YRC1000 OPTIONS
INSTRUCTIONS
FOR PROGRAMMING PENDANT CUSTOMIZATION FUNCTION

Upon receipt of the product and prior to initial operation, read these instructions thoroughly, and retain for future reference.

MOTOMAN INSTRUCTIONS

MOTOMAN-□□□ INSTRUCTIONS
YRC1000 INSTRUCTIONS
YRC1000 OPERATOR’S MANUAL (GENERAL) (SUBJECT SPECIFIC)
YRC1000 MAINTENANCE MANUAL
YRC1000 ALARM CODES (MAJOR ALARMS) (MINOR ALARMS)

The YRC1000 operator’s manual above corresponds to specific usage. Be sure to use the appropriate manual.
The YRC1000 operator’s manual above consists of “GENERAL” and “SUBJECT SPECIFIC”.
The YRC1000 alarm codes above consists of “MAJOR ALARMS” and “MINOR ALARMS”.

Please have the following information available when contacting Yaskawa Customer Support:
- System
- Primary Application
- Software Version (Located on Programming Pendant by selecting: (Main Menu) - (System Info) - (Version))
- Robot Serial Number (Located on robot data plate)
- Robot Sales Order Number (Located on controller data plate)

Part Number: 179331-1CD
Revision: 1
DANGER

• This manual explains the programming pendant customization function of the YRC1000 system. Read this manual carefully and be sure to understand its contents before handling the YRC1000. Any matter, including operation, usage, measures, and an item to use, not described in this manual must be regarded as "prohibited" or "improper".

• General information related to safety are described in "Chapter 1. Safety" of the YRC1000 INSTRUCTIONS. To ensure correct and safe operation, carefully read "Chapter 1. Safety" of the YRC1000 INSTRUCTIONS.

CAUTION

• In some drawings in this manual, protective covers or shields are removed to show details. Make sure that all the covers or shields are installed in place before operating this product.

• YASKAWA is not responsible for incidents arising from unauthorized modification of its products. Unauthorized modification voids the product warranty.

NOTICE

• The drawings and photos in this manual are representative examples and differences may exist between them and the delivered product.

• YASKAWA may modify this model without notice when necessary due to product improvements, modifications, or changes in specifications. If such modification is made, the manual number will also be revised.

• If your copy of the manual is damaged or lost, contact a YASKAWA representative to order a new copy. The representatives are listed on the back cover. Be sure to tell the representative the manual number listed on the front cover.
Notes for Safe Operation

Read this manual carefully before installation, operation, maintenance, or inspection of the YRC1000.

In this manual, the Notes for Safe Operation are classified as “DANGER”, “WARNING”, “CAUTION”, or “NOTICE”.

**DANGER**
Indicates an imminently hazardous situation which, if not avoided, will result in death or serious injury. Safety Signs identified by the signal word DANGER should be used sparingly and only for those situations presenting the most serious hazards.

**WARNING**
Indicates a potentially hazardous situation which, if not avoided, will result in death or serious injury. Hazards identified by the signal word WARNING present a lesser degree of risk of injury or death than those identified by the signal word DANGER.

**CAUTION**
Indicates a hazardous situation, which if not avoided, could result in minor or moderate injury. It may also be used without the safety alert symbol as an alternative to “NOTICE”.

**NOTICE**
NOTICE is the preferred signal word to address practices not related to personal injury. The safety alert symbol should not be used with this signal word. As an alternative to “NOTICE”, the word “CAUTION” without the safety alert symbol may be used to indicate a message not related to personal injury.

Even items described as “CAUTION” may result in a serious accident in some situations. At any rate, be sure to follow these important items.

**NOTE**
To ensure safe and efficient operation at all times, be sure to follow all instructions, even if not designated as “DANGER”, “WARNING” and “CAUTION”.


Before operating the manipulator, make sure the servo power is turned OFF by performing the following operations. When the servo power is turned OFF, the SERVO ON LED on the programming pendant is turned OFF.

- Press the emergency stop buttons on the front door of the YRC1000, on the programming pendant, on the external control device, etc.
- Disconnect the safety plug of the safety fence. (when in the play mode or in the remote mode)

If operation of the manipulator cannot be stopped in an emergency, personal injury and/or equipment damage may result.

Before releasing the emergency stop, make sure to remove the obstacle or error caused the emergency stop, if any, and then turn the servo power ON.

Failure to observe this instruction may cause unintended movement of the manipulator, which may result in personal injury.

Observe the following precautions when performing a teaching operation within the manipulator's operating range:

- Be sure to perform lockout by putting a lockout device on the safety fence when going into the area enclosed by the safety fence. In addition, the operator of the teaching operation must display the sign that the operation is being performed so that no other person closes the safety fence.
- View the manipulator from the front whenever possible.
- Always follow the predetermined operating procedure.
- Always keep in mind emergency response measures against the manipulator’s unexpected movement toward a person.
- Ensure a safe place to retreat in case of emergency.

Failure to observe this instruction may cause improper or unintended movement of the manipulator, which may result in personal injury.

Confirm that no person is present in the manipulator's operating range and that the operator is in a safe location before:

- Turning ON the YRC1000 power
- Moving the manipulator by using the programming pendant
- Running the system in the check mode
- Performing automatic operations

Personal injury may result if a person enters the manipulator's operating range during operation. Immediately press an emergency stop button whenever there is a problem. The emergency stop buttons are located on the front panel of the YRC1000 and on the right of the programming pendant.

Read and understand the Explanation of the Warning Labels before operating the manipulator.
Definition of Terms Used Often in This Manual

The MOTOMAN is the YASKAWA industrial robot product.

The MOTOMAN usually consists of the manipulator, the controller, the programming pendant, and supply cables.

In this manual, the equipment is designated as follows.

<table>
<thead>
<tr>
<th>Equipment</th>
<th>Manual Designation</th>
</tr>
</thead>
<tbody>
<tr>
<td>YRC1000 Controller</td>
<td>YRC1000</td>
</tr>
<tr>
<td>YRC1000 Programming Pendant</td>
<td>Programming Pendant</td>
</tr>
<tr>
<td>Cable between the manipulator and the controller</td>
<td>Manipulator cable</td>
</tr>
</tbody>
</table>

WARNING

• Perform the following inspection procedures prior to conducting manipulator teaching. If there is any problem, immediately take necessary steps to solve it, such as maintenance and repair.
  – Check for a problem in manipulator movement.
  – Check for damage to insulation and sheathing of external wires.
• Always return the programming pendant to the hook on the YRC1000 cabinet after use.

If the programming pendant is left unattended on the manipulator, on a fixture, or on the floor, etc., the Enable Switch may be activated due to surface irregularities of where it is left, and the servo power may be turned ON. In addition, in case the operation of the manipulator starts, the manipulator or the tool may hit the programming pendant left unattended, which may result in personal injury and/or equipment damage.
Descriptions of the programming pendant keys, buttons, and displays are shown as follows:

<table>
<thead>
<tr>
<th>Equipment</th>
<th>Manual Designation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programming Pendant</td>
<td><strong>Character Keys /Symbol Keys</strong> The keys which have characters or symbols printed on them are denoted with []. ex. [ENTER]</td>
</tr>
<tr>
<td>Axis Keys /Numeric Keys</td>
<td>[Axis Key] and [Numeric Key] are generic names for the keys for axis operation and number input.</td>
</tr>
<tr>
<td>Keys pressed simultaneously</td>
<td>When two keys are to be pressed simultaneously, the keys are shown with a “+” sign between them, ex. [SHIFT]+[COORD]</td>
</tr>
<tr>
<td>Displays</td>
<td>The menu displayed in the programming pendant is denoted with { }. ex. {JOB}</td>
</tr>
</tbody>
</table>

**Description of the Operation Procedure**

In the explanation of the operation procedure, the expression "Select • • • " means that the cursor is moved to the object item and [SELECT] is pressed, or that the item is directly selected by touching the screen.

**Registered Trademark**

In this manual, names of companies, corporations, or products are trademarks, registered trademarks, or brand names for each company or corporation. The indications of (R) and ™ are omitted.
Contents

1 Programming Pendant Customization Function ............................................................................. 1-1
2 Function Outline.............................................................................................................................. 2-1
3 Parameter Setting........................................................................................................................... 3-1
4 API for Programming Pendant Customization Function (MotoConnect API)................................. 4-1
5 Interface with Programming Pendant Main Software....................................................................... 5-1
6 Startup Method of the User-Customization Application .................................................................. 6-1
   6.1 Start from the Main Menu ....................................................................................................... 6-1
   6.2 Auto Run at Power ON ......................................................................................................... 6-1
   6.3 One Touch Run with a Button.............................................................................................. 6-2
7 Installing Applications ...................................................................................................................... 7-1
   7.1 Create the Installing Data ..................................................................................................... 7-1
      7.1.1 Zip File......................................................................................................................... 7-1
      7.1.2 INI File ....................................................................................................................... 7-1
   7.2 Creating Installer.................................................................................................................... 7-4
      7.2.1 Necessary File............................................................................................................... 7-4
      7.2.2 Describing “pp_install.ini” ....................................................................................... 7-5
   7.3 Installation .............................................................................................................................. 7-6
8 Guideline for Creating Application .................................................................................................. 8-1
   8.1 Priority of the Application ..................................................................................................... 8-1
   8.2 CPU Utilization by Application ........................................................................................... 8-1
   8.3 Successive Access to the YRC1000 ..................................................................................... 8-1
   8.4 Files Created by Application .............................................................................................. 8-2
   8.5 Response to Alarm ............................................................................................................... 8-2
   8.6 Communication between Programming Pendant Main Software and User-Customized Application ................................................................................................................................. 8-2
   8.7 Application Test .................................................................................................................... 8-3
      8.7.1 Standard Operation Test .............................................................................................. 8-3
      8.7.2 Response to Alarm ....................................................................................................... 8-4
      8.7.3 Continuous Run........................................................................................................... 8-4
      8.7.4 Window Switching ...................................................................................................... 8-4
      8.7.5 Cycle Time Check ...................................................................................................... 8-4
8.8 Other.............................................................................................................................. 8-4

9 Development of Application in VC++ .................................................................................. 9-1
9.1 Preparation of the PC for Developing Application.............................................................. 9-1
9.2 Development of Application............................................................................................... 9-4
  9.2.1 Controller Communication Library ........................................................................ 9-4
  9.2.2 Initialization/Closing Process of Library ................................................................... 9-6
  9.2.3 Interface with Programming Pendant Main Software ............................................. 9-7
9.3 Application Debugging...................................................................................................... 9-11
  9.3.1 Parameter Setting..................................................................................................... 9-11
  9.3.2 Debugging.............................................................................................................. 9-11
  9.3.3 Operation Check...................................................................................................... 9-13

10 Development of Application in C#..................................................................................... 10-1
10.1 Preparation of the PC for Developing Application.......................................................... 10-1
10.2 Development of Application............................................................................................. 10-2
  10.2.1 Creation of Software (PC for Development)....................................................... 10-2
  10.2.2 Controller Communication Library ...................................................................... 10-2
  10.2.3 Initialization Process of Library .......................................................................... 10-3
  10.2.4 Interface with Programming Pendant Main Software ....................................... 10-5
10.3 Application Debugging..................................................................................................... 10-9
  10.3.1 Parameter Setting............................................................................................... 10-9
  10.3.2 Debugging........................................................................................................... 10-9

11 Interface Specification with Programming Pendant Main Software.................................... 11-1
11.1 Command from the User Customized Application to Programming Pendant Main
     Software............................................................................................................................. 11-1
  11.1.1 Completion of Generation......................................................................................... 11-1
  11.1.2 Closing of Application ............................................................................................ 11-1
  11.1.3 Keymask................................................................................................................ 11-2
  11.1.4 Text Input Function Start ....................................................................................... 11-3
  11.1.5 Switching Control................................................................................................... 11-3
  11.1.6 Setting of Notifying Information............................................................................ 11-4
  11.1.7 Information Request............................................................................................... 11-4
  11.1.8 Version Information Acquisition............................................................................. 11-5
11.2 Command from the Programming Pendant Main Software to the User-Customized Application ....................................................................................................................... 11-6

11.2.1 Input Character .................................................................................................. 11-6

11.2.2 Request of Control System Switching ............................................................. 11-6

11.2.3 Status Information ........................................................................................... 11-7

11.2.4 Alarm ................................................................................................................ 11-11

11.2.5 Message ............................................................................................................ 11-12

11.2.6 Language ......................................................................................................... 11-12

11.2.7 Version ............................................................................................................. 11-12
1 Programming Pendant Customization Function

With this function, user-customized windows can be created by into the YRC1000 programming pendant, incorporating a customized application (WindowsCE application software) developed by the user (system integrator).

Also, this function provides APIs for the customized function (function group) which are available at the customized application (WindowsCE application software) developed by the user (system integrator).

With these APIs, from the user-developing applications, acquiring of the YRC1000 status information or writing/reading of the data are possible.

The size of the windows that can be created with this function are full-screen sized window only.

Indication of the user-customized window when turning ON the power supply of the YRC1000 or the standard window when it is online status can be chosen.
2 Function Outline

This function enables creating windows for the user-customized (a separate exe file from the YASKAWA standard application) and incorporating them into the programming. To execute this function, the following three functions are provided to realize this function.

1. The API that can access (Read/Write) the data in the YRC1000.
2. Data sending/receiving function (communication function between processes) between the YASKAWA standard application and the application for the user-customized window.
3. Function to install the user-customized application to the YRC1000 and register it to the menu of the system.
## 3 Parameter Setting

<table>
<thead>
<tr>
<th>No.</th>
<th>Function</th>
<th>Setting Value</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>S2C262</td>
<td>Programming Pendant customized function (applied during start-up)</td>
<td>0: Invalid</td>
<td>0 (Invalid)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: Valid</td>
<td></td>
</tr>
<tr>
<td>S2C531</td>
<td>Automatic start of user-customized application function in the programming when start-up.</td>
<td>0: Invalid</td>
<td>0 (Invalid)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: Valid</td>
<td></td>
</tr>
<tr>
<td>S2C533</td>
<td>Debugging function (applied during start-up)</td>
<td>0: Invalid</td>
<td>0 (Invalid)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: Valid</td>
<td></td>
</tr>
<tr>
<td>S2C541</td>
<td>Writing variables and I/O during play mode.</td>
<td>0: Permitted</td>
<td>1 (Not permitted)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: Not permitted</td>
<td></td>
</tr>
<tr>
<td>S2C542</td>
<td>Writing variables and I/O during edit-lock status.</td>
<td>0: Invalid</td>
<td>1 (Not permitted)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: Valid</td>
<td></td>
</tr>
<tr>
<td>S4C1057</td>
<td>Specification of WM_COPYDATA message communication method (applied during start-up)</td>
<td>0: Message notice by SendMessage() Not 0: Message notice by SendMessageTimeout() Timeout period is set as the specified value multiplied by 100 (msec)</td>
<td>30 (Timeout in 3 sec)</td>
</tr>
</tbody>
</table>

**NOTE**
Writing of variables and I/Os become valid when "0" is set to S2C541, however, it may influence the manipulator's cycle time.

**SUPPLEMENT**
The term "during edit-lock status" in the explanation of parameter No. S2C542 above means following statuses.
- During alarm occurrence
- During external memory device operation
- During the execution of data transmission function
- During specific input “EDIT_LOCK(#40064)” ON
4 API for Programming Pendant Customization Function
(MotoConnect API)

For reading/writing data to/from the user-customized application to/from the YRC1000, APIs are prepared.

There are four main types of APIs as follows. For the details of each API, refer to “YRC1000 OPTIONS INSTRUCTIONS REFERENCE MANUAL FOR PROGRAMMING PENDANT CUSTOMIZATION FUNCTION” (Manual No.: HW1484233, HW1484235).

<table>
<thead>
<tr>
<th>System</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>System monitor</td>
<td>Acquires the YRC1000 data. APIs for data reading.</td>
</tr>
<tr>
<td>System control</td>
<td>Operates the YRC1000. APIs for data writing are included.</td>
</tr>
<tr>
<td>Step movement</td>
<td>API for performing the equivalent of step movements.</td>
</tr>
<tr>
<td>File transmission</td>
<td>Transfer files with the function equivalent to External Memory Function.</td>
</tr>
</tbody>
</table>
5 Interface with Programming Pendant Main Software

The interface to perform screen switching and data transmission between the YASKAWA standard applications and the user-customized applications are prepared.

For the details of the interface with the YASKAWA standard window applications, refer to “YRC1000 OPTIONS INSTRUCTIONS REFERENCE MANUAL FOR PROGRAMMING PENDANT CUSTOMIZATION FUNCTION” (Manual No.: HW1484233, HW1484235).
6 Startup Method of the User-Customization Application

There are three ways of starting method of the user-customized application as follows.

- Start from the main menu.
- Auto run at power ON.
- One touch run with a button.

Customized applications cannot be started concurrently.

In order to start another application while one is already active, close the active application, and then start another application.

6.1 Start from the Main Menu

Like indicating the standard window, this application can be started from the main menu. Where to register on the main menu or which notation to indicate can be specified on the application basis.

6.2 Auto Run at Power ON

The user-customized application can be started same time when the power supply of the YRC1000 is turned ON. Set this auto run function when installing the application.

Only one user-customized application can be automatically started in the YRC1000 system and this application can be started also from the main menu.

Before using this function, be sure to set “1(valid)” to the programming pendant customization application (S2C531=1) Auto Run function of at power ON.
6.3 One Touch Run with a Button

To start the user-customized application by a single operation, start-up function of this application can be allocated to a button at the lower right of the programming pendant window. Allocate the application to the button when installing it.

Only one application can be allocated to the button for One Touch Run function and this application can be started also from the main menu.

When putting a name to the button, the name will be the same one as the name of the sub menu. Up to fourteen characters can be used as its name and it may not be indicated properly if it exceeded.
7 Installing Applications

7.1 Create the Installing Data

7.1.1 Zip File

Zip applications (*.exe), accessory files (dll, data files, etc.), and accessory folders with its application's name. And place the application (*.exe) on the root of zip file.

For example, if the name of the created application is AmpleApp.exe, and that of the library used in the application is SampleApp.dll and zip them. The name of the zip file should be SampleApp.zip.

For zipping, use the zipping tool regularly used on the desk top.

7.1.2 INI File

To an INI file, please describe the information necessary for installing the application. For the name of the INI file, put the same name as the zip file.

Following is an example format of the SampleApp’s INI file (SampleApp.ini).

```
[APPLICATION_INFO]
APP_NAME = SampleApp
APP_VER = 1.00-00-YE
SYS_VER = N50.00-00
FILE_SIZE = 53
FOLDER_NAME = SampleNx
APP_TYPE = APP_TYPE_OVERLAPPED_EXE
SYS_STATUS = 0x01
AUTO_RUN = FALSE

[MENU_INFO]
TOPMENU_ID = 12
SUBMENU_NAME_JAPANESE = “サンプル アプリ”
SUBMENU_NAME_ENGLISH = “SampleApp”
SUBMENU_ICON = SampleApp.ico
```
### 7 Installing Applications

#### 7.1 Create the Installing Data

<table>
<thead>
<tr>
<th>[APPLICATION_INFO]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>APP_NAME</strong></td>
</tr>
<tr>
<td>Name of the application. It is same as application root folder name and exe.file name (extension is excluded). This name functions as the identifier of this application. Restrictions: Within 12 or less of single-byte characters.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>APP_VER</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>(Application Version) Version of the application itself, and it is described as “<strong>.</strong>-<strong>-developer name”. *: one-byte character Developer name: abbreviated name of the company which developed the application. Ex.1.00-00-YE Restrictions: Use 11 of single-byte characters like “</strong>.<strong>-</strong>-**”.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>SYS_VER</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>(System Version) To keep compatibility with the NX100, the version is not checked. Specify “SYS_VER=NS0.00-00”.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>FILE_SIZE</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>(File Size) Describes the total size of decompressed file group. (including files in the sub folders) It is used to check the disk capacity when installing the file. Unit: KB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>FOLDER_NAME</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>(Installation Folder Name) Name of the folder created in the programming pendant when this application is installed. The application is decompressed and saved under DiskOnChip/PP_APP “folder name” when installed. If the name is not specified, the application name is adopted instead.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>APP_TYPE</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>(Application Type) Specify “APP_TYPE_OVERLAPPED_EXE”. In the programming pendant customization function, only the application with which a full-screen sized window type can be created.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>SYS_STATUS</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>(Display condition on the basis of system state) Indication of the programming pendant customization function window can be restricted on the basis of its system status. When a certain status cannot be displayed, the menu of the status will not be displayed on the main menu. 0x01: Concealed when edit is locked 0x02: Concealed when alarming 0x04: Concealed in the teach mode 0x08: Concealed in the play mode 0x00: Displayed in the operation mode or higher 0x40: Displayed in the edit mode or higher 0x80: Displayed in the management mode or higher Note: Specified in hex. notation. Ex.SYS_STATUS=0x4a – Displayed as a menu on the main menu in the edit or management mode. – Concealed from the main menu in the play mode or when alarming.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>AUTO_RUN</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>(Auto Run Function Enabled/Disabled) Auto Run function valid = “TRUE” Auto Run function valid = “FALSE” If a certain application is to be started up automatically, set “TRUE” to Auto Run function when installing the application. Note: It is usually set “TRUE” when the operation from the programming pendant is not necessary.</td>
</tr>
</tbody>
</table>
## Installing Applications

### 7.1 Create the Installing Data

### [MENU_INFO]

<table>
<thead>
<tr>
<th>TOPMENU_ID</th>
<th>Top menu to add the application.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top menu</td>
<td>Setting value</td>
</tr>
<tr>
<td>JOB</td>
<td>1</td>
</tr>
<tr>
<td>WORK CONDITION</td>
<td>2</td>
</tr>
<tr>
<td>VARIABLE</td>
<td>3</td>
</tr>
<tr>
<td>IN/OUT</td>
<td>4</td>
</tr>
<tr>
<td>ROBOT</td>
<td>5</td>
</tr>
<tr>
<td>SYSTEM INFO.</td>
<td>6</td>
</tr>
<tr>
<td>EX.MEMORY</td>
<td>7</td>
</tr>
<tr>
<td>PARAMETER</td>
<td>8</td>
</tr>
<tr>
<td>SETUP</td>
<td>9</td>
</tr>
<tr>
<td>SAFETY FUNC.</td>
<td>10</td>
</tr>
<tr>
<td>PM</td>
<td>11</td>
</tr>
<tr>
<td>OPTION</td>
<td>12</td>
</tr>
<tr>
<td>MANUFACTURE RESERVE</td>
<td>13-20</td>
</tr>
<tr>
<td>APPLICATION</td>
<td>21</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SUBMENU_NAME_*</th>
<th>(Submenu Name)</th>
</tr>
</thead>
<tbody>
<tr>
<td>{Submenu Name}</td>
<td>Specify a character string for indication in the sub menu.</td>
</tr>
<tr>
<td>Note: Up to 20 single-byte characters.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Up to 14 single-byte characters for the name of OneTouch Run function button.</td>
</tr>
<tr>
<td>SUBMENU_NAME_JAPANESE</td>
<td>“12345678901234567890”</td>
</tr>
<tr>
<td>SUBMENU_NAME_ENGLISH</td>
<td>“BBB”</td>
</tr>
<tr>
<td>SUBMENU_NAME_GERMAN</td>
<td>“CCC”</td>
</tr>
<tr>
<td>SUBMENU_NAME_SWEDISH</td>
<td>“DDD”</td>
</tr>
<tr>
<td>SUBMENU_NAME_FRENCH</td>
<td>“EEE”</td>
</tr>
<tr>
<td>SUBMENU_NAME_FINNISH</td>
<td>“FFF”</td>
</tr>
<tr>
<td>SUBMENU_NAME_ITALIAN</td>
<td>“GGG”</td>
</tr>
<tr>
<td>SUBMENU_NAME_SPANISH</td>
<td>“HHH”</td>
</tr>
<tr>
<td>SUBMENU_NAME_HANGLE</td>
<td>“III”</td>
</tr>
<tr>
<td>SUBMENU_NAME_CHINESE</td>
<td>“JJJ”</td>
</tr>
<tr>
<td>SUBMENU_NAME_TAIWANESE</td>
<td>“KKK”</td>
</tr>
<tr>
<td>SUBMENU_NAME_CZECH</td>
<td>“LLL”</td>
</tr>
<tr>
<td>SUBMENU_NAME_POLISH</td>
<td>“MMM”</td>
</tr>
<tr>
<td>SUBMENU_NAME_RUSSIAN</td>
<td>“NNN”</td>
</tr>
<tr>
<td>SUBMENU_NAME_THAI</td>
<td>“”</td>
</tr>
<tr>
<td>SUBMENU_NAME_THAI</td>
<td>“”</td>
</tr>
<tr>
<td>SUBMENU_NAME_TAIWANESE</td>
<td>“”</td>
</tr>
<tr>
<td>SUBMENU_NAME_PORTUGUESE</td>
<td>“”</td>
</tr>
<tr>
<td>SUBMENU_NAME_TURKISH</td>
<td>“”</td>
</tr>
<tr>
<td>SUBMENU_NAME_SLOVENIAN</td>
<td>“”</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SUBMENU_ICON</th>
<th>(Submenu Icon) (for future expansion)</th>
</tr>
</thead>
<tbody>
<tr>
<td>From the icon file, specify an icon to be indicated in the sub menu.</td>
<td></td>
</tr>
</tbody>
</table>
7 Installing Applications
7.2 Creating Installer

7.2 Creating Installer

7.2.1 Necessary File

Create PP_APP folder on the USB memory or SD for version up root folder.

Example of PP_APP folder

<table>
<thead>
<tr>
<th>File name</th>
<th>Function</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>YE_AppInstall.exe</td>
<td>Application installer</td>
<td>Manufacturer-supplied</td>
</tr>
<tr>
<td>pp_install.ini</td>
<td>Configuration file of the installer. Specifies the default selection of the added application when startup.</td>
<td>Manufacturer-supplied</td>
</tr>
<tr>
<td>Unzip.dll</td>
<td>Zip file decompresses</td>
<td>Manufacturer-supplied</td>
</tr>
<tr>
<td>*.zip</td>
<td>Application compressed file</td>
<td></td>
</tr>
<tr>
<td>*.ini</td>
<td>Application configuration information file</td>
<td></td>
</tr>
</tbody>
</table>

Save the following folders in the PP_APP folder.
7 Installing Applications
7.2 Creating Installer

7.2.2 Describing “pp_install.ini”

For the applications described in pp_install.ini, “add” is automatically set to “Operation” when the installer is started-up.

And for the applications not described in pp_install.ini, they can be installed by setting “add” before installation.

APP_NUM: Specify the numbers of applications to be set
APP_NAMEEN: Specify the name of applications to be set

[Default Setting]
APP_NUM=3
AMM_NAME1=APITest1
AMM_NAME2=APITest2
AMM_NAME3=OperationArea
7 Installing Applications
7.3 Installation

Press [INTERLOCK] + [7]+ [SELECT] and turn ON the control power supply of the YRC1000.

The user-customized application installation window opens if the user-customized application installing environment is equipped to the SD or USB memory.

1. Check that “Add” is set for “Operation” of an applications to install.

2. Press (Option) when installing the application which uses the following functions.
   – Auto Run function or
   – One Touch Run function
3. Optional setting window appears. Set Auto Run Application and One Touch Run Application. The already-set applications are indicated if any.

[Auto Run Application]
Select an application which specifies the Auto Run function at power-ON. In the list, an application which specifies “AUTO_RUN=TRUE” with the “application name.ini” file is indicated. In case canceling this section, select a blank line.

[One Touch Run Application]
Select an application which specifies the One Touch Run with a Button function. In the list, already-registered applications and those applications to which “Add” is set to “Operation” on the window are indicated. In case canceling this selection, select a blank line.
4. Press {Return} after the above setting. The window returns to the previous one. The indication varies depending on the setting as follows.

- “A” is indicated to the application specified as the Auto Run application at power ON.
- “O” is indicated to the application specified as the One Touch Run with a button.
- “*” is indicated to the application to which both functions are specified.

5. After all settings are completed, press {Execute}. Installation of the applications starts. Do not turn OFF the power supply before the installation is done because the files are being developed to both YRC1000 and programming pendant.
6. When the installation is completed, the “Operation” section is cleared and versions and sizes of each application are indicated. (This state is granted as completion of installation.)

<table>
<thead>
<tr>
<th>No</th>
<th>Operation</th>
<th>Name</th>
<th>Version</th>
<th>Size</th>
<th>Version</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>noRun</td>
<td></td>
<td>1.00-00-VE</td>
<td>250</td>
<td>1.00-00-VE</td>
<td>250</td>
</tr>
<tr>
<td>002</td>
<td>APTtest</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>003</td>
<td>APTtest01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The installed application data is controlled by both YRC1000 and programming pendant. Therefore, for the data consistency, within the installed data, the inconsistent data between the YRC1000 and the programming pendant are deleted next time when the data is newly installed.

In case the programming pendant is replaced after the user-customized application is installed, the inconsistency of the data may occur between the YRC1000 and the programming pendant.
8 Guideline for Creating Application

The created user-customized application is incorporated to the manipulator and functions as a part of the manipulator system. Therefore, it may interfere the system if it is not processed properly. To avoid this, a guideline for creating the user-customized application is provided. Follow this guideline when creating user-customized applications.

8.1 Priority of the Application

For the system stability, it is recommended to set “BELOW_NORMAL” to the thread priority of the user-customized application. However, it is prohibited to set higher than “NORMAL” to the thread priority since it may influence a lot to the YASKAWA standard programming pendant window.

In case the higher priority than “NORMAL” is set to the application, the YASKAWA standard programming pendant window does not work properly and communication faults (PP communication fault) with the YRC1000 may occur.

8.2 CPU Utilization by Application

It is prohibited to execute a process that the user-customized application continuously occupies the CPU. To avoid this to happen, create a software in which the user-customized application does not occupy the CPU for more than 500sec (within 300sec is recommended).

In case a process is execute which the created application continuously occupies the YRC1000, the YASKAWA standard programming pendant window does not work properly and communication faults (PP communication fault) with the YRC1000 may occur.

8.3 Successive Access to the YRC1000

It is prohibited to execute a process that the user-customized application continuously make access to the YRC1000. (Consecutive writing of the variables, for example.) The application is recommended to make a call to the API at intervals of more than 50sec (100msec is strongly recommended).

In case a process is execute which the created application continuously access to YRC1000, processing operation in the YRC1000 may be influenced as follows.

- Operations during the teach mode may slow down.
- Cycle times when play back operation may slow down.
8.4 Files Created by Application

Save the file created by the application to the SD (Storage Card) in the programming pendant, to the USB memory or to the root folder (on the memory). Saving it to the flash memory (DiskOnChip) in the programming pendant, including to the temporary file, is prohibited.

Since necessary files for the robot controller system is saved in the DiskOnChip, the YASKAWA standard programming pendant window does not work properly or it won’t start up in case those files are influenced.

8.5 Response to Alarm

An alarm occurs when an error is found in the robot controller system. When an alarm occurs while the user-customized application window is displayed, it is reported from the YASKAWA standard window to the user-customized application.

To deal with this error, please prepare methods at the user-customized application side.

Create applications when an alarm occurs as follows.

- Create an application which closes the user-customized application and indicates an alarm window on the pendant YASKAWA standard window to notify the user that it is alarming.
- Create an application which creates methods to notify the user that an alarm is occurring in the user-customized application.
- Other than above.

8.6 Communication between Programming Pendant Main Software and User-Customized Application

Communications between the application in the programming pendant main software and the user-customized application are performed by transmitting WM_COPYDATA by using SendMessage() (SendMessageTimeout() in YAS1.38-00 or later). Thus, create the software in the way that the reception process of WM_COPYDATA does not occupy the CPU for 3 seconds or more.

If the reception process of WM_COPYDATA occupies the CPU, processes in the programming pendant main software may not be completed normally, and a communication error with the controller (PP communication error) may occur. In YAS1.38-00 or later, as a default setting, if the reception process of WM_COPYDATA occupies the CPU for 3 seconds or more, an error message appears and the user-customized application is force-quit. Since a force-quit may be regarded as unauthorized access to the data pointer, create the software in the way that the reception process of WM_COPYDATA does not occupy the CPU for 3 seconds or more.
8.7 Application Test

Carry out sufficient test to the created user-customized applications.

8.7.1 Standard Operation Test

As tests for the application’s standard operations, check the occurrence of errors or alarms while the created application is running. By conducting theses tests, influences by the user-customized application to the robot controller system and to the YASKAWA standard window can be confirmed.

The definition of the standard operations of the created applications are as follows.

1. Change the mode to the teach mode.
2. Create a new job.
3. After moving by the job operation, teach the move instruction. (MOVJ VJ=10 (or around 10))
4. Insert JUMP*LABEL to the end of the job.
5. Insert *LABEL to the head of the job.
6. Move the “First step” by pressing [FWD] to move.
7. Change the mode to the play mode.
8. Indicate the cycle time.
9. Indicate the job stack.
10. Press {Start} and execute playback operation of the Job.

Input B variables by following the procedures below.
(For inputting I, D, and R variables procedures, substitute the same procedures.)

1. Select {VARIABLE} - {BYTE VARIABLE} on the main menu.
2. Change the valuable on B variable window. Input 0, 128, 255, 256 or 1024. When inputting 256 or 1024, check that the range check works properly.
8.7.2 Response to Alarm

To confirm the alarming condition when indicating the user-created application, conduct tests to the following two cases. And check if the alarm is properly processed as defined at the user-created application side.

- 24V power supply error: Start an alarm by disconnecting AIF board.
- Alarm during play back operation: Intentionally exceed the limit by a move instruction in which the P variable (cartesian value) is used.

8.7.3 Continuous Run

As for the Jobs for continuous run tests, prepare Jobs for programming pendant continuous test including layer check of variety of calculation, R1,S1 and call job. Execute continuous run of these Jobs and check the occurrence of alarms or errors.

8.7.4 Window Switching

Test if switching of the window from the YASKAWA standard window to the user-customized application window is possible.

For the user-customized application, keep indicating the window to call up the API while test is executed. And check if those windows alternate properly.

8.7.5 Cycle Time Check

Prepare a cycle time checking Job. Before the test, record the cycle time (in the status that any APIs are not executed in the user-customized application), then check the cycle time after API is executed in the application. There will be an error for several segments time to the cycle time due to the Job start time, but it can be ignored.

8.8 Other

Other than mentioned above, it prohibited to execute any processes than may influence the whole robot controller system or the YASKAWA standard programming pendant window.
9 Development of Application in VC++

9.1 Preparation of the PC for Developing Application

   - An update is available for Visual Studio 2008 SP1 that adds support for Windows Embedded Compact 7
     https://support.microsoft.com/en-us/kb/2483802
   - Windows Embedded Compact 7 ATL Update for Visual Studio 2008 SP1
   - MSDN Library for Visual Studio 2008 SP1
5. Windows Embedded Compact 7 Update: Download the latest version from official Microsoft download center and install.
6. SDK: Install manufacturer-provided SDK for the programming pendant.
7. Visual Studio 2008 Environment
   (1) From the menu bar, select {Tools} - “Options”.
   (2) In the “Options” window, select “Projects and Solutions” - “VC++ Directories”.
   (3) Select the target SDK from the “Platform”, and then select “Include files”.
   (4) Add a directory.

C:\Program Files\Microsoft Visual Studio 9.0\VC\ce7\atlmfc\include
C:\Program Files\Microsoft Visual Studio 9.0\VC\ce7\include
C:\WINCE700\public\COMMON\sdk\inc
C:\WINCE700\public\COMMON\oak\inc
9 Development of Application in VC++

9.1 Preparation of the PC for Developing Application

(5) Select the target SDK from the “Platform”, and then select “Library files”.

(6) Add a directory.

C:\WINCE700\public\COMMON\oak\lib\armv7\retail

(7) In the “Options” window, select “Device Tools” - “Devices”.

(8) Under “Show devices for platform”, select the target SDK, and then select (Properties).
9 Development of Application in VC++

9.1 Preparation of the PC for Developing Application

(9) In the “Device Properties” window, select {Configure} under “Transport”.

![Device Properties Window](image)

(10) In the “Configure TCP/IP Transport” window, check the check box for “Use specific IP address” under “Device IP address”.

(11) Enter the IP address of the programming pendant.

The default IP address for the programming pendant is: 10.0.0.4

8. IP address setting for PC for developing application.

- Set the IP address, PC for developing application as follows.

  IP address: 10.0.0.3
  Subnet mask: 255.255.255.0
9 Development of Application in VC++

9.2 Development of Application

Create a new project in Visual Studio 2008.

In the “Platforms” window, select “YRC1000_PENDANT(ARMV41)” for the SDK.

9.2.1 Controller Communication Library

1. Add #include “YppCustomizeAPI.h” to the source in which API is to be used.

2. From the MFC settings, select the storage folder for the controller communication library to be used.
   - Use_MFC_in_a_Static_Library
   - Use_MFC_in_a_Shared_DLL

3. Copy the MotoConnect folder to the application folder to be created.

   ---- SampleApp
   |---- MotoConnect
   |   |---- YppCustomizeAPI.h
   |   |---- ARMV4IDbg MotoConnect.lib ※debug version
   |   |---- ARMV4IRel MotoConnect.lib ※release version
9 Development of Application in VC++
9.2 Development of Application

4. Add “MotoConnect” to “C/C++” - “Additional Include Directories” in the “Configuration Project”.

5. Add “Ws2.lib” and “MotoConnect/ARMV4IRel/MotoConnect.lib” to “Linker” - “Input” - “Additional Dependencies” in the “Configuration Project”.

6. Set the Debug mode same as the above.
9.2.2 Initialization/Closing Process of Library

1. Call YppConnectOpen() when the application is started up.
   
   **LONG YppConnectOpen()**
   
   Argument: None (not required)
   
   Return value: OK(=0) Normal termination / NG(=-1) Abnormal termination

2. Call YppConnectClose() when the application is closed.
   
   **LONG YppConnectClose()**
   
   Argument: None (not required)
   
   Return value: OK(=0) Normal termination / NG(=-1) Abnormal termination
9.2.3 Interface with Programming Pendant Main Software

For sending and receiving data with the programming pendant main software, the user-customized application employs WM_COPYDATA for the communication between APIs. Following are the important processing operations.

**Initializing Process <1>**

The following four arguments are delivered from the pendant main software, when the customized application is started up.

- Language code (/L)
- Window handle of the programming pendant main software (/H)
- IP address of the YRC1000 (/CIP)
- IP address of the programming pendant (/PIP)

For the user-customized application, obtain the language code and the window handle of the programming pendant main software by CWinApp::InitInstance() and save them.

**[Code example]**

```cpp
class CSampleApp::public CWinApp
{
public:
  HWND m_hSendWnd;
  DWORD m_dwLangID;
}

BOOL CSampleApp::InitInstance()
{
  . . .
  
  // Analyze the startup parameter
  CString strLang, strHwnd;
  int cnt1, cnt2;
  
  CString strCmd( m_lpszCmdLine );
  cnt1 = strCmd.Find( L"/L" ) + 2;
  if (cnt1 - 2 != EOF)
  {
    cnt2 = strCmd.Find( L" ", cnt1 );
    if( cnt2 == EOF)
      strLang = strCmd.Mid( cnt1 );
    else
      strLang = strCmd.Mid( cnt1, cnt2 - cnt1 );
  
```
m_dwLangID = (DWORD)_wtoi(strLang);

cnt1 = strCmd.Find(_T("/H")) + 2;
cnt2 = strCmd.Find(_T(" "), cnt1);
if (cnt2 == EOF)
    hWnd = strCmd.Mid(cnt1);
else
    hWnd = strCmd.Mid(cnt1, cnt2 - cnt1);
m_hSendWnd = (HWND)_wtoi(hWnd);

...
9 Development of Application in VC++
9.2 Development of Application

Initializing Process <2>
Send the window handle of the application CMainFrame to the programming pendant main software at the timing when the window (CMainFrame) is displayed. Do not fail to deliver the window handle of the CMainFrame.

[Code example]

```cpp
int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
    . . .

    // add code ///////////////////////////////////////////////////////////////////////////////////
    // Send the created command
    HWND hWnd = GetSafeHwnd();

    SendCopyData( EXE2YPP_CREATE_CMD, sizeof(HWND), &hWnd );

    /////////////////////////////////////////////////////////////////////////////////////////////

    return 0;
}
```

VOID CMainFrame::SendCopyData(
DWORD dwCommandID, /* [I] EXE->YPP command ID */
DWORD dwDataSize, /* [I] lpSendData size */
LPVOID lpSendData /* [I] Pointer to the sending data */
)
{
    HWND hSendWnd;
    COPYDATASTRUCT CopyDataStruct;
    LRESULT lrc;

    // Window handle destination
    hSendWnd = ((CSampleApp*)::AfxGetApp())->m_hSendWnd;;
    if( hSendWnd == NULL )
        return;

    // Send message
    CopyDataStruct.dwData = dwCommandID;
    CopyDataStruct.cbData = dwDataSize;
    CopyDataStruct.lpData = lpSendData;
```
9 Development of Application in VC++
9.2 Development of Application

```cpp
lrc = ::SendMessage(hSendWnd, WM_COPYDATA, (WPARAM)m_hWnd,
                     (LPARAM)&CopyDataStruct);
```

Closing Process

Notify the programming pendant main software of the closing when the customized application is closed.

**[Code example]**

```cpp
int CMainFrame::OnClose()
{
    . . .

    /// add code ///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
    // Send closing command
    SendCopyData(EXE2YPP_CLOSE_CMD, 0, NULL);
    ///////////////////////////////////////////////////////////////////////////

    . . .
}
```

**NOTE**

Code examples are mentioned above, however, always create the appropriate code for the application.
9.3 Application Debugging

Parameter setting is necessary for debugging the user-customized application.

9.3.1 Parameter Setting

For the parameter set S2C533=1 for debugging the created applications.

**NOTE**

Do not fail to set S2C533=0 before releasing the application.

9.3.2 Debugging

Following is the procedures for debugging the application which is being created.

1. Select {DEBUG} - {DEBUG OVERLAP} under the main menu.
9 Development of Application in VC++
9.3 Application Debugging

2. “for ExternalExe Debug” window appears.

3. Select “Configuration Properties” – “Debugging” under the “NxAPITestId Property Pages” window. Enter the “argument:” of “for ExternalExe Debug” window which were displayed on the programming pendant in step 2 to the “Command Arguments”.

4. Press (Connect) on “for External Debug” window. The application appears on the programming pendant window. Procedures after this are same as the ones executed on the desk top.
9.3.3 Operation Check

Like the checking operation to the released version, operations of the created application can be checked without installed.

1. Set PPAPP.exe as a name of the application.
2. Place the PPAPP.exe on the root of DiskOnChip.
3. Select {DEBUG} - {RUN OVERLAP} under the main menu.
4. The application starts.
10 Development of Application in C#

10.1 Preparation of the PC for Developing Application

   - An update is available for Visual Studio 2008 SP1 that adds support for Windows Embedded Compact 7
     https://support.microsoft.com/en-us/kb/2483802
   - Windows Embedded Compact 7 ATL Update for Visual Studio 2008 SP1
   - MSDN Library for Visual Studio 2008 SP1
5. Windows Embedded Compact 7 Update: Download the latest version from official Microsoft download center and install.
6. IP address setting for PC for developing application
   - Set the IP address for PC for developing application as follows.
     IP address: 10.0.0.3
     Subnet mask: 255.255.255.0
10 Development of Application in C#

10.2 Development of Application

Create a new project in Visual Studio 2008.

10.2.1 Creation of Software (PC for Development)

With Visual Studio, create a new project for creating application for Smart Device.

10.2.2 Controller Communication Library

1. Copy MotoConnectCSU.dll to under the C#_Library folder to the application folder to be created.

2. Add MotoConnectCSU.dll to the project by “Reference setting”.

![Visual Studio Project Settings](image-url)
10.2.3 Initialization Process of Library

Create a MotoConnectCS object in order to utilize the API of MotoConnectCS.

**[Code example]**

```csharp
public partial class Form1 : Form
{
    #region Initialize
    /// <summary>
    /// Command Line
    /// </summary>
    private string[] CommandLine;
    public string[] SetCommandLine
    {
        set
        {
            CommandLine = value;
        }
    }

    /// <summary>
    /// AppOption : Analyze Command Line
    /// </summary>
    private AppOption AppOpt;

    /// <summary>
    /// AppMessageWindow : Communicate YppMain
    /// </summary>
    private AppMessageWindow AppMsg;

    /// <summary>
    /// Endian true:big / false:little
    /// </summary>
    private bool EndianData = false;

    #endregion
    #region MotoConnectCSU
    private MotoConnectCSU.MotoConnect MotoConnect;
    #endregion
```
public Form1()
{
    InitializeComponent();
}

private void Form1_Load(object sender, EventArgs e)
{
    this.Visible = false;
    // Set the command line argument
    AppOpt = new AppOption();
    AppOpt.SetOptionData(CommandLine);
    AppMsg =
    new AppMessageWindow(this, AppOpt.GetMainHWnd);
    MotoConnect =
    new MotoConnectCS.MotoConnect(EndianData);
    // Completion of start up command sending process
    AppMsg.SendCreateCommand(Handle);

    this.Visible = true;
}

...
10.2.4 Interface with Programming Pendant Main Software

For sending and receiving data with the programming pendant main software, the user-customized application employs WM_COPYDATA for the communication between APIs. Followings are the important processing operations.

**Initializing Process <1>**

The following four arguments are delivered from the pendant main software, when the customized application is started up.

- Language code (/L)
- Window handle of the programming pendant main software (/H)
- IP address of the YRC1000 (/CIP)
- IP address of the programming pendant (/PIP)

For the user-customized application, obtain the language code and the window handle of the programming pendant main software and save them.

**[Code example]**

```csharp
static class Program
{
    /// <summary>
    /// It is the main entry point of the application.
    /// </summary>
    [MTAThread]
    static void Main(string[] args)
    {
        Form1 form1 = new Form1();
        form1.SetCommandLine = args;
        Application.Run(form1);
    }
}
```

```csharp
public partial class Form1 : Form
{
    #region Initialize
    /// <summary>
    /// Command Line
    /// </summary>
    private string[] CommandLine;
    public string[] SetCommandLine
    {
        set
        {
            set
```

10 Development of Application in C#
10.2 Development of Application

```csharp
    { 
        CommandLine = value;
    }

/// <summary>
/// AppOption : Analyze Command Line
/// </summary>
private AppOption AppOpt;

/// <summary>
/// AppMessageWindow : Communicate YppMain
/// </summary>
private AppMessageWindow AppMsg;

... 

For the AppOption class, refer to its source code (AppOption.cs) which will be provided.
```
10 Development of Application in C#
10.2 Development of Application

**Initializing Process <2>**

Send the window handle of the application to the programming pendant main software at the timing when the user-customized application window is displayed. The programming pendant main software sends messages to the window to which the window handle is sent.

**[Code example]**

```csharp
public partial class Form1 : Form
{
    ...

    private void Form1_Load(object sender, EventArgs e)
    {
        this.Visible = false;

        // Set the command line argument
        AppOpt = new AppOption();
        AppOpt.SetOptionData(CommandLine);
        AppMsg = new AppMessageWindow(this, AppOpt.GetMainHWnd);
        MotoConnect = new MotoConnectCS.MotoConnect(EndianData);

        // Completion of start upcommand sending process
        AppMsg.SendCreateCommand(Handle);

        this.Visible = true;
    }

    ...
}
```
10.2 Development of Application

Closing Process

Notify the programming pendant main software of the closing when the customized application is closed.

[Code example]

```csharp
public partial class Form1 : Form
{
    ...

    private void button2_Click(object sender, EventArgs e)
    {
        // Close
        AppMsg.SendCloseCommand();
    }
}
```

For the AppMessageWindow class, refer to its source code (AppMessageWindow.cs) which will be provided.

AppMessageWindow.cs includes the following codes.

- WM_COPYDATA message receiving from the programming pendant main software
- WM_COPYDATA message sending to the programming pendant main software
- Examples of important process such as Initializing or closing process
### 10.3 Application Debugging

Parameter setting is necessary for debugging the user-customized application.

#### 10.3.1 Parameter Setting

For the parameter set S2C533=1 for debugging the created applications.

**NOTE**

Do not fail to set S2C533=0 before releasing the application.

#### 10.3.2 Debugging

Following is the procedures for debugging the application which is being created.

1. Select {DEBUG} - {DEBUG OVERLAP}.
10. Development of Application in C#

10.3 Application Debugging

2. “for ExternalExe Debug” window appears.

3. From “Property Pages”, select “Debug”. Enter the “argument:” of “for ExternalExe Debug” window which were displayed on the programming pendant in step 2 to the “Command line arguments”.

4. Press [Connect] on “for External Debug” window. The application appears on the programming pendant window. Procedures after this are same as the ones executed on the desk top.
11 Interface Specification with Programming Pendant Main Software

11.1 Command from the User Customized Application to Programming Pendant Main Software

11.1.1 Completion of Generation

<table>
<thead>
<tr>
<th>Code name</th>
<th>Data configuration (Type)</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXE2YPP_CREATE_CMD</td>
<td>HWND hwnd : Window handle for message sending destination</td>
</tr>
</tbody>
</table>

<Description>

The user-customized application sends this command to the programming pendant software when it is started up.

The sending window handle is the main frame handle of the user-customized application.

The programming pendant main software sends messages to this window while the user-customized application is running.

11.1.2 Closing of Application

<table>
<thead>
<tr>
<th>Code name</th>
<th>Data configuration (Type)</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXE2YPP_CLOSE_CMD</td>
<td>Not exist</td>
</tr>
</tbody>
</table>

<Description>

The user-customized application sends this command to the programming pendant software when it is closed.

After receiving this command, the software side will discard or release each object.
11 Interface Specification with Programming Pendant Main Software

11.1 Command from the User Customized Application to Programming Pendant Main Software

11.1.3 Keymask

<Description>

With this command, keys sent to the YRC1000 can be limited from the user-customized application. When setting “1” to the objective key, it is sent to the YRC1000.

<Key-Bit table>

<table>
<thead>
<tr>
<th>Code name</th>
<th>Data configuration (Type)</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXE2YPP_KEYMASK_CMD</td>
<td>EBYTE byte[8] : Keymask bit</td>
</tr>
<tr>
<td></td>
<td>Not exist</td>
</tr>
</tbody>
</table>

<Example>

Set as follows when sending only [CANCEL] + [SELECT] to the YRC1000.

byte[8] = {0x00, 0x08, 0x00, 0x00, 0x00, 0x00, 0x10};

When there are no sending data, present specified key mask at the programming pendant software side is set.

<table>
<thead>
<tr>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td>AREA</td>
<td>PAGE</td>
<td>COORD</td>
<td>MULTI</td>
<td>ARROW DOWN</td>
<td>ARROW UP</td>
<td>ARROW RIGHT</td>
<td>ARROW LEFT</td>
</tr>
<tr>
<td>Y+/L+</td>
<td>Y-/L-</td>
<td>X+/S+</td>
<td>X-/S-</td>
<td>CANCEL</td>
<td>DIRECT OPEN</td>
<td>SIMPLE MENU</td>
<td>MAIN MENU</td>
</tr>
<tr>
<td>SLOW</td>
<td>FAST</td>
<td>HIGH SPEED</td>
<td>SERVO ON READY</td>
<td>7+</td>
<td>7-</td>
<td>Z+/U+</td>
<td>Z-/U-</td>
</tr>
<tr>
<td>8+</td>
<td>8-</td>
<td>Z+/U+</td>
<td>Z-/T-</td>
<td>Y+/B+</td>
<td>Y-/B-</td>
<td>X+/R+</td>
<td>X-/R-</td>
</tr>
<tr>
<td>ROBOT</td>
<td>SHIFT RIGHT</td>
<td>TEST RUN</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>INTER LOCK</td>
<td>SHIFT LEFT</td>
</tr>
<tr>
<td>USAGE</td>
<td>EX AXIS</td>
<td>FWD</td>
<td>BWD</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>INFORM LIST</td>
</tr>
<tr>
<td>0</td>
<td>MOTION TYPE</td>
<td>AUX</td>
<td>INSERT</td>
<td>DELETE</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Spare</td>
<td>Spare</td>
<td>Spare</td>
<td>SELECT</td>
<td>ENTER</td>
<td>MODIFY</td>
<td>-</td>
<td>.</td>
</tr>
</tbody>
</table>
11.1.4 Text Input Function Start

<table>
<thead>
<tr>
<th>Code name</th>
<th>Data configuration (Type)</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXE2YPP_KEYPAD_CMD</td>
<td>DWORD type : keypad type (English/Japanese, etc.)</td>
</tr>
<tr>
<td></td>
<td>DWORD maxlen : limit of inputting character</td>
</tr>
<tr>
<td></td>
<td>SCHAR buf[128] : default character strings for keypad</td>
</tr>
</tbody>
</table>

<Description>

With this command, the character inputting keypad becomes available from the user-customized application.

11.1.5 Switching Control

<table>
<thead>
<tr>
<th>Code name</th>
<th>Data configuration (Type)</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXE2YPP_CONTROL_CMD</td>
<td>DWORD target 1: programming pendant main software</td>
</tr>
<tr>
<td></td>
<td>2: user-customized application</td>
</tr>
</tbody>
</table>

<Description>

With this command, the control system can be alternated between the programming pendant main software and the user-customized application by sending data.

If it is switched to the programming pendant main software, the software side changes the display or releases the keymask, etc.

If it is switched to the user-customized application, the application side acquires the latest necessary information and displays it, and it should execute some initializing operations such as re-setting the keymask, etc.
11.1.6 Setting of Notifying Information

<table>
<thead>
<tr>
<th>Code name</th>
<th>Data configuration (Type)</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXE2YPP_GET_INFOAUTO_CMD</td>
<td>DWORD notifyingfo</td>
</tr>
<tr>
<td></td>
<td>Information to be notified (specified by bit)</td>
</tr>
</tbody>
</table>

<Description>
With this command, setting of notifying information such as changes of the programming pendant software side status or information sent to the user-customized application when it is requested become available. If it is not set, all the information is notified.

Bit is used for this setting.
0 bit: Group axis
1 bit: Motion coordinate system
2 bit: Manual speed
3 bit: Security
4 bit: Motion cycle
5 bit: Executing status
6 bit: Page shift
7 bit: Synchronizing status
8 bit: Battery
9 bit: Teach/play mode
10 bit: Multi window mode

11.1.7 Information Request

<table>
<thead>
<tr>
<th>Code name</th>
<th>Data configuration (Type)</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXE2YPP_GET_INFO_CMD</td>
<td>Not exist</td>
</tr>
</tbody>
</table>

<Description>
With this command, request of the information to the programming pendant main software becomes available.

To this request, the software side sends the information previously set as sending information to the user-customized application.
11.1.8 Version Information Acquisition

<table>
<thead>
<tr>
<th>Code name</th>
<th>Data configuration (Type)</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXE2YPP_GET_VERSION_CMD</td>
<td>Not exist</td>
</tr>
</tbody>
</table>

<Description>

With this command, request of the version to the programming pendant main software becomes available.

To this request, the software side sends the version which is under running to the user-customized application.
11.2 Command from the Programming Pendant Main Software to the User-Customized Application

11.2.1 Input Character

<table>
<thead>
<tr>
<th>Code name</th>
<th>Data configuration (Type)</th>
</tr>
</thead>
<tbody>
<tr>
<td>YPPEXE_KEYPADRET_CMD</td>
<td>WORD okcancel, WCHAR buf[128]</td>
</tr>
</tbody>
</table>

<Description>

The programming pendant main software sends this command after completion of inputting character strings keypad started by the user-customized application.

Receiving this command, the application side can acquire the input character strings.

11.2.2 Request of Control System Switching

<table>
<thead>
<tr>
<th>Code name</th>
<th>Data configuration (Type)</th>
</tr>
</thead>
<tbody>
<tr>
<td>YPPEXE_CONTROL_CMD</td>
<td>DWORD target</td>
</tr>
</tbody>
</table>

<Description>

With this command, the control system can be switched between the programming pendant main software and the user-customized application as specified by the sent data.

When switching to the programming pendant main software control system, the software side will change the indication or release the keymask after receiving this command.

When switching to the user-customized application control system, the application side should execute various initializing operations such as acquiring and indicating the necessary information, or resetting the keymask, etc. after receiving this command.
11.2.3 Status Information

<Description>

With this command, the programming pendant software side sends the status information when

- the status information which is previously-set as sending information is changed

or

- there is information which is set as a sending information when requested by the user-customized application.

When the sending information is changed, “1” is set to “enable”. And a status value is set to “value”.

Followings are the element numbers of both “enable” and “value” arrangements.

<table>
<thead>
<tr>
<th>Code name</th>
<th>Data configuration (Type)</th>
</tr>
</thead>
<tbody>
<tr>
<td>YPPEXE_STATUS_CMD</td>
<td>DWORD enable[16] Valid information</td>
</tr>
<tr>
<td></td>
<td>DWORD value[16] Status value</td>
</tr>
</tbody>
</table>

0 : Group axis
1 : Motion coordinate system
2 : Manual speed
3 : Security
4 : Motion cycle
5 : Executing status
6 : Page shift
7 : Synchronizing status
8 : Battery/Accessing to SD/Touch operation invalid
9 : Teach/play mode
10 : Multi window mode
11-15 : (Unused)
11.2 Command from the Programming Pendant Main Software to the User-Customized Application

Followings are the status value of each element.

<table>
<thead>
<tr>
<th>Setting value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>R1</td>
</tr>
<tr>
<td>1</td>
<td>R2</td>
</tr>
<tr>
<td>2</td>
<td>R3</td>
</tr>
<tr>
<td>3</td>
<td>R4</td>
</tr>
<tr>
<td>4</td>
<td>R5</td>
</tr>
<tr>
<td>5</td>
<td>R6</td>
</tr>
<tr>
<td>6</td>
<td>R7</td>
</tr>
<tr>
<td>7</td>
<td>R8</td>
</tr>
<tr>
<td>8</td>
<td>B1</td>
</tr>
<tr>
<td>9</td>
<td>B2</td>
</tr>
<tr>
<td>10</td>
<td>B3</td>
</tr>
<tr>
<td>11</td>
<td>B4</td>
</tr>
<tr>
<td>12</td>
<td>B5</td>
</tr>
<tr>
<td>13</td>
<td>B6</td>
</tr>
<tr>
<td>14</td>
<td>B7</td>
</tr>
<tr>
<td>15</td>
<td>B8</td>
</tr>
<tr>
<td>16</td>
<td>S1</td>
</tr>
<tr>
<td>17</td>
<td>S2</td>
</tr>
<tr>
<td>18</td>
<td>S3</td>
</tr>
<tr>
<td>19</td>
<td>S4</td>
</tr>
<tr>
<td>20</td>
<td>S5</td>
</tr>
<tr>
<td>21</td>
<td>S6</td>
</tr>
<tr>
<td>22</td>
<td>S7</td>
</tr>
<tr>
<td>23</td>
<td>S8</td>
</tr>
<tr>
<td>24</td>
<td>S9</td>
</tr>
<tr>
<td>25</td>
<td>S10</td>
</tr>
<tr>
<td>26</td>
<td>S11</td>
</tr>
<tr>
<td>27</td>
<td>S12</td>
</tr>
<tr>
<td>28</td>
<td>S13</td>
</tr>
<tr>
<td>29</td>
<td>S14</td>
</tr>
<tr>
<td>30</td>
<td>S15</td>
</tr>
<tr>
<td>31</td>
<td>S16</td>
</tr>
<tr>
<td>32</td>
<td>S17</td>
</tr>
<tr>
<td>33</td>
<td>S18</td>
</tr>
<tr>
<td>34</td>
<td>S19</td>
</tr>
<tr>
<td>35</td>
<td>S20</td>
</tr>
<tr>
<td>36</td>
<td>S21</td>
</tr>
<tr>
<td>37</td>
<td>S22</td>
</tr>
<tr>
<td>38</td>
<td>S23</td>
</tr>
<tr>
<td>39</td>
<td>S24</td>
</tr>
</tbody>
</table>
11 Interface Specification with Programming Pendant Main Software
11.2 Command from the Programming Pendant Main Software to the User-Customized Application

1: Motion coordinate system

<table>
<thead>
<tr>
<th>Setting value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Link coordinate</td>
</tr>
<tr>
<td>1</td>
<td>Cartesian coordinate</td>
</tr>
<tr>
<td>2</td>
<td>Cylindrical coordinate</td>
</tr>
<tr>
<td>3</td>
<td>Tool coordinate</td>
</tr>
<tr>
<td>4</td>
<td>User coordinate</td>
</tr>
<tr>
<td>5</td>
<td>External reference point coordinate</td>
</tr>
<tr>
<td>6</td>
<td>Teaching line coordinate</td>
</tr>
<tr>
<td>7</td>
<td>(Unused)</td>
</tr>
<tr>
<td>8</td>
<td>(Unused)</td>
</tr>
<tr>
<td>9</td>
<td>(Unused)</td>
</tr>
<tr>
<td>10</td>
<td>(Unused)</td>
</tr>
<tr>
<td>11-74</td>
<td>User coordinate (1-64)</td>
</tr>
</tbody>
</table>

2: Manual speed

<table>
<thead>
<tr>
<th>Setting value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Inching</td>
</tr>
<tr>
<td>1</td>
<td>Low speed</td>
</tr>
<tr>
<td>2</td>
<td>Middle speed</td>
</tr>
<tr>
<td>3</td>
<td>High speed</td>
</tr>
</tbody>
</table>

3: Security

<table>
<thead>
<tr>
<th>Setting value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Operation mode</td>
</tr>
<tr>
<td>1</td>
<td>Edit mode</td>
</tr>
<tr>
<td>2</td>
<td>Management mode</td>
</tr>
<tr>
<td>3</td>
<td>Safety mode</td>
</tr>
</tbody>
</table>

4: Motion cycle

<table>
<thead>
<tr>
<th>Setting value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Continuous</td>
</tr>
<tr>
<td>1</td>
<td>1 cycle</td>
</tr>
<tr>
<td>2</td>
<td>Step</td>
</tr>
</tbody>
</table>
### 5: Executing status

<table>
<thead>
<tr>
<th>Setting value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Alarming</td>
</tr>
<tr>
<td>1</td>
<td>Stopped by emergency stop</td>
</tr>
<tr>
<td>2</td>
<td>Hold</td>
</tr>
<tr>
<td>3</td>
<td>Playback</td>
</tr>
<tr>
<td>4</td>
<td>Playback</td>
</tr>
<tr>
<td>5</td>
<td>Stopped</td>
</tr>
</tbody>
</table>

### 6: Page switch

<table>
<thead>
<tr>
<th>Setting value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Page does not switch</td>
</tr>
<tr>
<td>1</td>
<td>Page switches</td>
</tr>
</tbody>
</table>

### 7: Synchronized motion

<table>
<thead>
<tr>
<th>Setting value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not synchronized</td>
</tr>
<tr>
<td>1</td>
<td>Synchronizing</td>
</tr>
</tbody>
</table>

### 8: Battery

<table>
<thead>
<tr>
<th>Setting value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal</td>
</tr>
<tr>
<td>1</td>
<td>Exhausted</td>
</tr>
<tr>
<td>2</td>
<td>Accessing to SD</td>
</tr>
<tr>
<td>3</td>
<td>Touch operation invalid</td>
</tr>
<tr>
<td>4</td>
<td>Exhausted and touch operation invalid</td>
</tr>
</tbody>
</table>

### 9: Teach/play mode

<table>
<thead>
<tr>
<th>Setting value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Initial status</td>
</tr>
<tr>
<td>1</td>
<td>Play mode</td>
</tr>
<tr>
<td>2</td>
<td>Teach mode</td>
</tr>
<tr>
<td>3</td>
<td>Play mode</td>
</tr>
<tr>
<td>4</td>
<td>Teach mode</td>
</tr>
</tbody>
</table>
11 Interface Specification with Programming Pendant Main Software
11.2 Command from the Programming Pendant Main Software to the User-Customized Application

10: Multi window indication mode

<table>
<thead>
<tr>
<th>Setting value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Single mode</td>
</tr>
<tr>
<td>1</td>
<td>Multi mode</td>
</tr>
<tr>
<td>2</td>
<td>Multi mode</td>
</tr>
</tbody>
</table>

Example
When changing the mode from teach to play
enable = {0.0.0.0.0.0.0.0.0.1.0.0.0.0.0.0}
value = {0.0.0.0.0.0.0.0.0.1.0.0.0.0.0.0}

When changing to return the mode from play to teach
enable = {0.0.0.0.0.0.0.0.0.1.0.0.0.0.0.0}
value = {0.0.0.0.0.0.0.0.0.2.0.0.0.0.0.0}

More than two “enable” s may become valid when several status change occurred.
In case no information exists, “0xffff” is set to “value”.

11.2.4 Alarm

<table>
<thead>
<tr>
<th>Code name</th>
<th>Data configuration (Type)</th>
</tr>
</thead>
<tbody>
<tr>
<td>YPPEXE_ALARM_CMD</td>
<td>DWORD status</td>
</tr>
<tr>
<td></td>
<td>1 = alarming /</td>
</tr>
<tr>
<td></td>
<td>2 = alarm released</td>
</tr>
</tbody>
</table>

<Description>
The programming pendant main software sends this command when alarm is occurred or released.
11.2.5 Message

<table>
<thead>
<tr>
<th>Code name</th>
<th>Data configuration (Type)</th>
</tr>
</thead>
<tbody>
<tr>
<td>YPPEXE_MESSAGE_CMD</td>
<td>DWORD msgnum  Numbers of message</td>
</tr>
<tr>
<td></td>
<td>struct{</td>
</tr>
<tr>
<td></td>
<td>WORD code Message code</td>
</tr>
<tr>
<td></td>
<td>WCHAR buf[128] Message (volume of data for all messages)</td>
</tr>
<tr>
<td></td>
<td>*msgnum</td>
</tr>
</tbody>
</table>

<Description>

When the message is modified, the programming pendant software sends this command.

More than two messages are sent at a time when several messages are modified.

Meanings of the message code

1: Error message
2: Warning message
3: System status message

11.2.6 Language

<table>
<thead>
<tr>
<th>Code name</th>
<th>Data configuration (Type)</th>
</tr>
</thead>
<tbody>
<tr>
<td>YPPEXE_LANGUAGE_CMD</td>
<td>DWORD language language code</td>
</tr>
</tbody>
</table>

<Description>

When receiving a information request command (EXE2YPP_GET_INFO_cmd) or the language code is modified, the programming pendant software sends this command.

11.2.7 Version

<table>
<thead>
<tr>
<th>Code name</th>
<th>Data configuration (Type)</th>
</tr>
</thead>
<tbody>
<tr>
<td>YPPEXE_VERSION_CMD</td>
<td>WCHAR buff[16] version information</td>
</tr>
</tbody>
</table>

<Description>

When the programming pendant software received a request for acquiring version information, it sends this command.