100μA

(c) The external power supply to output signals must satisfy the following:

Power supply voltage:	+24V ±10%
Power supply current:	For each printed circuit board of this type (Total sum of maximum load currents including momentary levels + 100mA or more)
Power-on timing:	At the same time when the controller is turned on or earlier
Power-off timing:	At the same time when the controller unit is turned off or later

(d) Spark killer diode

Rated peak reverse voltage:	100V or more
Rated effective forward current:	1A or more

(e) Driver for output signals

In the driver device, the current of each output signal is monitored, and when an overcurrent is detected, the relevant output is turned off. After an output has been turned off by overcurrent, the overcurrent state is released because the output is off, so the output on state is restored. Therefore, in the ground fault or overcurrent state, the output is turned on and off repeatedly. Such a condition is found also when a load with a high surge current is connected.
The driver device also includes an overheat detection circuit, which turns off all outputs of the device when the internal temperature of the device has increased as a result of a continued overcurrent state due to a ground fault of an output and so on. The outputs are held off, but their normal states can be restored by turning the power to the controller on and off after the internal temperature of the device has lowered.

(f) Note on use

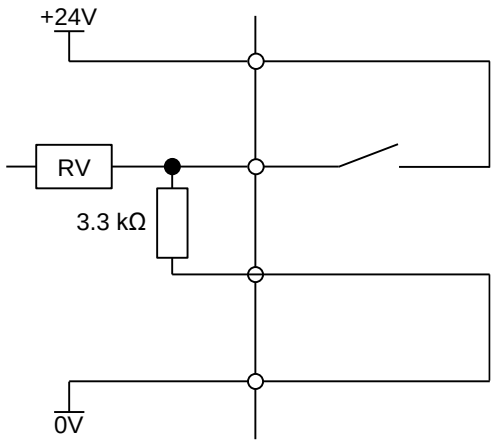
24V power supply of the controller can be used only when the total current including for DI/DO, Safety DI/DO and peripheral devices is less than 0.7A. If the total current is 0.7A or more, please take 24V from external power supply. Please refer to section 13.5 “24V power supply for external” for details.
When adding a relay, solenoid, or the like directly to the circuit, connect a diode for counter electromotive voltage protection in parallel to the load.
When using a load, such as a lamp, that generates surge current when it is turned on, install a protection resistor.

(g) Applicable signals

Output signals of main board I/O board JRM18
CMDENBL, FAULT, BATALM, BUSY, DO101 to DO112

(2) Input signals in peripheral device interface

(a) Example of connection (+24V common)

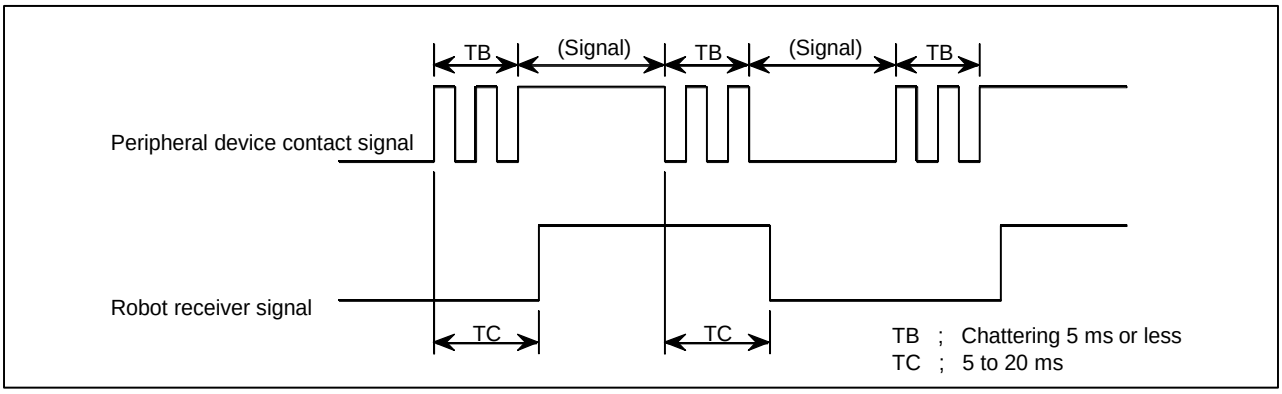


(b) Electrical specifications of the receiver

Type:	Grounded voltage receiver
Rated input voltage:	Contact close: +20V to +28V
	Contact open: 0V to +4V
Maximum applied input voltage:	+28VDC
Input impedance:	3.3kΩ (approx.)
Response time:	5ms to 20ms

(c) Specifications of the peripheral device contact

Voltage and Current:	DC24V, 0.1A
	(Use a contact which minimum load is 5mA or less.)
Input signal width:	200ms or more (on/off)
Chattering time:	5ms or less
Closed circuit resistance:	100Ω or less
Opened circuit resistance:	100kΩ or more



(d) Note on use

Apply the +24 V power at the robot to the receiver.
However, the above signal specifications must be satisfied at the robot receiver.

(e) Applicable signals

Input signals of main board JRM18
XHOLD, RESET, START, ENBL, DNS1 to DNS4, DI101 to DI112

13.5 24V POWER SUPPLY FOR EXTERNAL

24V power supply of the controller can be used only when the total current including for DI/DO, Safety DI/DO and peripheral devices is less than 0.7A.
Depending on the availability of the I/O terminal block conversion module, connection points are changed as followings.

Table 13.5 (a) 24V power supply for external and connection points

Conditions	Connection points
No I/O terminal block conversion module	“24V-2” and “0V” from JRM18 connector on main board
Internal I/O terminal block conversion module (“A05B-2695-J150” or “A05B-2690-K440”)	“24V-2” and “0V” from TB1 on terminal conversion module
Internal I/O terminal block conversion module (“A05B-2695-J160” or “A05B-2695-K160”)	“24V-2” and “0V” from CRMC24 on terminal conversion module

14 ECLIPSE DEVELOPMENT ENVIRONMENT

This chapter describes the setup and operation of the Eclipse development environment provided to assist with the FANUC plugin software development. The Eclipse development environment provided by the plugin software development SDK supports the designation of colors for keywords and comments as well as building and debugging program files. The development environment supports the following.

Table 14 (a) Eclipse development environment support list

Name	Extension	Color designation	Building	Debugging
KAREL	*.kl	Yes	Yes	Yes
iHMI component	*.ishtm	No	Yes	No

⚠ Caution
The Eclipse development environment only supports ROBOGUIDE V9.

Note
The purpose of the Eclipse development environment is to assist with the FANUC plugin software development, although it is possible to develop the plugin software without the Eclipse development environment.

14.1 OPERATING ENVIRONMENT

14.1.1 Development Environment for Eclipse Plugins

The following or a later version of Eclipse IDE is required for a guaranteed development environment.

- Eclipse IDE for Eclipse Committers (Version: 2019-03 (4.11.0))

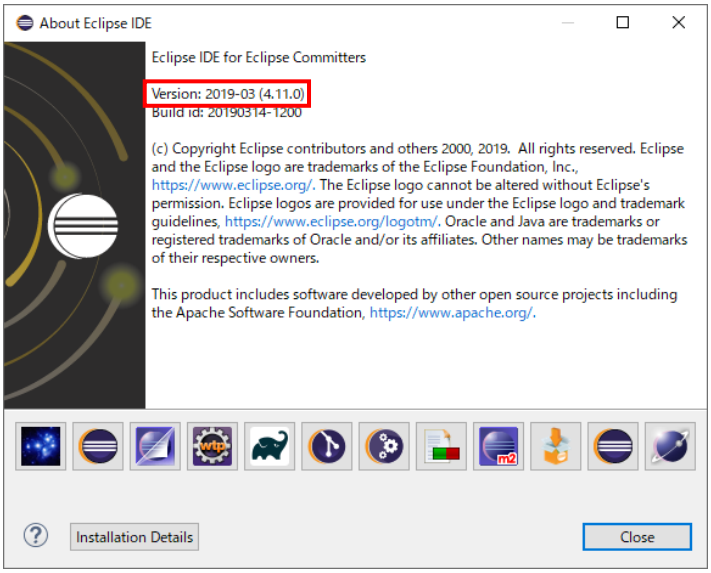


Fig. 14.1.1 (a) Eclipse IDE for Eclipse Committers (Version: 2019-03 (4.11.0))

14.1.2 Operating Environment for Eclipse Plugins

The following or a later version of Eclipse IDE is required for a guaranteed operating environment.

- Eclipse IDE for Enterprise Java Developers (Version: 2019-03 (4.11.0))

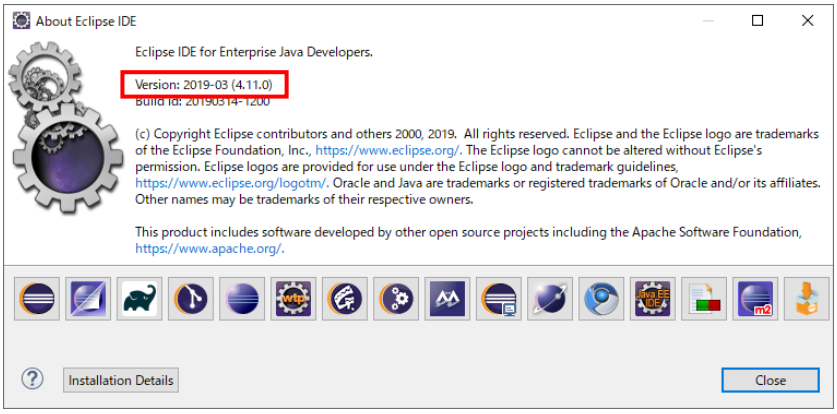


Fig. 14.1.2 (a) Eclipse IDE for Enterprise Java Developers (Version: 2019-03 (4.11.0))

14.2 SETTINGS ECLIPSE PLUGINS

The following software must be installed in order to use the Eclipse development environment.

- Eclipse
- ROBOGUIDE

14.2.1 Installing Eclipse

This section describes the procedure to install Eclipse and Fanuc support plugins.

Procedure 14-1 Installing Eclipse

Procedure

1. Install the following Eclipse software.

   Eclipse (English version)
   https://www.eclipse.org/downloads/

> **Note**
> - Please install JDK (Java Development Kit) before Eclipse installation.
> - This document does not describe how to use Eclipse itself.

2. After the installation, copy all files and folders for the FANUC support plugins (“fanuc_support” folder) into the Eclipse “dropins” folder, and perform Eclipse clean execution.

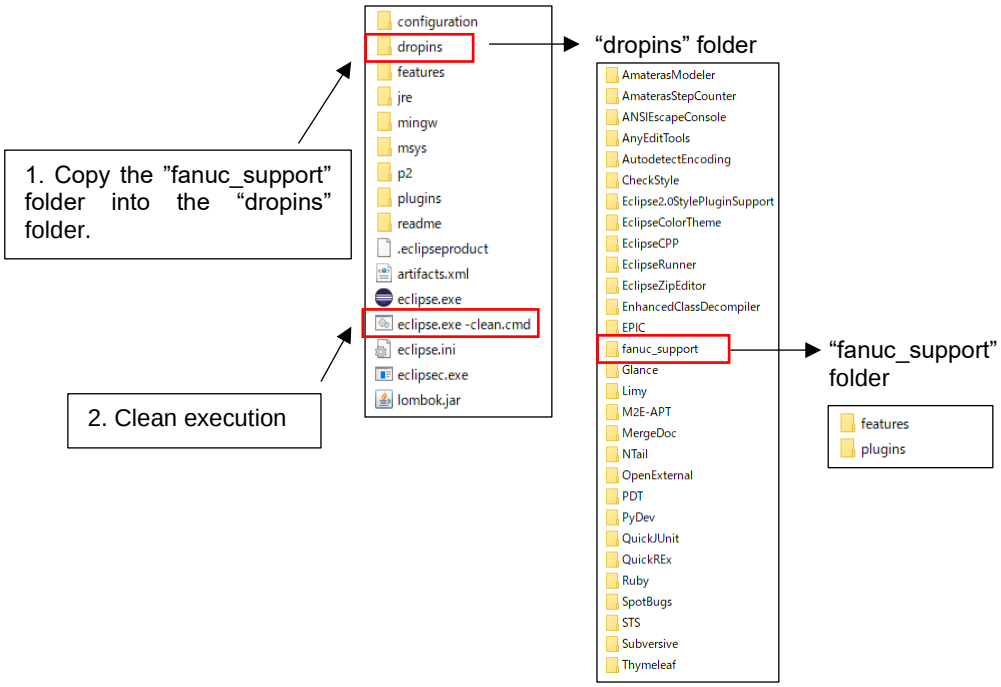


Fig. 14.2.1 (a) Eclipse dropins folder and clean execution file

14.2.2 Installing ROBOGUIDE

Be sure to complete installing ROBOGUIDE before running Eclipse with the installed Fanuc support plugins. If ROBOGUIDE is not installed, Eclipse and the installed Fanuc support plugin will not run. If you would like to run Eclipse in an environment where ROBOGUIDE is not installed, you will need to remove the Fanuc support plugin.

14.2.3 Running Eclipse

Run "eclipse.exe" in the folder where Eclipse was installed to run Eclipse software. For the procedures to setup the development environment, please refer to the following sections.

14.3 KAREL DEVELOPMENT ENVIRONMENT

14.3.1 Preparation for KAREL Development Environment

This section describes the procedure to setup KAREL development environment.

Procedure 14-2 Preparation for KAREL Development Environment

Procedure

1. Select "File" -> "New" -> "Other" from the Eclipse main menu.

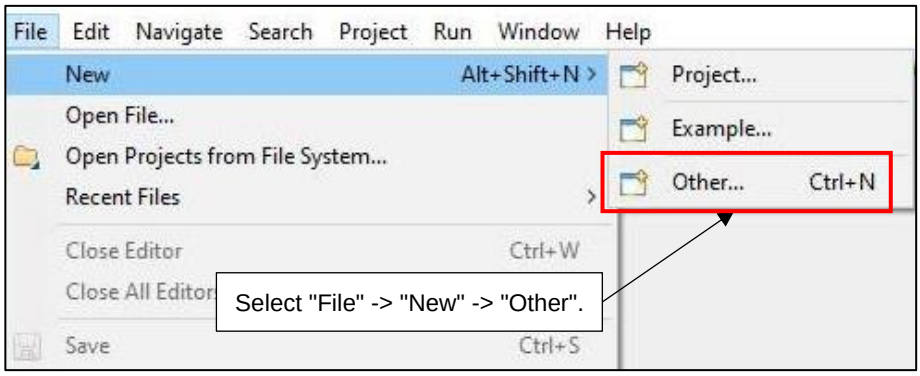


Fig. 14.3.1 (a) Running Eclipse

2. Select “KAREL Project” under “FANUC” in the wizard.

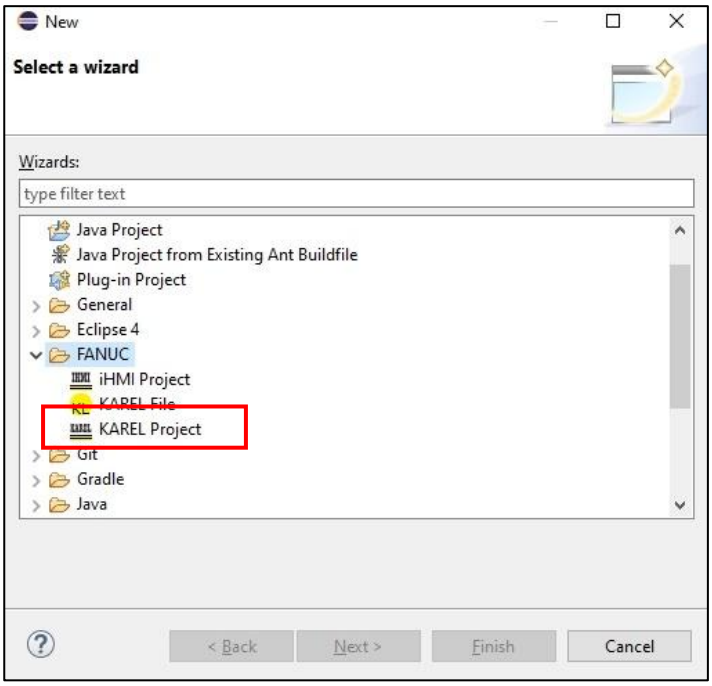


Fig. 14.3.1 (b) KAREL Project

3. Enter any project name in the “Project name” field.

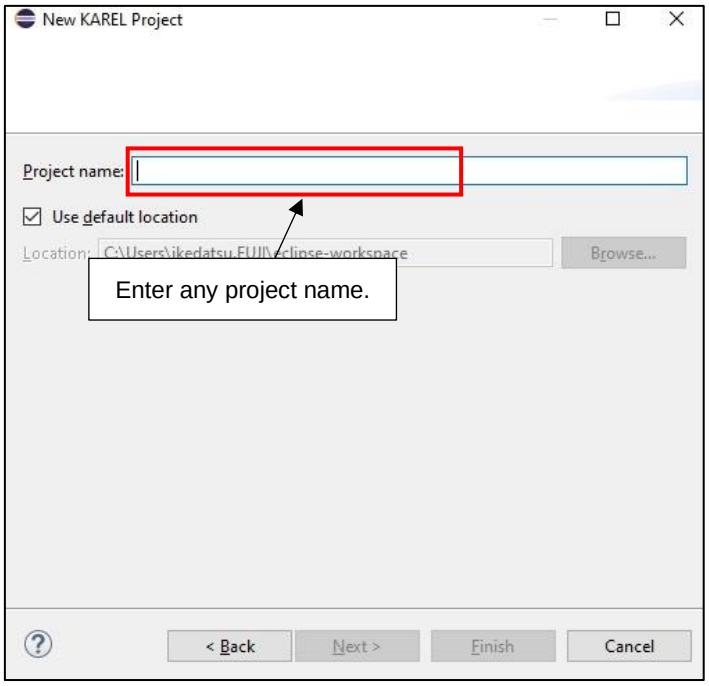


Fig. 14.3.1 (c) Entering KAREL project name

4. When creating a project while automatic building is enabled, a dialog box will appear to confirm whether it should be disabled. Click “Yes”. The automatic build settings can be checked from “Project” -> “Build Automatically”.

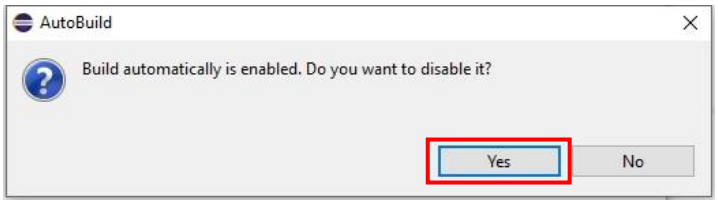


Fig. 14.3.1 (d) Dialog to confirm for automatic building

5. A workspace is created with the following folder structure.

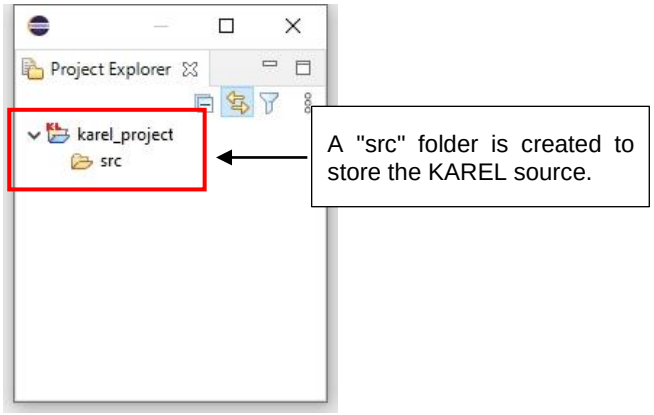


Fig. 14.3.1 (e) Folder to store the KAREL source

6. Perform the following settings on the tablet TP to enable debugging on the ROBOGUIDE / Robot controller.

   (1) Go to the “TEST CYCLE Setup” screen, and set “Group Motion” to “DISABLE”, and set “Step statement type” to “ROUTINE”.

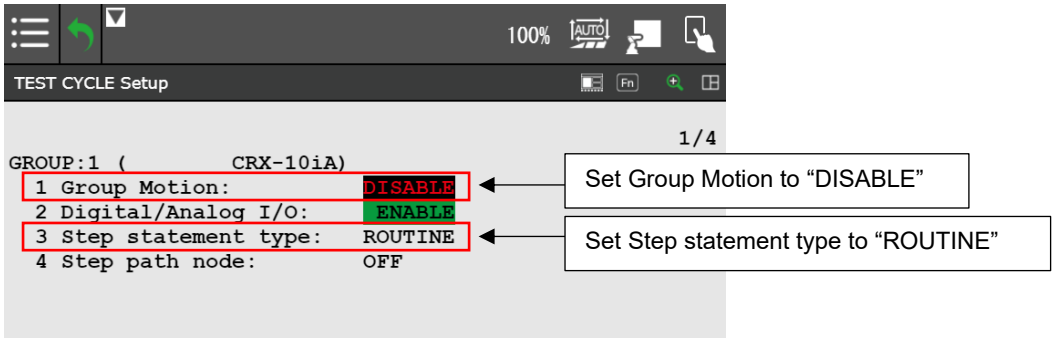


Fig. 14.3.1 (f) Setting Group Motion and Step statement type

(2) Go to the “SYSTEM” -> “System/Config” screen, and set “Remote/Local setup” to “Remote”.

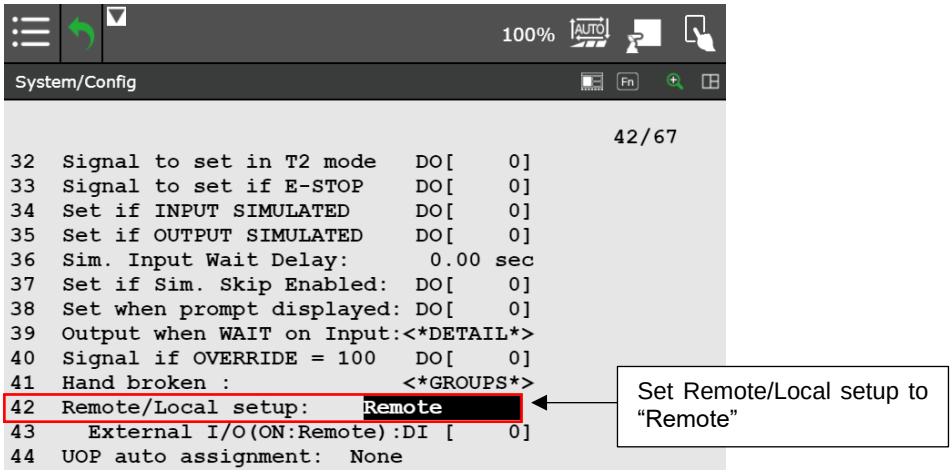


Fig. 14.3.1 (g) Remote/Local setup

(3) Go to the “SYSTEM” -> “SYSTEM Variables” screen, and set the system variable “$RMT_MASTER” to 1.

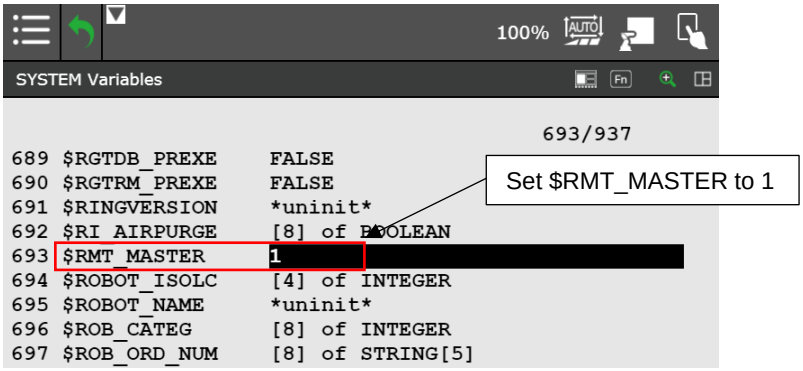


Fig. 14.3.1 (h) Setting $RMT_MASTER

(4) Turn off the TP switch.

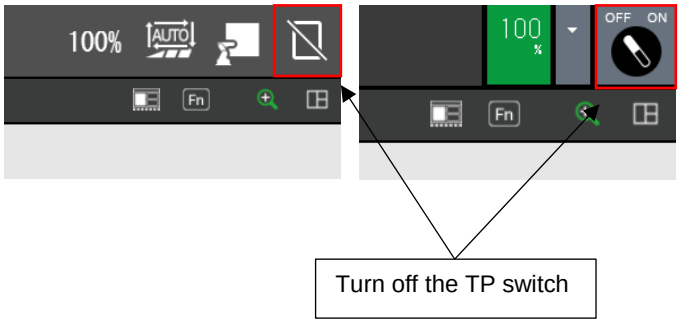


Fig. 14.3.1 (i) Turning off the TP switch

14.3.2 Building KAREL Programs

This section describes the procedure for building the KAREL program (*.kl) to generate the binary file (*.pc) and routine stack file (*.rs).

Procedure 14-3 Building KAREL Programs

Procedure

1. Right-click the target project, and select “Properties”. Then change the version of the build target to “V9.40-1”.

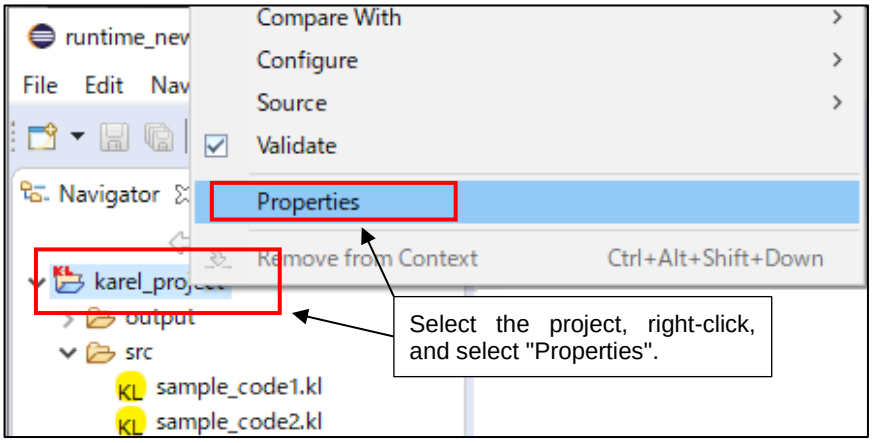


Fig. 14.3.2 (a) Properties of the project

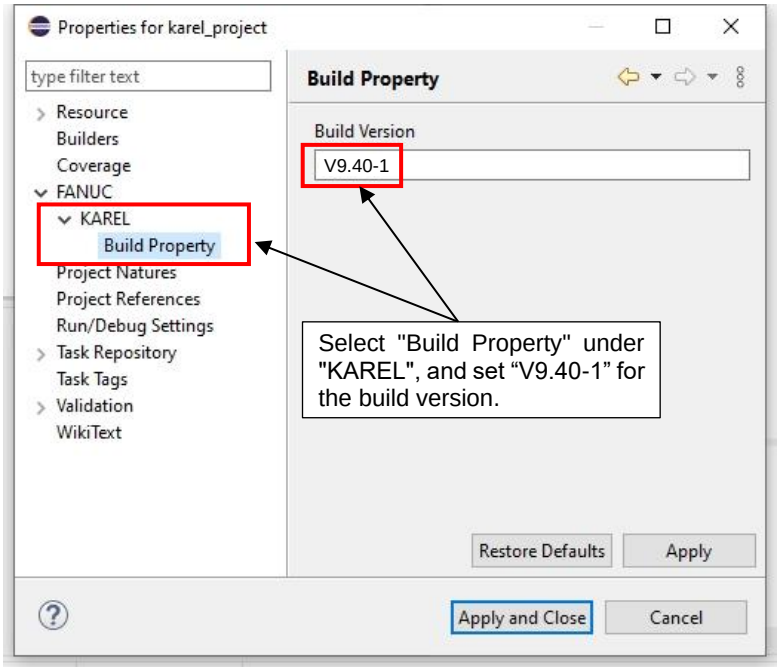


Fig. 14.3.2 (b) Changing the version of the build target

2. Select the target project, and then select "Project" -> "Build Project" from the Eclipse main menu to build the target KAREL program (*.kl). If the build process is successfully done, the binary file (*.pc) and routine stack file (*.rs) will be generated in the output folder.

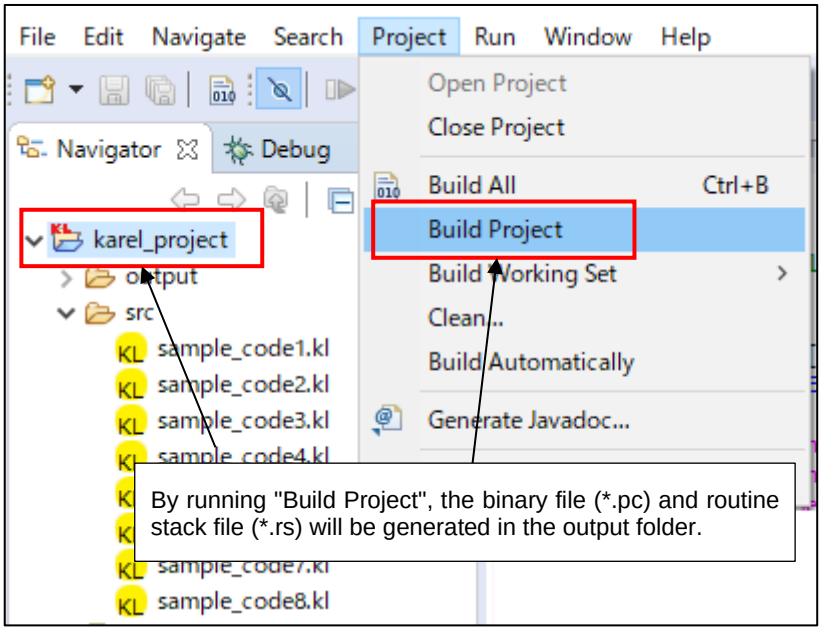


Fig. 14.3.2 (c) Building KAREL Project

3. If an error occurs during the build process, take actions as described below according to the error that was displayed on the console.

Table 14.3.1 (a) Error details

No.	Error details	Cause	Remedy
1	'ktrans.exe' is not recognized as an internal or external command, operable command, operable program, or batch file.	The path to the build tool, which is incorporated in ROBOGUIDE, is not registered in the environment variable.	Register the path in the environment variable where “ktrans.exe” is stored. For 64 bit OS Windows 10, the file location is “C:¥Program Files (x86)¥FANUC¥WinOLPC¥bin”.

14.3.3 Debugging KAREL Programs

Load the binary files (*.pc) that was generated by the build process into ROBOGUIDE / Robot controller to debug it. This section describes the procedure to debug the KAREL programs.

Procedure 14-4 Debugging KAREL Programs

Procedure

1. Build the target KAREL program (*.kl) to generate the binary file (*.pc) and routine stack file (*.rs) in the output folder in advance.

2. Select “Window” -> “Perspective” -> “Open Perspective” -> “Other” from the Eclipse main menu.

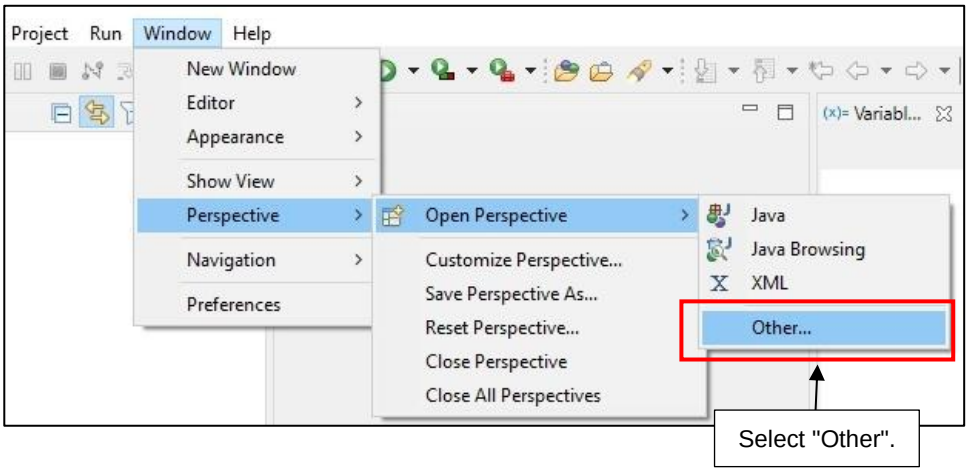


Fig. 14.3.3 (a) Eclipse Perspective

3. Select “Debug” in the Perspective Dialog, and then click “Open”. The debug menu will be added to the Eclipse menu bar.

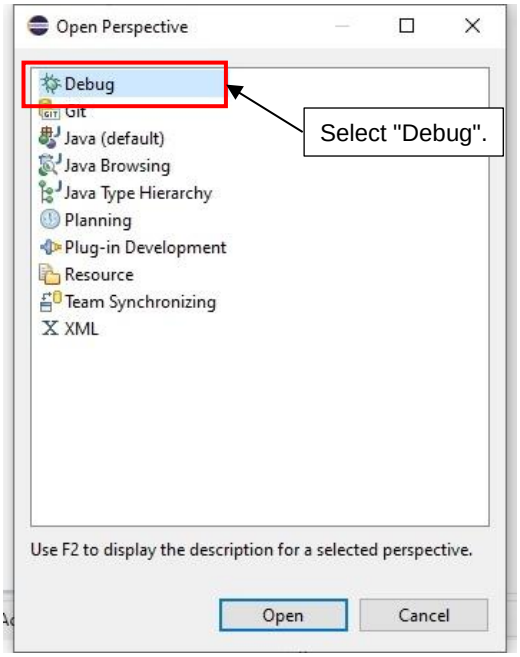


Fig. 14.3.3 (b) Debug menu

4. UNLOCK KCL in the “SETUP” -> “Host Comm” -> “HTTP” screen on the tablet TP in the ROBOGUIDE / Robot controller so as to prevent KCL Command Error.

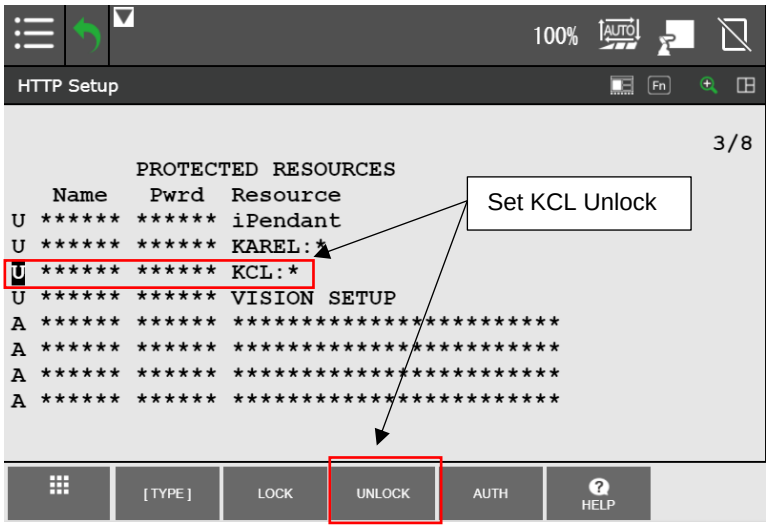


Fig. 14.3.3 (c) Unlocking KCL

5. Click “Run Configurations” under the “Run” icon pull-down menu to create the debug launcher.

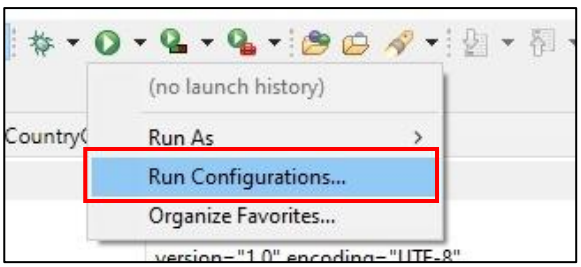


Fig. 14.3.3 (d) Run Configurations

6. Right-click “FANUC KAREL” and select “New Configuration” in the shortcut menu to create a new executable program for debugging.

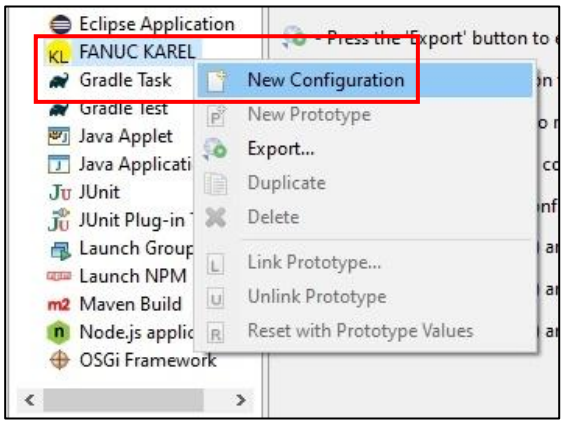


Fig. 14.3.3 (e) Creating a new executable program for debugging

7. Enter any name and create a launcher.



Fig. 14.3.3 (f) Creating a launcher for debugging

8. Click the “Browse” button under “projects” and select the project.

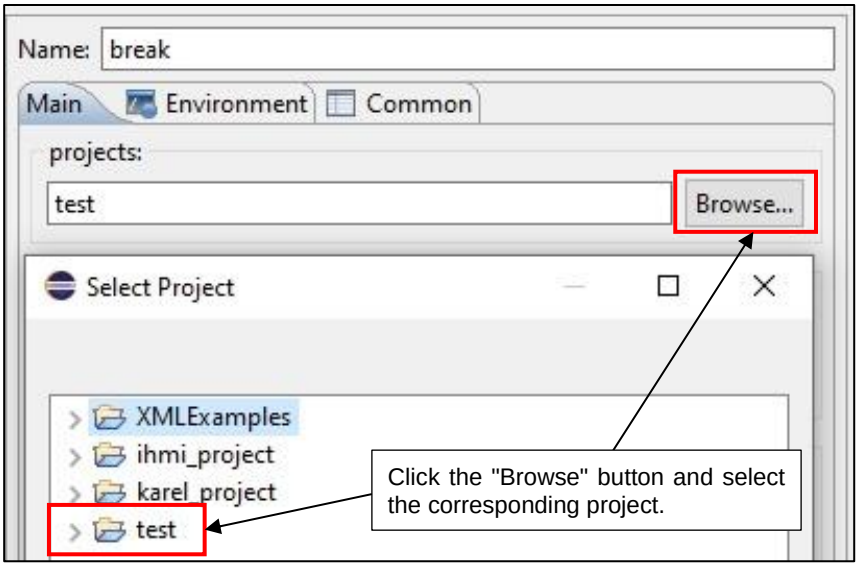


Fig. 14.3.3 (g) Selecting the project

9. Click the “Browse” button under “target programs” and select the file name under “Select File”. The file name and the program name will be automatically entered when the file is selected.

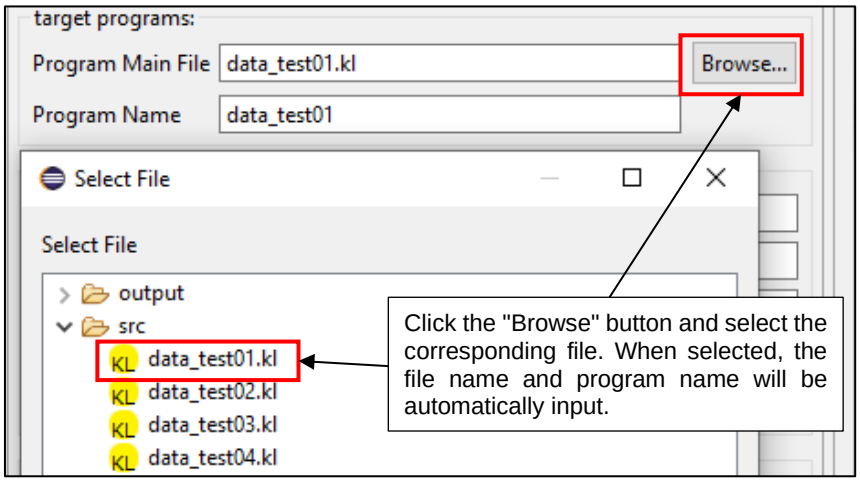


Fig. 14.3.3 (h) Selecting the file

10. Set up the destination server that will be connected to.
    - IP Address: Set “localhost” in case of ROBOGUIDE. Set the IP address in case of robot controller.
    - Port Number: Set “80” as default. If you have changed the port number, set the port number you set.
    - FTP Port Number: Set “21” as default. If you have changed the FTP port number, set the FTP port number you set.

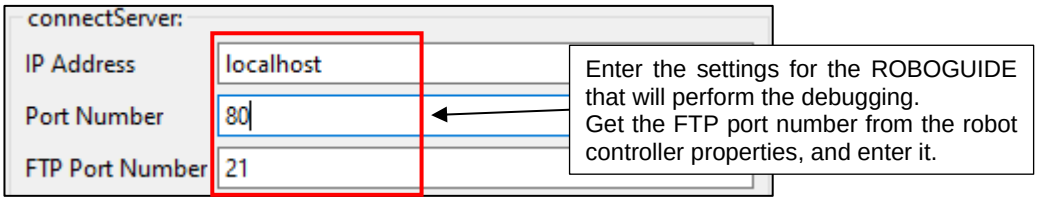


Fig. 14.3.3 (i) Setting the destination server

11. If the user has protected the ROBOGUIDE / Robot controller with a password, enter the FTP user name and FTP password.

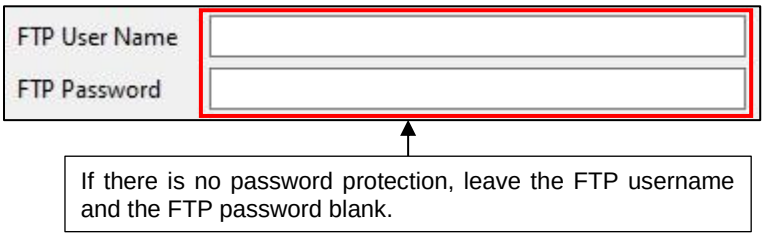


Fig. 14.3.3 (j) Entering the FTP user name and FTP password

12. Open the KAREL program to be debugged on Eclipse, and set a breakpoint by double-clicking anywhere on the ruler.

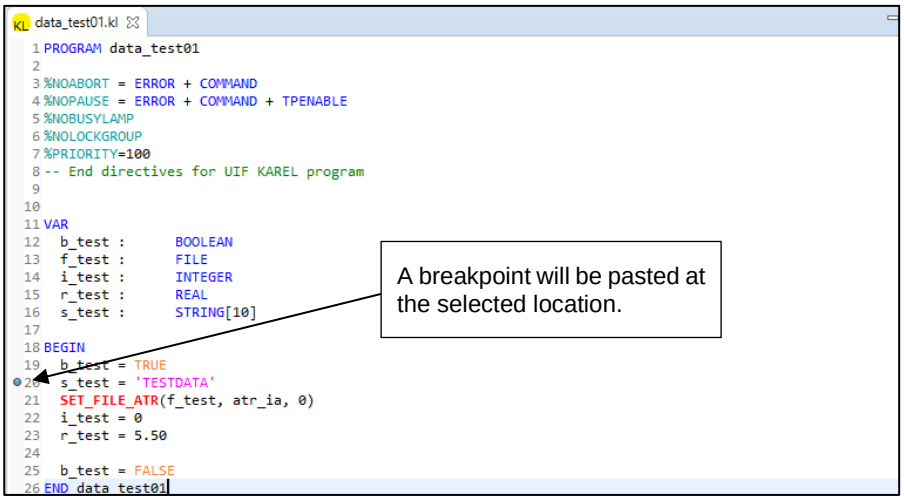


Fig. 14.3.3 (k) Setting a breakpoint

13. Run the created launcher to load the binary file (*.pc) into the target ROBOGUIDE / Robot controller, and run the target program.

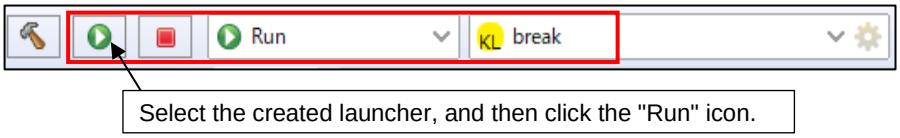


Fig. 14.3.3 (l) Running the launcher for debugging

14. When the binary files generated through building are loaded in the ROBOGUIDE / Robot controller, the following dialog box may be displayed. In that case, click the “OK” button, perform a cold start on the ROBOGUIDE / Robot controller, and then retry debugging.

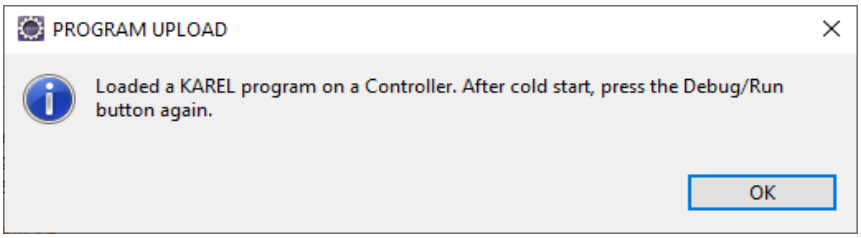


Fig. 14.3.3 (m) Confirmation of KAREL program loading

15. When get the program running, the KAREL source file will be opened in Eclipse view, and the cursor will be positioned at the line where the program stopped due to the breakpoint.

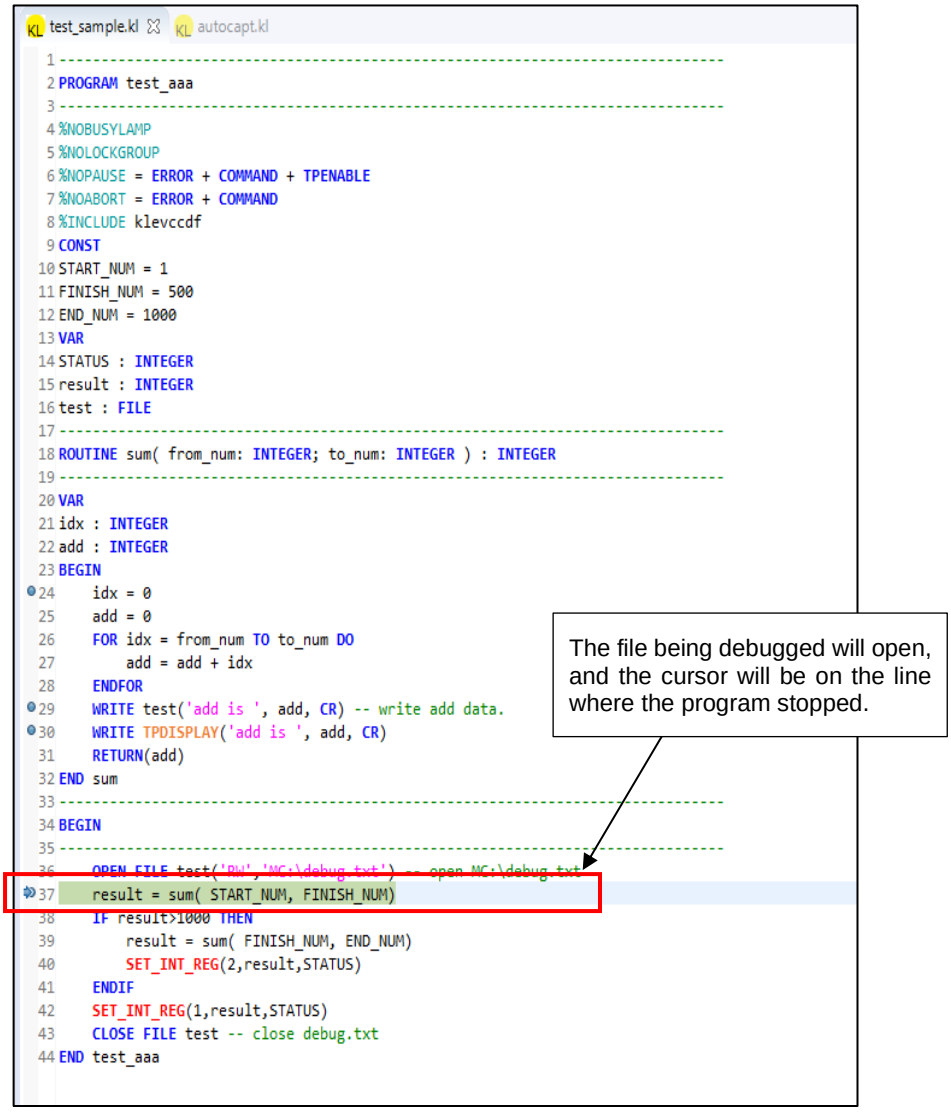


Fig. 14.3.3 (n) Cursor at the line program stopped

16. The program name and the line where the program stopped are displayed in the debug window. If the stop position is in the routine, the line that calls the routine and the line within the routine where the program stopped will be displayed.

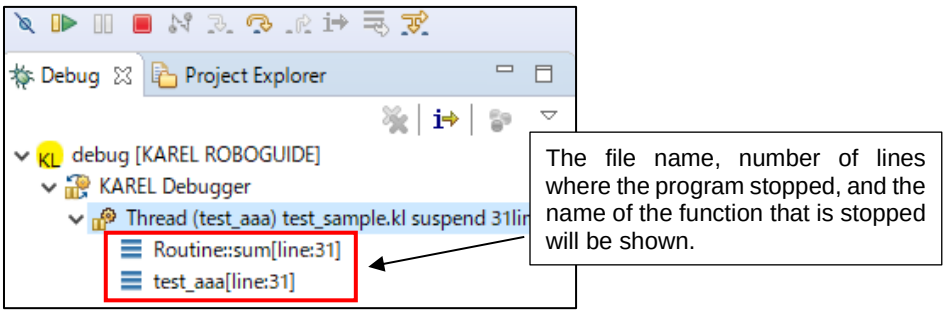


Fig. 14.3.3 (o) Program name and the lines where the program stopped

17. Select “KAREL Debugger” to enable the “Resume” and “Terminate” buttons. Select “Thread” to enable the “Resume”, “Terminate”, and “Step Over” buttons.

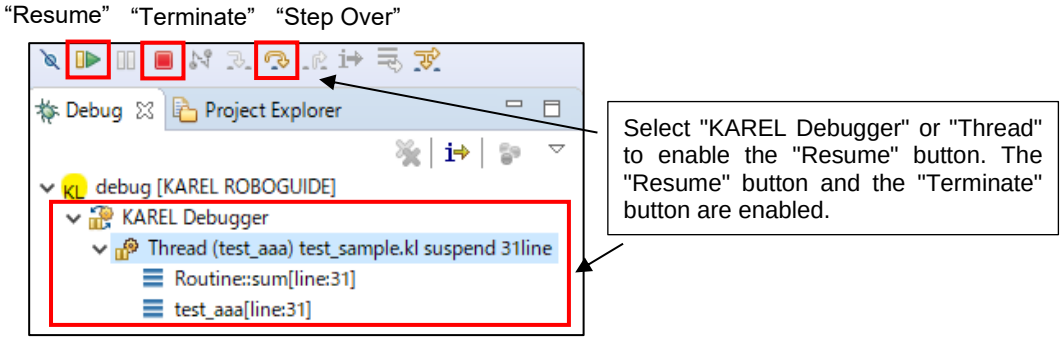


Fig. 14.3.3 (p) Resume button and Terminate button

18. When the "Resume" button is clicked, the program continues running and stops at the next breakpoint.

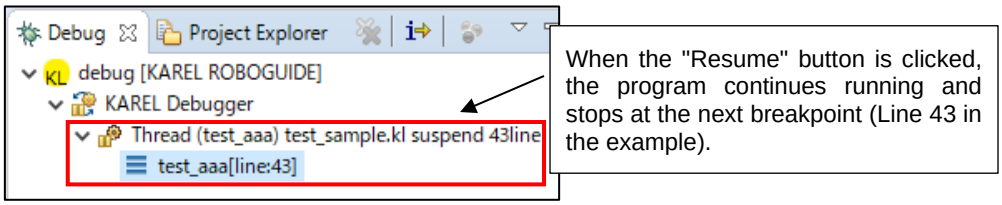


Fig. 14.3.3 (q) Resume button

19. When the "Step Over" button is clicked, the program stops at the next line.

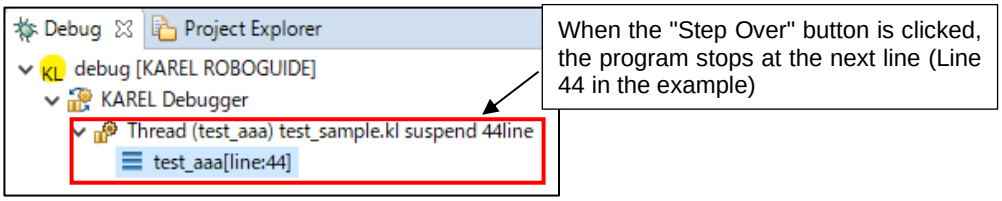


Fig. 14.3.3 (r) Step Over button

20. The thread is terminated when the "Terminate" button is clicked or the program is terminated.

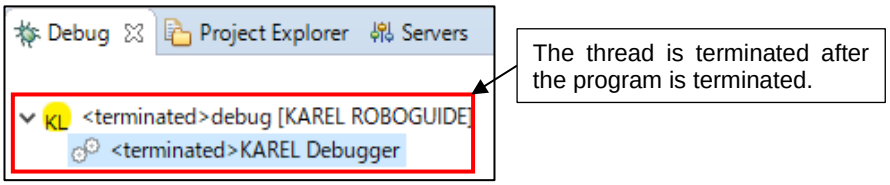


Fig. 14.3.3 (s) Terminating the thread

14.3.4 Monitoring KAREL Variables While Debugging

To monitor KAREL variables while debugging, select the KAREL program name or routine name when the program is stopped at a breakpoint. The value of the global variable of the running program and the value of the local variable in the routine are displayed.

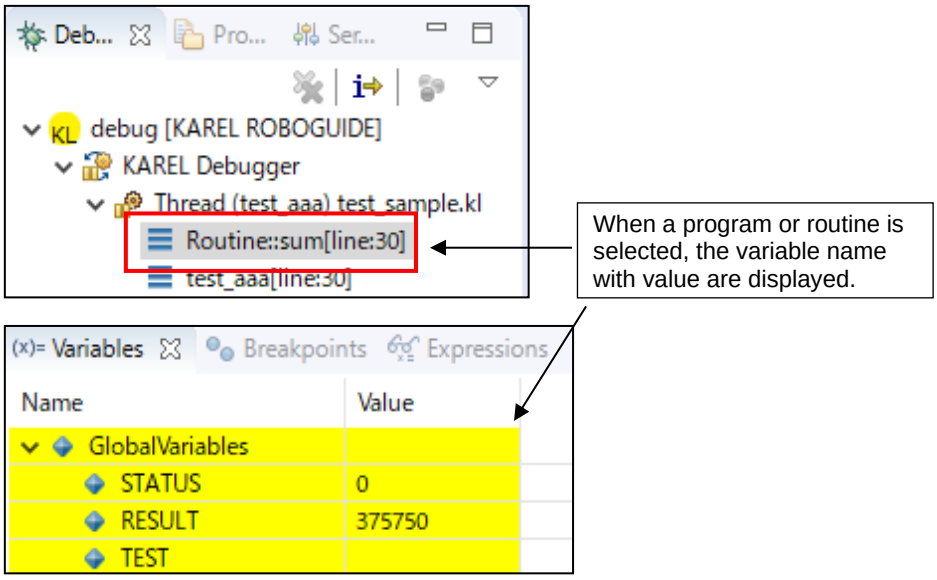


Fig. 14.3.4 (a) Checking the KAREL variable while debugging

14.3.5 Changing KAREL Variables While Debugging

You may change the value shown in the “Value” field for the KAREL variables in the KAREL program being debugged while the program is stopped at a breakpoint.

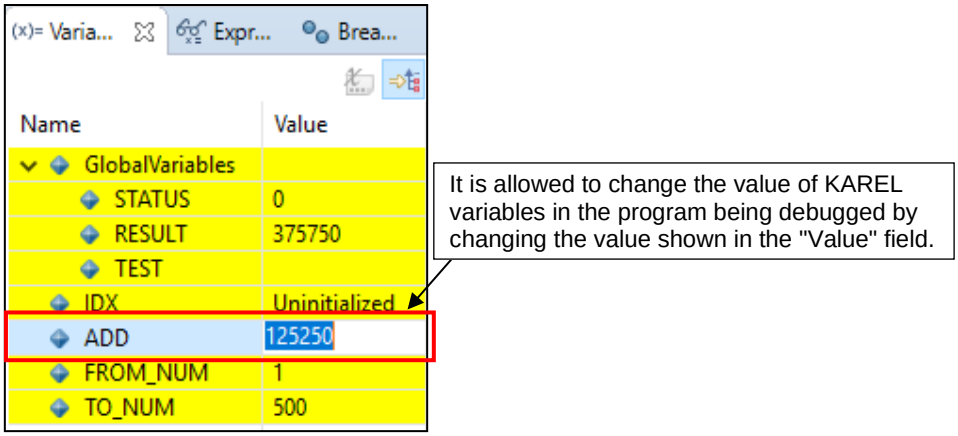


Fig. 14.3.5 (a) Changing KAREL variable while debugging

14.3.6 Customizing KAREL Editor

The text color displayed in the view of KAREL program editor may be customized to your preferred colors. This section describes the procedure to change the color.

Procedure 14-5 Customizing KAREL Editor

Procedure

1. Select “Window” -> “Preferences” from the Eclipse main menu to open the preferences dialog box.

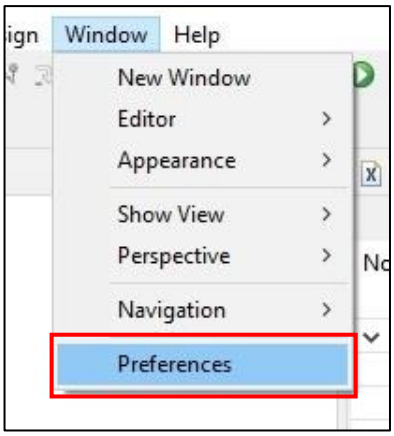


Fig 14.3.6 (a) Preferences dialog box

2. Enter “KAREL” in the filter input field.

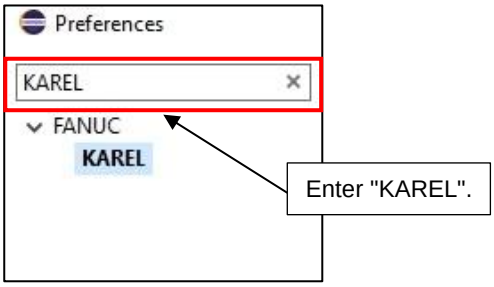


Fig. 14.3.6 (b) Filter input field

3. Change the color freely using the displayed KAREL editor color preference.

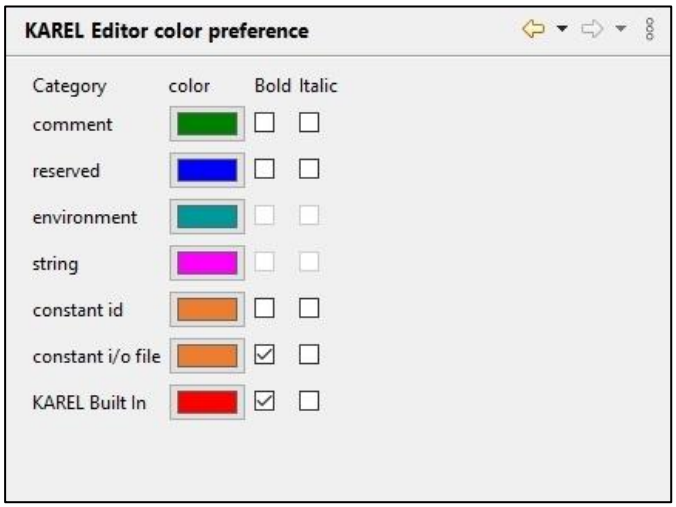


Fig. 14.3.6 (c) Changing KAREL editor color

14.4 iHMI COMPONENT DEVELOPMENT ENVIRONMENT

14.4.1 Preparation for iHMI Component Development Environment

This section describes the procedure to setup iHMI component development environment.

Procedure 14-6 Preparation for iHMI component development environment

Procedure

1. Select “File” -> “New” -> “Other” from the Eclipse main menu.

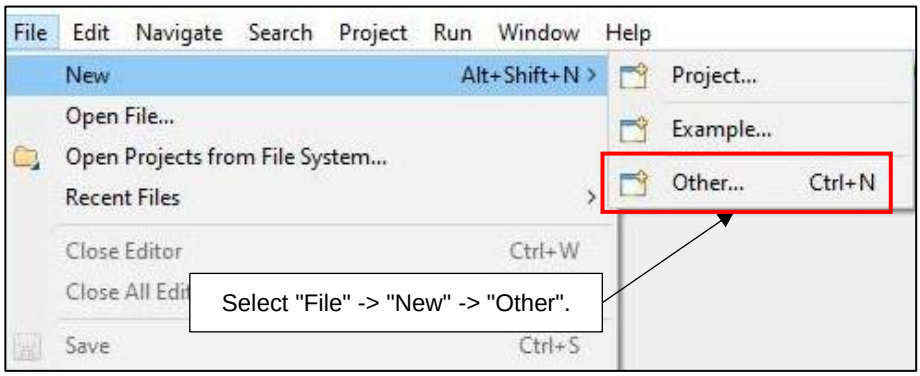


Fig. 14.4.1 (a) Running Eclipse

2. Select “iHMI Project” under “FANUC” in the wizard.

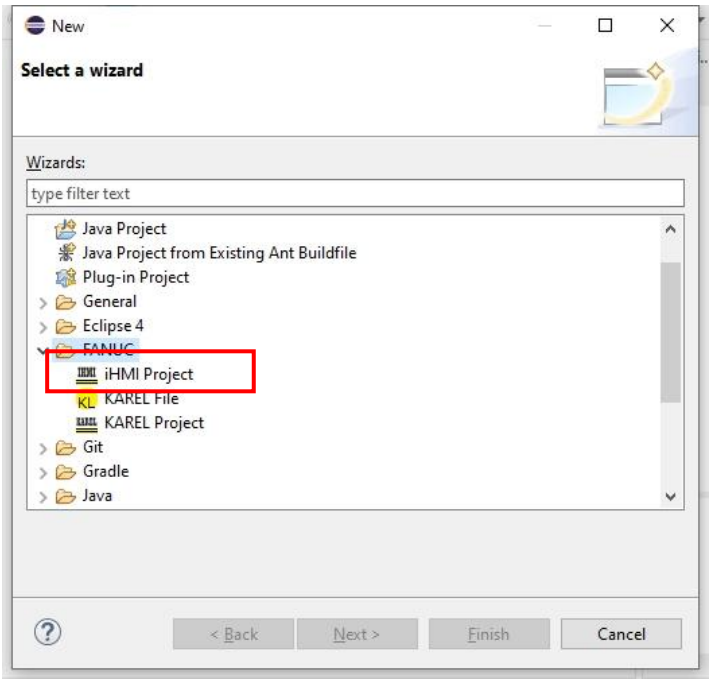


Fig. 14.4.1 (b) iHMI Project

3. Enter any project name in the “Project name” field.

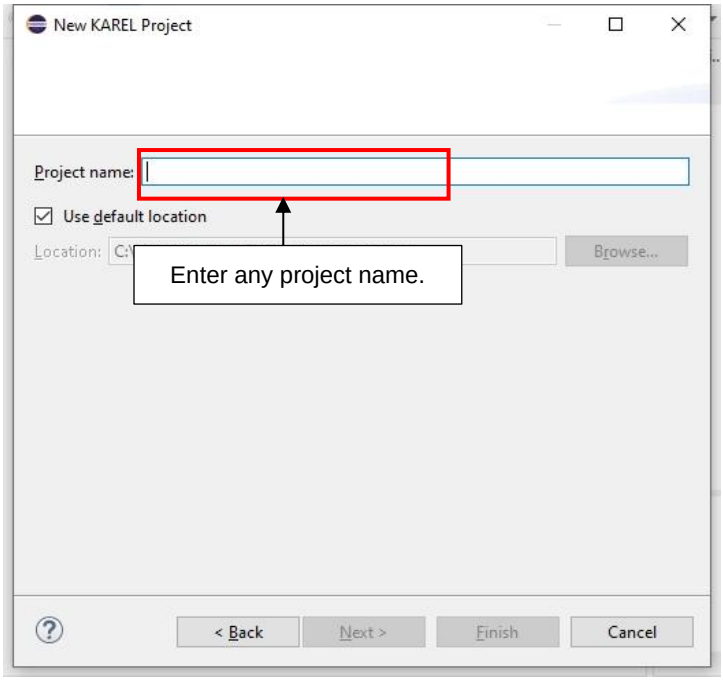


Fig. 14.4.1 (c) Entering iHMI project name

4. A workspace is created with the following folder structure.

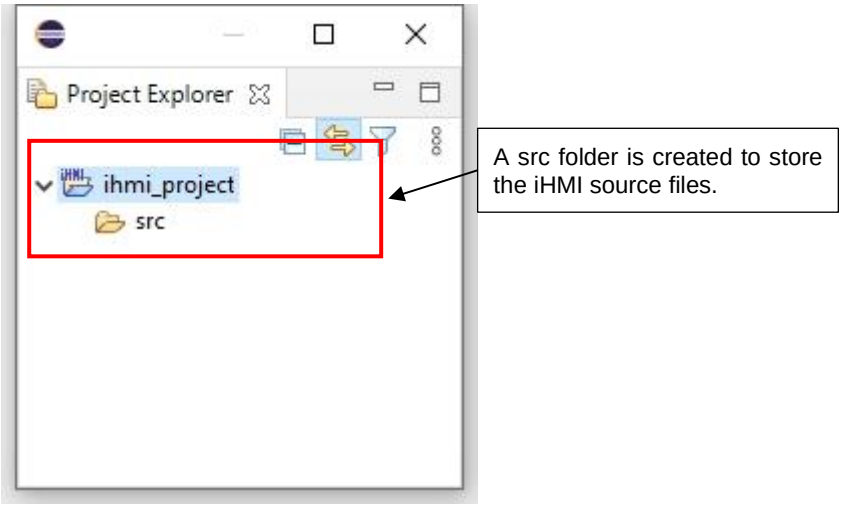


Fig. 14.4.1 (d) Folder to store iHMI source file

14.4.2 Building iHMI Component Programs

The iHMI component development environment generates *.html file or *.stm (*.htm) files through bulding the base file *.ishtm (*.ihtml) and the dictionary files to be loaded into the Robot controller. Please refer to chapter 7 for creating a base file *.ishtm (*.ihtml), and chapter 7 and 8 for building and installing the generated file *.stm (*.htm).

Right-click the work project folder and select "Build Project" to run the build and generate a *.stm (*.htm) file in the output folder to be loaded into the Robot controller. An *.ipl file (FANUC plugin installer package file), which is a zipped package of all of the output files, will also be generated in the output folder. The generated *.ipl file is used for installing plugin software into the Robot controller.

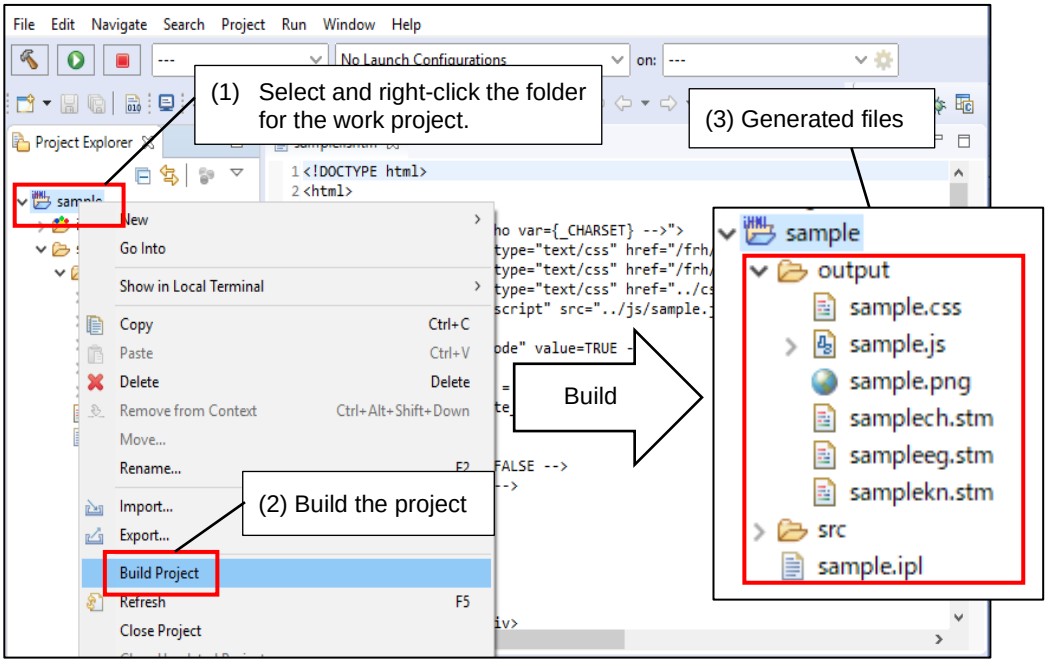


Fig. 14.4.2 (a) Building iHMI component program

14.4.3 Eclipse Clean Execution

The build process will not be performed if the files are not updated. This section describes the procedure to run the Eclipse Clean function to force a project to build.

Procedure 14-7 Running the Eclipse Clean Function

Procedure

1. Select “Project” -> “Clean” from the Eclipse main menu to display the “Clean” dialog box.

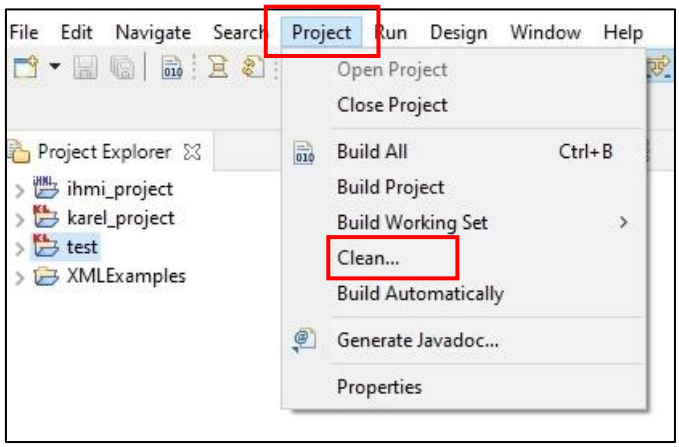


Fig. 14.4.3 (a) Clean dialog box

2. Click the “Clean” button in the “Clean” dialog box.

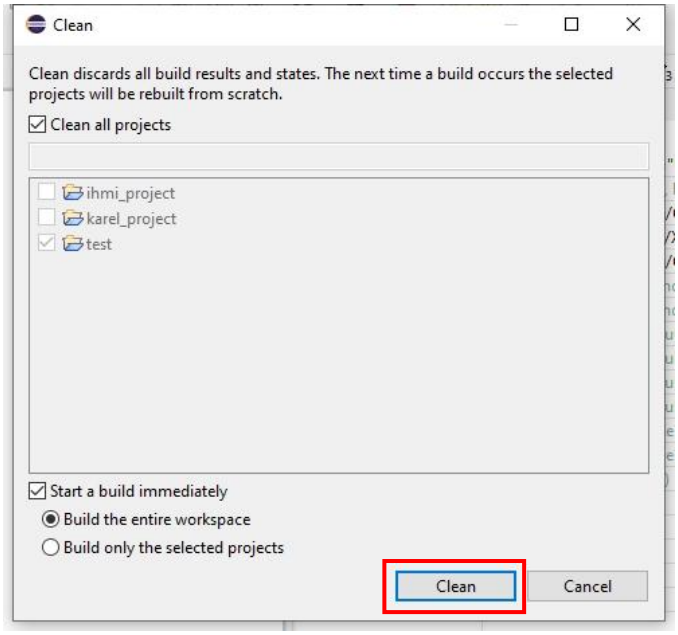


Fig. 14.4.3 (b) Clean execution

15 CRX SERIES MECHANICAL UNIT

This chapter describes the specification of FANUC Robot CRX series mechanical unit.

15.1 SPECIFICATIONS

The following table shows the robot specifications.

Table 15.1 (a) Specifications (Note 1)

Item		Specification	
Model		CRX-10*i*A	CRX-10*i*A/L
Type		Articulated type	
Controlled axes		6-axis (J1, J2, J3, J4, J5, J6)	
Reach		1249 mm	1418 mm
Installation		Floor, Upside-down, Wall & Angle mount (Note 2)	
Motion range	J1-axis	380° (6.63 rad)	360° (6.28 rad)
	J2-axis	360° (6.28 rad)	
	J3-axis	570° (9.95 rad)	540° (9.95 rad)
	J4-axis	380° (6.63 rad)	
	J5-axis	360° (6.28 rad)	
	J6-axis	380° (6.63 rad)	
Maximum speed (NOTE 3)		1000mm/s (NOTE 4) (maximum speed 2000mm/s (NOTE 5))	
Maximum load at wrist		10kg	
Allowable load moment at wrist	J4-axis	34.8Nm	
	J5-axis	26.0Nm	
	J6-axis	11.0Nm	
Allowable load inertia at wrist	J4-axis	1.28kg.m^2	
	J5-axis	0.90kg.m^2	
	J6-axis	0.30kg.m^2	
Repeatability (NOTE 6)		±0.04mm	
Robot mass		40kg	40kg
Dust proof and drip proof mechanism (NOTE 7)		Conform to IP67	
Acoustic noise level		Less than 70dB (NOTE 8)	
Installation environment		Ambient temperature: Operating 0 to 45℃ (NOTE 9) Storage,Transport -10 to 60℃ Ambient humidity: Normally: 75%RH or less (No dew or frost allowed) Short time: 95%RH or less (Within 1 month) Permissible altitude: Above the sea 1000m or less Free of corrosive gases (NOTE 10) Vibration acceleration : 4.9m/s^2 (0.5G) or less (NOTE 11) Environment without fire	

NOTE

1 Even if the robot is used according to the defined specifications, motion programs might shorten reducer life or cause the robot to overheat. Use ROBOGUIDE for further evaluation before running production.
2 There is no limit of operating space for all the installation types.
3 During short distance motions, the axis speed may not reach the maximum value stated.
4 It is necessary to set a motion speed according to risk assessment of system considering pinching with the surroundings.
5 If the area is monitored by a safety sensor (located separately).
6 Compliant with ISO9283.
7 Definition of IP code
Definition of IP 67
6 = Dust-tight: Complete protection against contact
7 = Protection from water immersion: Ingress of water in harmful quantity shall not be possible when the enclosure is immersed in water under defined conditions of pressure and time.
8 This value is equivalent continuous A-weighted sound pressure level, which applied with ISO11201 (EN31201). This value is measured with the following conditions.
- Maximum load and speed
- Operating mode is AUTO
9 When robot is used in low temperature environment that is near to 0ºC, or not operated for a long time in the environment that is less than 0ºC in a holiday or the night, collision detection alarm (SRVO-050) etc. may occur since the resistance of the drive mechanism could be high immediately after starting the operation.
10 Contact the service representative, if the robot is to be used in an environment or a place subjected to hot/cold temperatures, severe vibrations, heavy dust, cutting oil splash and or other foreign substances.
11 Depending on the vibration of the floor or the hand, robot may stop due to the vibration in less than this value.

15.1.1 Dimensions of The Robot Base

The following figures show the robot base dimensions.

⚠ CAUTION

Flatness of robot installation surface must be less than or equal to 0.5mm.
Inclination of robot installation surface must be less than or equal to 0.5°.
If robot base is placed on uneven ground, it may result in the base breakage or low performance of the robot.

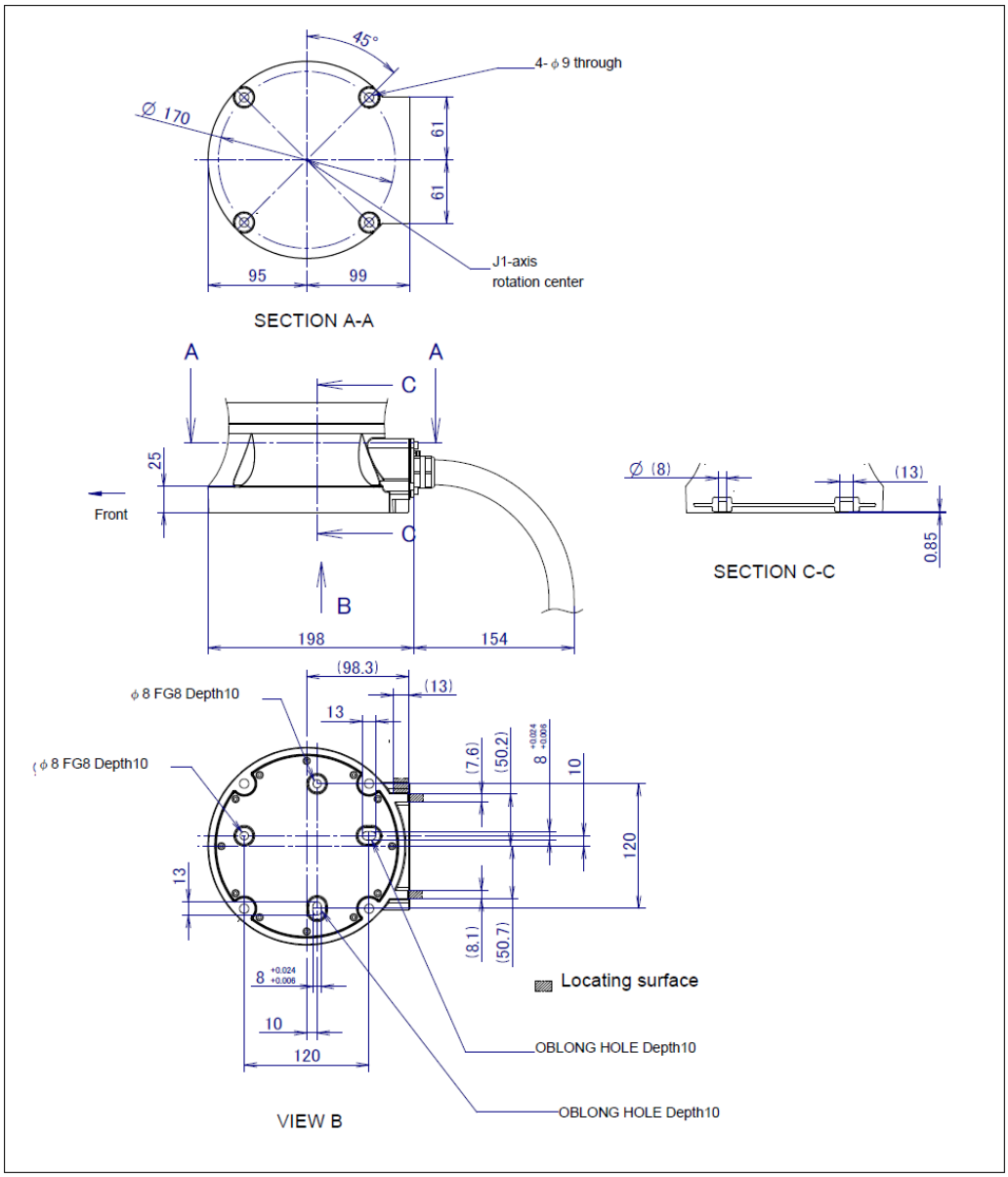


Fig. 15.1.1 (a) Dimensions of the robot base (back side connector)

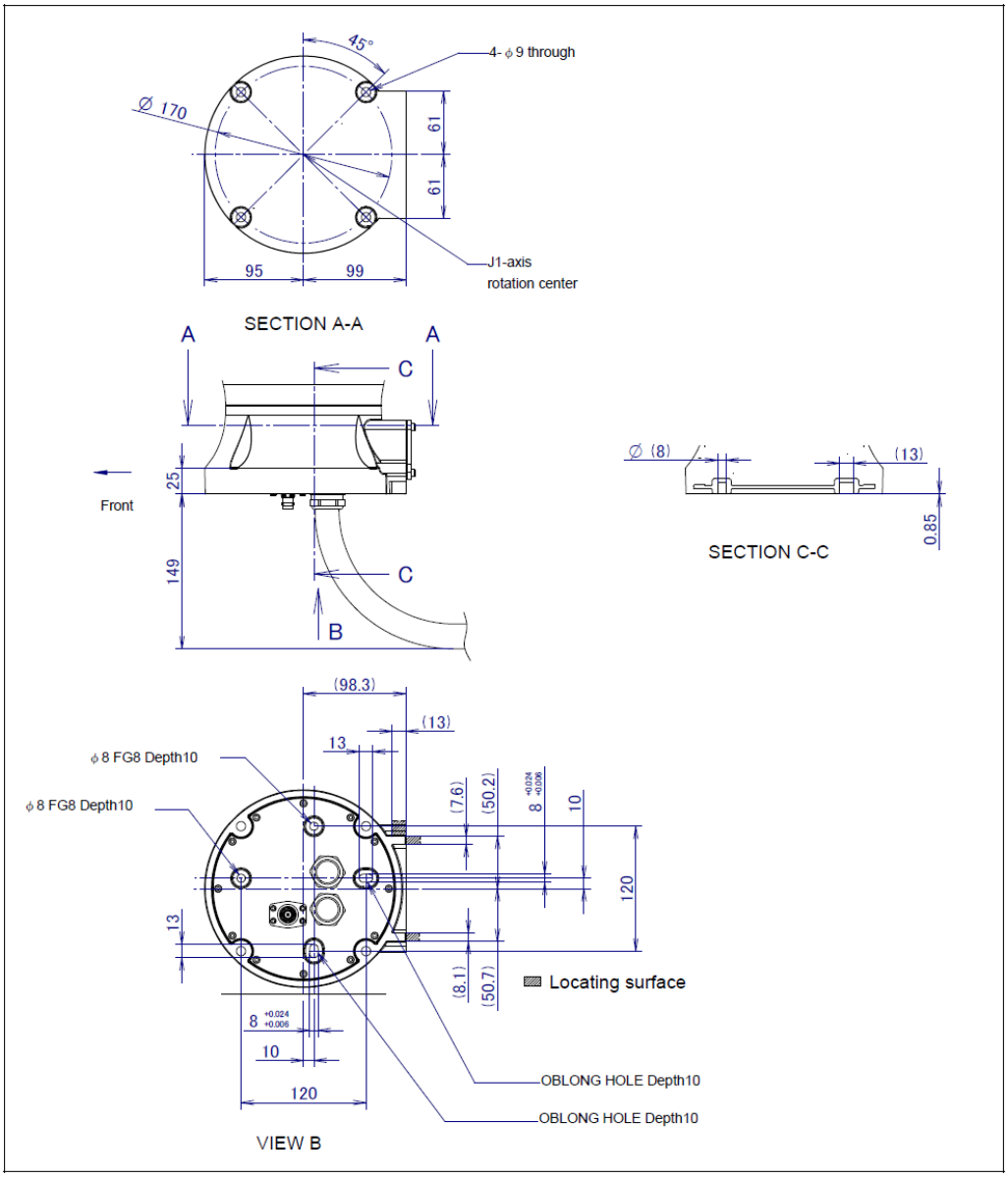


Fig. 15.1.1 (b) Dimensions of the robot base (bottom connector)

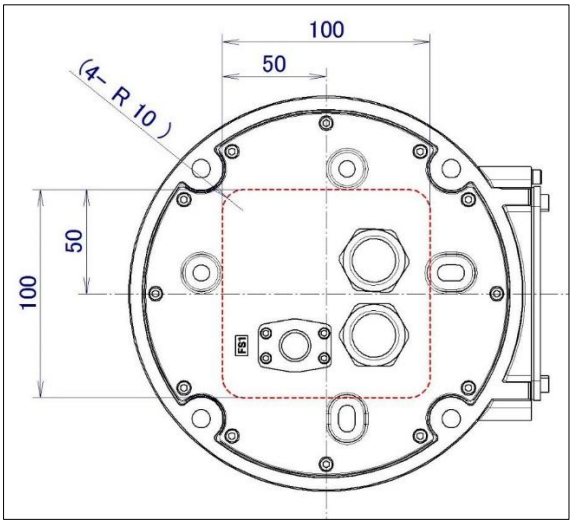


Fig. 15.1.1 (c) Hole Dimensions of Base (Bottom Connector)

15.1.2 Force and Moment That Acts on The Robot Base

The following figure and table indicate the force and moment applied to the robot base.

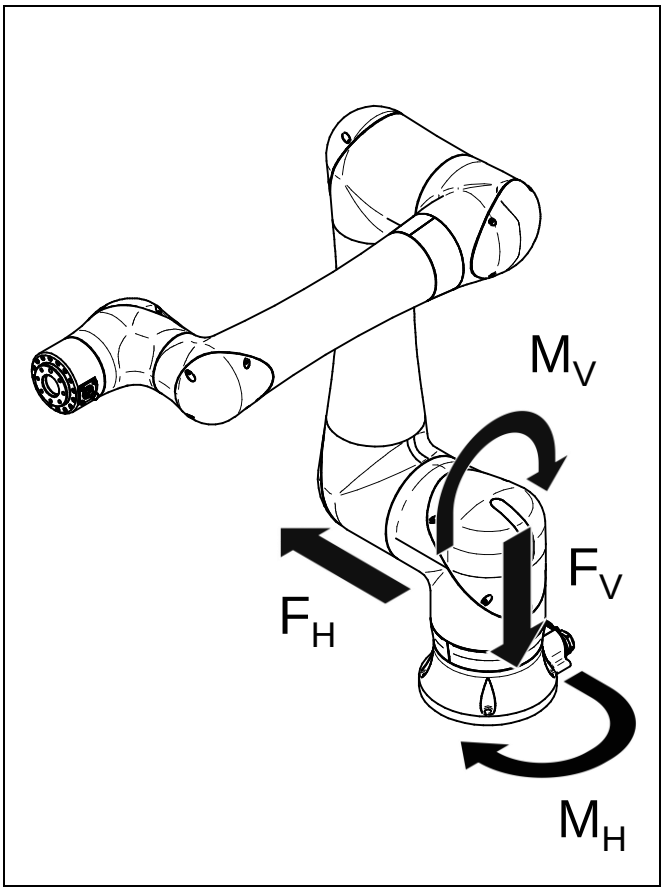


Fig. 15.1.2 (a) Force and moment that acts on the robot base

Table 15.1.2 (a) Force and moment that acts on the robot base

	Vertical moment MV [Nm]	Force in vertical direction FV N	Horizontal moment MH Nm	Force in horizontal direction FH N (kgf)
During stillness	310	490	0	0
During acceleration or deceleration	380	550	120	180
During Smooth stop	610	800	250	220

15.1.3 Tool Installation to Wrist

The following figure shows the figures for installing tools on the wrist. Select screws and positioning pins of a length that matches the depth of the tapped holes and pin holes.

⚠ CAUTION

- Notice the tooling coupling depth to wrist flange should be shorter than the flange coupling length.
- When the robot mounted the tool operates, the tool may collide with the robot, and if a part of body is pinched between the tool and the robot, it might cause heavy injury.

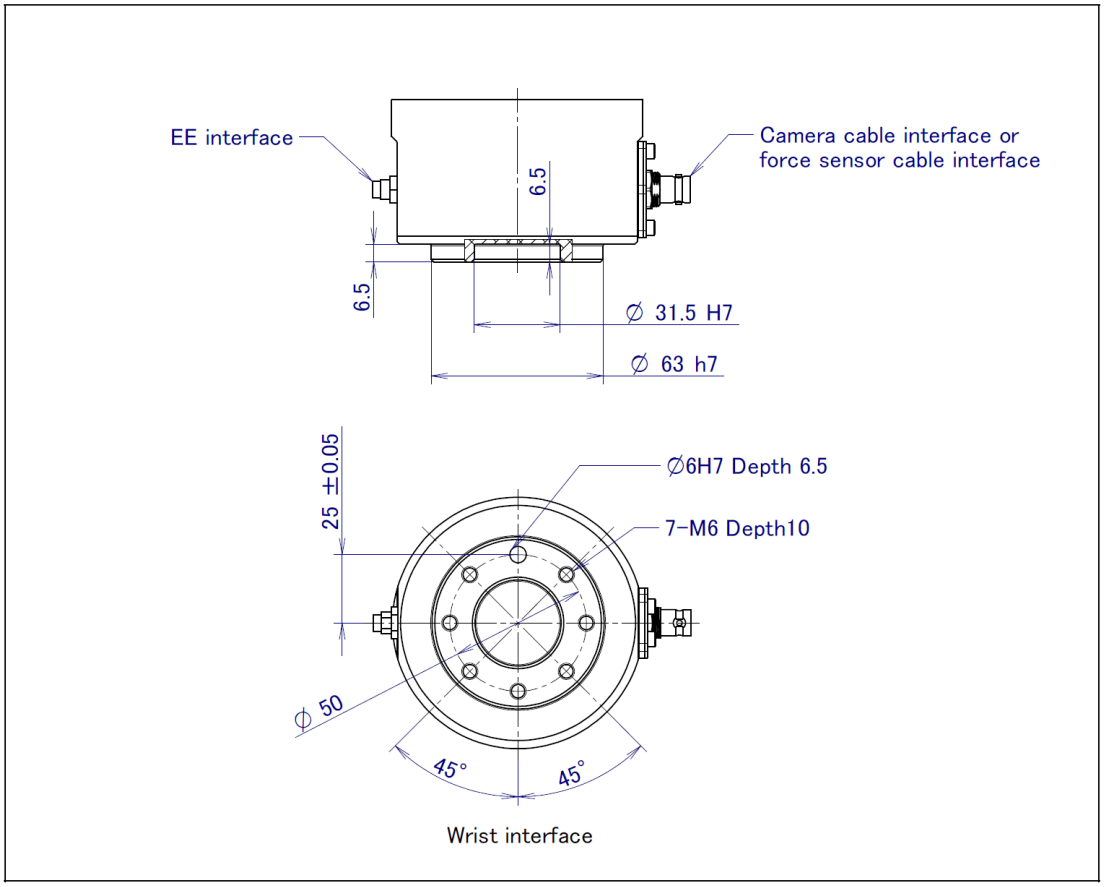


Fig. 15.1.3 (a) Tool interface (CRX-10*i*A, CRX-10*i*A/L)

15.1.4 Wrist Load Conditions

The following figure is diagrams showing the allowable load that can be applied to the wrist section.

- Apply a load within the region indicated in the graph.
- Please use it to meet the requirement of the allowable load moment and inertia at wrist. See Table 14.1 (a) about allowable load moment and inertia at wrist.

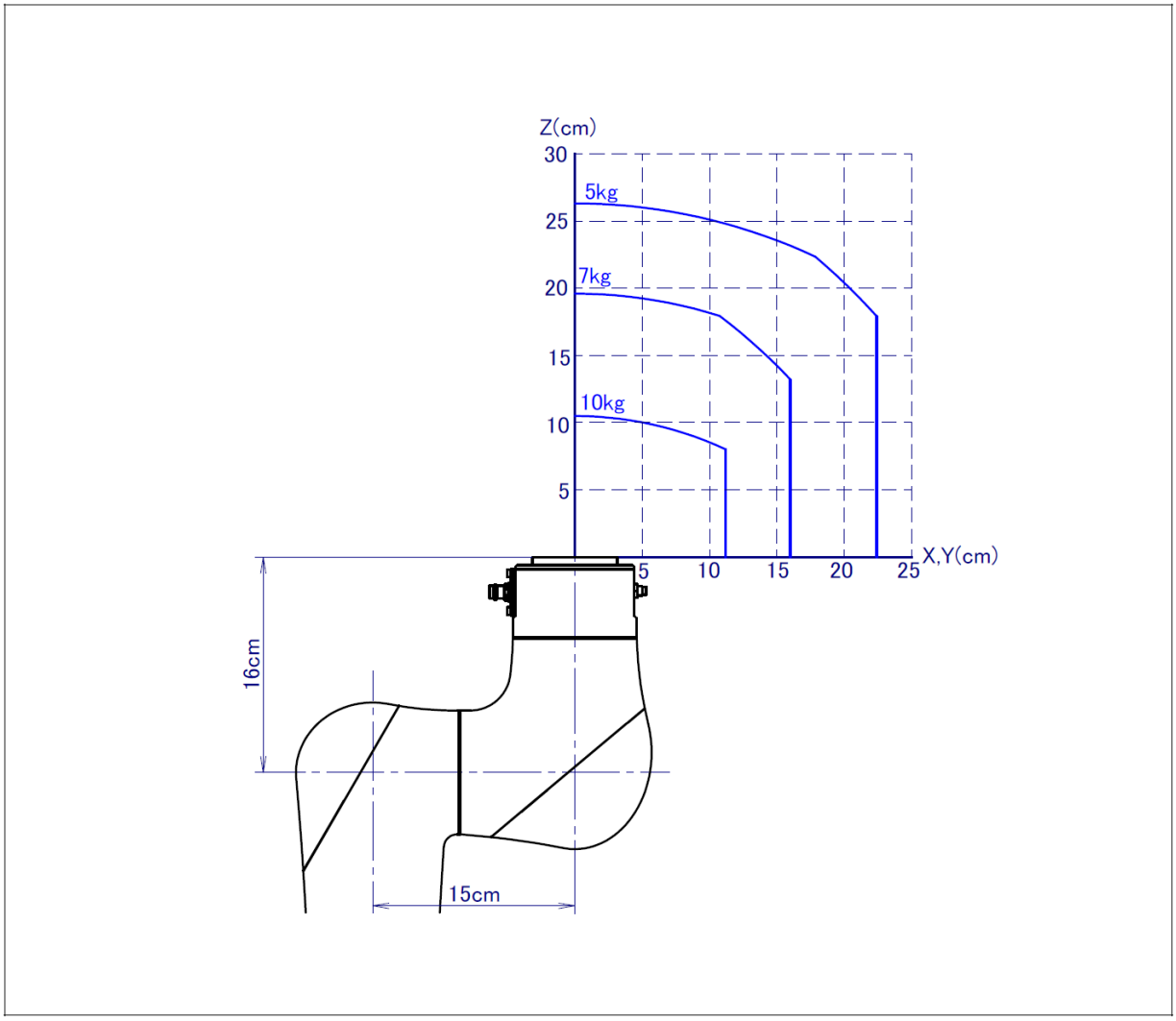


Fig. 15.1.4 (a) Wrist load diagram (CRX-10*i*A, CRX-10*i*A/L)

REVISION RECORD

Revision	Date	Modification history
12	2025. Oct	・Section 5.1.2, Add description for the character code for Taiwanese, German, French, Italian, and Spanish ・Section 6.3, Add description for the character code for Taiwanese, German, French, Italian, and Spanish ・Section 7.1.3, Add description for the character code for Taiwanese, German, French, Italian, and Spanish ・Section 7.5, Correct descriptions and add examples for JavaScript APIs ・Chapter 11, Add description for SSET command to set Tool I/F configuration
11	2025. Jan.13	・Section 5.1.2, Add description for the character code for Chinese ・Section 5.2.6.3, Add description to set Payload change distance, External force limit, and Safe I/O to disable each force limit using SSET command ・Section 5.2.7, Add description for PLGMTOP command ・Section 6.3, Add description for the character code for Chinese ・Section 7.1.3, Add description for the character code for Chinese ・Section 7.4.2, Add caution for inputType for textboxInteger ・Section 7.4.3, Add caution for inputType for textboxReal ・Section 7.5, Add and new JavaScript APIs and update the descriptions ・Section 8.3, Add caution for TP program and KAREL program of plugin instruction ・Section 8.3.7 Add description for IPL_SETPLD to setup Payload Information ・Section 8.3.8 Add description for IPL_GETPLD to get Payload Information ・Section 8.3.9 Add description for IPL_GETMACADDR to get MAC Address ・Section 8.5.1, Add caution in case if passing a REAL value as an argument of the icon instruction ・Section 10.1.2, Update the description for the update interval of Digital output signal ・Section 10.1.3, Update the description for the update interval of Digital input signal ・Section 11.4, Update the description for the update interval of Digital input/output signal ・Section 11.7, Update the description for HOST-412 and HOST-428 ・Chapter 12, Add description of Tool Change function for CRX series
10	2022. Jun.10	・Section 5.2.3, Update the description for RUNPROG/RUNPRGWT command ・Section 7.2.1, Update the description for the Perl environment ・Section 7.2.2, Add the caution for the case to install the plugin software that contains the updated JavaScript and CSS files ・Section 7.4.5, Add the note for listSelect & listSelectOption ・Section 8.2, Update the description for param/func attribute of init tag in icon definition XML file ・Section 8.3.6, Add the chapter of Setting User Socket Messaging by a KAREL Program ・Section 11.2, Add the caution for Digital output 1,2 and Table 11.2 (a) ・Section 11.8, Update description for HOST-412, 425 ・[ihmi_build] Fixed the build_ihmi.bat to always generate *eg,kn,ch.xml ・[ihmi_build] Fixed the issue that a build error occurs when executing the build_ihmi.bat placed in file directory that includes a space
09	2021. Dec.24	・Chapter 2, Add description for Plugin software development environment ・Chapter 4, Add description of Plugin specific alarm definition file ・Section 5.1, Add the caution for the license information feature and the linefeed code ・Section 5.1, Add the caution for the commands, PLGID, OPTION or REQUIRE ・Section 5.1.3, Add the caution for the commands, PLGID, OPTION or REQUIRE ・Section 5.1.4, Add the caution for the commands, PLGID, OPTION or REQUIRE ・Section 5.2.2, Add the caution for LOAD command about Multiple program selection ・Section 5.2.3, Add the caution for RUNPROG/RUNPRGWT command ・Section 5.2.5, Add the description for SYSC command ・Section 5.2.6, Add the description for SSET command ・Section 7.4.2, Add the figure for the software keyboard specified by inputType parameter ・Section 8.2, Add “cat_plugin” for the instruction category ・Section 8.3.5, Add the chapter of Use of Third Party Vision Interface ・Section 9.1, Add the description for SDTL-084 alarm ・Section 10.1.5, Add the description for Termination resistor for RS485 communication

Revision	Date	Modification history
		・Chapter 10, Add precautions for the tool connections to the EE connector ・Chapter 11, Add precautions for the tool connections to the EE connector ・Section 11.3, Add the description for automatic assignment of AI signals ・Section 11.4, Add the description for the system variables for serial communication ・Section 11.6.1, Update the description for TLIF_WRITE ・Section 11.8, Update the description for HOST-412 ・Chapter 12, Add description for Internal I/O terminal block conversion module ・Chapter 12, Add the caution for 24V power supply to the plugin device ・Chapter 13, Add changes on the procedures to setup the Eclipse development envrionment ・Chapter 14, Add description for the specification of FANUC Robot CRX series mechanical unit
08	2020. Nov.26	・Chapter 4, Add description of License information file ・Section 5.1, Add description of OPTION/REQUIRE commands ・Chapter 6, Update the implementation example of dedicated screen ・Section 8.2, Add description for <program> tag ・Section 8.3.3, Add description for Plugin specific alarm message ・Section 8.5, Correct the file path of js file that sampleapp.ishtm refers ・Section 8.6.1, Update the List of Error Code ・Section 9.1, Add description of error messages during plugin installation ・Chapter 10, Update the figure of Tool interface ・Section 10.1, Update the specification information for EE connector ・Section 11.1, Change the description of the caution for power suuply mode, add table 11.1 (b) ・Section 11.2, Change the description of the caution for power suuply mode ・Chapter 13, Add the chapter for Eclipse plugin development environment
07	2020. Jul.30	・Chapter 4, Add note about directory hierarchy in the installation package file, and add note about the image files ・Section 5.1.1, this entire section has been added. ・Section 5.2.4, Add description for settings system variable with DSET command ・Section 5.3, Add description for tool coordinate system setting ・Section 7.4.2, Add description for minimum and maximum value for textboxInteger ・Section 7.5, Add io_type of GI/GO, AI/AO ・Section 7.6.1, Add caution for <FanucMenu> tag ・Section 7.7, Add a new section for verification of created screen using ROBOGUIDE ・Section 8.2, Add description for <disp> tag ・Section 8.3.1, Add caution for the TP program with Payload instruction ・Section 8.3.2, Update TP instructions supported with Local registers ・Section 9.1, Add EE Connector specifications on the tool side ・Chapter 10, Add caution ・Section 10.1, Change the specification of system variable and add new system variable ・Section 10.3, Change the specification of system variable, and add caution ・Section 10.4, Change the specification of system variable and add/delete new system variable ・Section 10.5, Add description for settings system variables of Tool Interface Communication ・Section 10.6.1, Change the description for details ・Section 10.6.3, Add description. ・Section 10.6.4, Add description. ・Section 10.7, Add description and change sample program. ・Section 10.8, Add alarm list for Tool Interface Communication. ・Chapter 11, Add a chapter for Peripheral Device Interface on Robot Controller
06	2020. Apr.26	・Chapter 4, Add Chapter 9 and 10 to the table 4 (a) ・Chapter 5-8, Change the naming rules for TP/KAREL program ・Chapter 8, Add the section 8.3.2 (Local Registers in TP program) ・Chapter 9, Add Figure 9 (a) ・Section 9.1.1, Add description for use a protective diode ・Section 9.1.4, Change the electrical specification ・Section 9.1.5, Add baud rate specification ・Chapter 10, Newly added
05	2020. Mar.27	・Chapter 4-5, Multi-lanugage support for Plugin info. Max length of PLGID corrected ・Chapter 4, add some configuration file necessary in the installation package

Revision	Date	Modification history
		・Chapter 6, update the implementation example ・Chapter 4-8, Add file naming convention and some correction ・Section 7.5, Add description for ihmi_iovalrd ・Section 8.3.2, Add description for RI/RO (DI/DO on Tool Interface)
04	2020. Feb.23	・Remove description for Chinese language and icon type
03	2020. Jan.23	・Correct the format of CSV command and the chapter 5 description, Add the section 5.2.3, Some correction
02	2020. Jan	・Some correction
01	2019. Dec	・Initial creation