DX100/DX200/FS100 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
DX100/DX200 INSTRUCTIONS
DX100/DX200 OPERATOR'S MANUAL
DX100/DX200 MAINTENANCE MANUAL

FS100 INSTRUCTIONS
FS100 OPERATOR'S MANUAL
FS100 MAINTENANCE MANUAL

The DX100/DX200/FS100 operator's manual above corresponds to specific usage.
Be sure to use the appropriate manual.

Part Number: 166581-1CD
Revision: 0

MANUAL NO.
HW1481481①

MANDATORY

- This manual explains the programming pendant customization function with the DX100/DX200/FS100 system. Read this manual carefully and be sure to understand its contents before handling the DX100/DX200.
- General items related to safety are listed in Chapter 1: Safety of the DX100/DX200 Instructions. To ensure correct and safe operation, carefully read the DX100/DX200 Instructions before reading this manual.

CAUTION

- Some drawings in this manual are shown with the protective covers or shields removed for clarity. Be sure all covers and shields are replaced before operating this product.
- 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.
- YASKAWA is not responsible for incidents arising from unauthorized modification of its products. Unauthorized modification voids your product's warranty.

Notes for Safe Operation

Read this manual carefully before installation, operation, maintenance, or inspection of the MOTOMAN-MH6-10/MH6F-10.

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

WARNING

Indicates a potentially hazardous situation which, if not avoided, could result in death or serious injury to personnel.

CAUTION

Indicates a potentially hazardous situation which, if not avoided, could result in minor or moderate injury to personnel and damage to equipment. It may also be used to alert against unsafe practices.

MANDATORY

Always be sure to follow explicitly the items listed under this heading.

PROHIBITED

Must never be performed.

Even items described as “CAUTION” may result in a serious accident in some situations.

At any rate, be sure to follow these important items

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

DX100/DX200>

WARNING

- Before operating the manipulator, check that servo power is turned OFF pressing the emergency stop buttons on the front door of the DX100/DX200 and the programming.
  When the servo power is turned OFF, the SERVO ON LED on the programming is turned OFF.

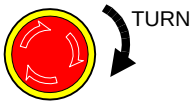
Injury or damage to machinery may result if the emergency stop circuit cannot stop the manipulator during an emergency. The manipulator should not be used if the emergency stop buttons do not function.

Figure 1: Emergency Stop Button

- Once the emergency stop button is released, clear the cell of all items which could interfere with the operation of the manipulator. Then turn the servo power ON.

Injury may result from unintentional or unexpected manipulator motion.

Figure 2: Release of Emergency Stop



- Observe the following precautions when performing teaching operations within the P-point maximum envelope of the manipulator:
  - View the manipulator from the front whenever possible.
  - Always follow the predetermined operating procedure.
  - Keep in mind the emergency response measures against the manipulator's unexpected motion toward you.
  - Ensure that you have a safe place to retreat in case of emergency.

Improper or unintended manipulator operation may result in injury.

- Confirm that no person is present in the P-point maximum envelope of the manipulator and that you are in a safe location before:
  - Turning ON the power for the DX100/DX200.
  - Moving the manipulator with the programming.
  - Running the system in the check mode.
  - Performing automatic operations.

Injury may result if anyone enters the P-point maximum envelope of the manipulator during operation. Always press an emergency stop button immediately if there is a problem.

The emergency stop buttons are located on the right of front door of the DX100/DX200 and the programming.

<FS100>

WARNING

- Before operating the manipulator, check that servo power is turned OFF when the emergency stop button on the programming is pressed.
  When the servo power is turned OFF, the SERVO ON LED on the programming is turned OFF.

Injury or damage to machinery may result if the emergency stop circuit cannot stop the manipulator during an emergency.

Figure 3: Emergency Stop Button

- In the case of not using the programming, be sure to supply the emergency stop button on the equipment. Then before operating the manipulator, check to be sure that the servo power is turned OFF by pressing the emergency stop button.
  Connect the external emergency stop button to the 5-6 pin and 16-17 pin of the robot system signal connector (CN2).
- Upon shipment of the FS100, this signal is connected by a jumper cable in the dummy connector. To use the signal, make sure to supply a new connector, and then input it.

If the signal is input with the jumper cable connected, it does not function, which may result in personal injury or equipment damage.

- Once the emergency stop button is released, clear the cell of all items which could interfere with the operation of the manipulator. Then turn the servo power ON.

Injury may result from unintentional or unexpected manipulator motion.

Figure 4: Release of Emergency Stop

- Observe the following precautions when performing teaching operations within the manipulator's operating range:
  – Always follow the predetermined operating procedure.
  – Keep in mind the emergency response measures against the manipulator's unexpected motion toward you.
  – Ensure that you have a safe place to retreat in case of emergency.

Improper or unintended manipulator operation may result in injury.

- Confirm that no person is present in the manipulator's operating range and that you are in a safe location before:
  – Turning ON the FS100 power.
  – Moving the manipulator with the programming.
  – Running the system in the check mode.
  – Performing automatic operations.

Injury may result if anyone enters the manipulator's operating range during operation. Always press the emergency stop button immediately if there is a problem.
The emergency stop button is located on the right of the programming.

CAUTION

- Perform the following inspection procedures prior to conducting manipulator teaching. If problems are found, repair them immediately,
  and be sure that all other necessary processing has been performed.
  - Check for problems in manipulator movement.
  - Check for damage to insulation and sheathing of external wires.
- For the DX100/DX200, always return the programming to the hook on the cabinet of the DX100/DX200 after use.
  For the FS100, always return the programming to a safe place after use.

The programming can be damaged if it is left in the manipulator's work area, on the floor, or near fixtures.

- Read and understand the Explanation of Warning Labels in the DX100/DX200/FS100 Instructions before operating the manipulator:

Definition of Terms Used Often in This Manual (DX100/ DX200)

The MOTOMAN is the YASKAWA industrial robot product.

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

In this manual, the equipment is designated as follows:

Equipment	Manual Designation
DX100/DX200 controller	DX100/DX200
DX100/DX200 programming	Programming
Cable between the manipulator and the controller	Manipulator cable

Definition of Terms Used Often in This Manual (FS100)

The MOTOMAN is the YASKAWA industrial robot product.

The MOTOMAN usually consists of the manipulator, the FS100 controller, manipulator cables, the FS100 programming (optional), and the FS100 programming dummy connector (optional).

In this manual, the equipment is designated as follows:

Equipment	Manual Designation
FS100 controller	FS100
FS100 programming	Programming
Cable between the manipulator and the controller	Manipulator Cable
FS100 programming dummy connector	Programming dummy connector

Descriptions of the programming keys, buttons, and displays are shown as follows:

<DX100>

Equipment		Manual Designation
Programming	Character Keys	The keys which have characters printed on them are denoted with [ ]. ex. [ENTER]
	Symbol Keys	The keys which have a symbol printed on them are not denoted with [ ] but depicted with a small picture. ex. page key  The cursor key is an exception, and a picture is not shown.
	Axis Keys Numeric Keys	“Axis Keys” and “Numeric Keys” are generic names for the keys for axis operation and number input.
	Keys pressed simultaneously	When two keys are to be pressed simultaneously, the keys are shown with a “+” sign between them, ex. [SHIFT]+[COORD]
	Displays	The menu displayed in the programming is denoted with { }. ex. {JOB}

<DX200>

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

<FS100>

Equipment		Manual Designation
Programming	Character Keys	The keys which have characters printed on them are denoted with [ ]. ex. [ENTER]
	Symbol Keys	The keys which have a symbol printed on them are not denoted with [ ] but depicted with a small picture. ex. PAGE key  The Cursor is an exception, and a picture is not shown.
	Axis Keys Numeric Keys	“Axis Keys” and “Numeric Keys” are generic names for the keys for axis operation and number input.
	Keys pressed simultaneously	When two keys are to be pressed simultaneously, the keys are shown with a “+” sign between them, ex. SHIFT key  +COORD key 
	Mode Key	Three kinds of modes that can be selected by the mode key are denoted as follows: REMOTE, PLAY, or TEACH
	Button	Three buttons on the upper side of the programming are denoted as follows: HOLD button START button EMERGENCY STOP button
	Displays	The menu displayed in the programming is denoted with { }. ex. {JOB}
PC Keyboard		The name of the key is denoted ex. Ctrl key on the keyboard

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 the SELECT key is pressed, or that the item is directly selected by touching the window.

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 TM are omitted.

1 Programing Pendant Customizing Function

With this function, user-customized windows can be created by, into the DX100/DX200/FS100 programming, 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 (Windows CE application software) developed by the user (system integrator).

With these APIs, from the user-developing applications, acquiring of the DX100/DX200/FS100 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 DX100/DX200/FS100 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 DX100/DX200/FS100.
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 DX100/DX200/FS100 and register it to the menu of the system.

3 Function Setting

This function is an optional function of the DX100/DX200/FS100.
To invalid this function, like other functions, controller-based setting is necessary.

4 Parameter Setting

No.	Function	Setting Value	Default
S2C531	Automatic start of user-customized application function in the programming when start-up.	0: Invalid 1: Valid	0 (Invalid)
S2C541	Writing variables and I/O during play mode.	0:Permitted 1: Not permitted	1 (Not permitted)
S2C542	Writing variables and I/O during edit-lock status.	0: Invalid 1: Valid	1 (Not permitted)

Writing of variables and I/Os become valid when “0” is set to S2C541, however, it may influence the manipulator’s cycle time.

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

5 API for Programming Pendant Customization Function (MotoConnect API)

For reading/writing data to/from the user-customized application to/from the DX100/DX200/FS100, APIs are prepared.

There are three main types of APIs as follows. For the details of each API, refer to “DX100/DX200/FS100 OPTIONS INSTRUCTIONS REFERENCE MANUAL FOR TEACHING CUSTOMIZING FUNCTION” (Part No.: 166582-1CD).

System	Description
System monitor	Acquires the DX100/DX200/FS100 data. APIs for data reading.
System control	Operates the DX100/DX200/FS100. APIs for data writing are included.
File transmission	Transfer files with the function equivalent to External Memory Function.

6 Interface with Programming 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 “DX100/DX200/FS100 OPTIONS INSTRUCTIONS REFERENCE MANUAL FOR TEACHING CUSTOMIZING FUNCTION” (Part No.: 166582-1CD).

7 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.

7.1 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.

7.2 Auto Run at Power ON

The user-customized application can be started same time when the power supply of the DX100/DX200/FS100 is turned ON. Set this auto run function when installing the application.

Only one user-customized application can be automatically started in the DX100/DX200/FS100 system and this application can be started also from the main menu.

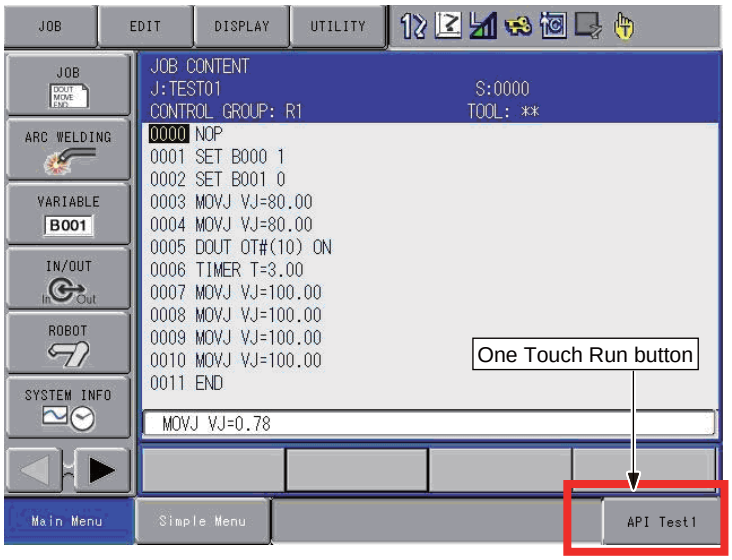
Before using this function, be sure to set “1(valid)” to the programming pendant customization application (S2C531=1) Auto Run function of at power ON.

7.3 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.



8 Installing Applications

8.1 Create the Installing Data

8.1.1 Zip File

Zip applications (*.exe), accessary files (dll, data files, etc.), and accessary 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.

8.1.2 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 = NS0.00-00
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
```

[APPLICATION_INFO]	
APP_NAME	{Application Name} 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.
APP_VER	{Application Version} Version of the application itself, and it is described as “*.**-**-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 “**.**-**-**”.
SYS_VER	{System Version} To keep compatibility with the controllers, the version is not checked. Specify “SYS_VER=NS0.00-00”.
FILE_SIZE	{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
FOLDER_NAME	{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.
APP_TYPE	{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.
SYS_STATUS	{Displaycondition 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.
AUTO_RUN	{f Auto Run Function Enabled/Disabled} Auto Run function valid = “TRUE” Auto Run function valid = “FAULSE” 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.

<DX100/FS100>

<table>
<tr><th colspan="3">[MENU_INFO</th></tr>
<tr><td>TOPMENU_ID</td><td colspan="2">{Top menu}
Specify a top menu to add the application.</td></tr>
<tr><td></td><td>Top menu</td><td>Setting value</td></tr>
<tr><td></td><td>JOB</td><td>1</td></tr>
<tr><td></td><td>WORK CONDITION</td><td>2</td></tr>
<tr><td></td><td>VARIABLE</td><td>3</td></tr>
<tr><td></td><td>I/O</td><td>4</td></tr>
<tr><td></td><td>ROBOT</td><td>5</td></tr>
<tr><td></td><td>SYSTEM INFO.</td><td>6</td></tr>
<tr><td></td><td>FD/PC CARD</td><td>7</td></tr>
<tr><td></td><td>PARAMETER</td><td>8</td></tr>
<tr><td></td><td>CONTROLLER SETTING</td><td>9</td></tr>
<tr><td></td><td>OPTIONAL</td><td>10</td></tr>
<tr><td></td><td>MANUFACTURE RESERVE</td><td>11</td></tr>
<tr><td></td><td>(FUTURE EXPANSION)</td><td>12-20</td></tr>
<tr><td></td><td>APPLICATION</td><td>21</td></tr>
<tr><td>SUBMENU_NAME_*</td><td colspan="2">{Submenu Name}
Specify a character string for indication in the sub menu
Desired language can be chosen from the followings.
Note: Up to 20 single-byte characters.
Up to 14 single-byte characters for the name of OneTouch Run function button.
SUBMENU_NAME_JAPANESE= “12345678901234567890”
SUBMENU_NAME_ENGLISH= “BBB”
SUBMENU_NAME_GERMAN= “CCC”
SUBMENU_NAME_SWEDISH= “DDD”
SUBMENU_NAME_FRENCH= “EEE”
SUBMENU_NAME_FINNISH= “FFF”
SUBMENU_NAME_ITALIAN= “GGG”
SUBMENU_NAME_SPANISH= “HHH”
SUBMENU_NAME_HANGLE= “III”
SUBMENU_NAME_CHINESE= “JJJ”
SUBMENU_NAME_TAIWANESE= “KKK”
SUBMENU_NAME_CZECH= “LLL”
SUBMENU_NAME_POLISH= “MMM”</td></tr>
<tr><td>SUBMENU_ICON</td><td colspan="2">{Submenu Icon} (for future expansion)
From the icon file, specify an icon to be indicated in the sub menu.</td></tr>
</table>

<DX200>

[MENU_INFO		
TOPMENU_ID	{Top menu} Specify a top menu to add the application.	
	Top menu	Setting value
	JOB	1
	WORK CONDITION	2
	VARIABLE	3
	IN/OUT	4
	ROBOT	5
	SYSTEM INFO.	6
	EX.MEMORY	7
	PARAMETER	8
	SETUP	9
	SAFETY FUNC.	10
	PM	11
	OPTION	12
	MANUFACTURE RESERVE	13-20
	APPLICATION	21
SUBMENU_NAME_*	{Submenu Name} Specify a character string for indication in the sub menu Desired language can be chosen from the followings. Note: Up to 20 single-byte characters. Up to 14 single-byte characters for the name of OneTouch Run function button. SUBMENU_NAME_JAPANESE= “12345678901234567890” SUBMENU_NAME_ENGLISH= “BBB” SUBMENU_NAME_GERMAN= “CCC” SUBMENU_NAME_SWEDISH= “DDD” SUBMENU_NAME_FRENCH= “EEE” SUBMENU_NAME_FINNISH= “FFF” SUBMENU_NAME_ITALIAN= “GGG” SUBMENU_NAME_SPANISH= “HHH” SUBMENU_NAME_HANGLE= “III” SUBMENU_NAME_CHINESE= “JJJ” SUBMENU_NAME_TAIWANESE= “KKK” SUBMENU_NAME_CZECH= “LLL” SUBMENU_NAME_POLISH= “MMM”	
SUBMENU_ICON	{Submenu Icon} (for future expansion) From the icon file, specify an icon to be indicated in the sub menu.	

8.2 Creating Installer

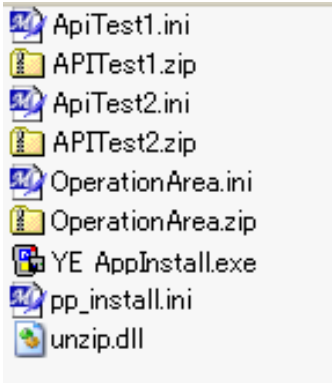
8.2.1 Necessary File

Create PP_APP folder on the CF for version up root folder.

Save the following folders in the PP_APP folder.

File name	Function	Remarks
YE_AppInstall.exe	Application installer	Manufacturer-supplied
pp_install.ini	Configuration file of the installer. Specifies the default selection of the added application when startup.	Manufacturer-supplied
Unzip.dll	Zip file decompresses	Manufacturer-supplied
*.zip	Application compressed file	
*.ini	Application configuration information file	

Example of PP_APP folder



8.2.2 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_NAMEn: Specify the name of applications to be set

[Default Setting]

APP_NUM=3

AMM_NAME1=APITest1

AMM_NAME2=APITest2

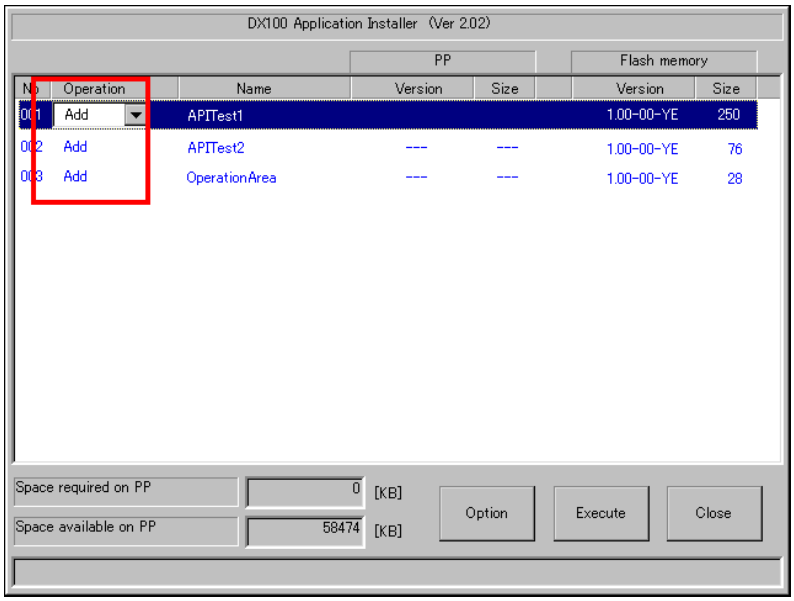
AMM_NAME3=OperationArea

8.3 Installation

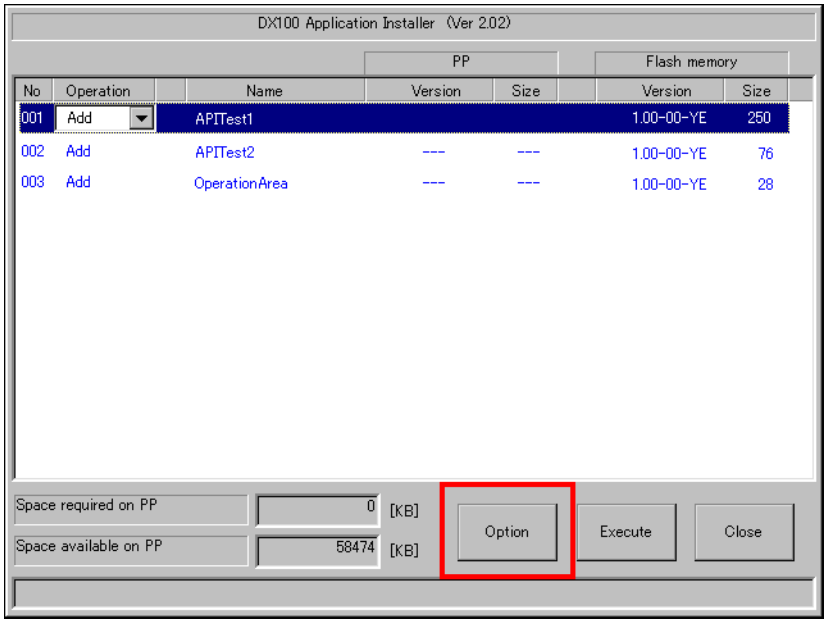
Press [INTERLOCK] + [7]+ [SELECT] keys and turn ON the power supply of the DX100/DX200/FS100.

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

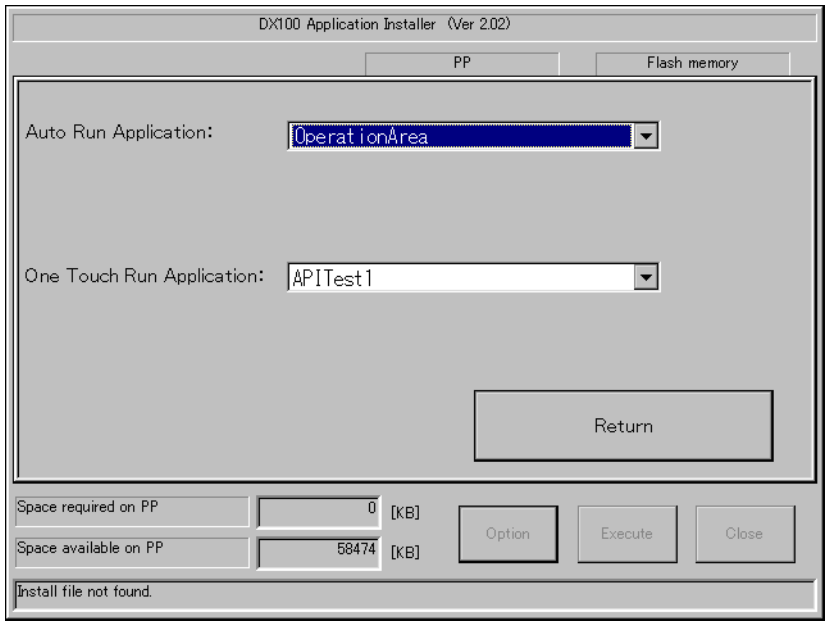
1. Check that “Add” is set to “Operation” of all the installing applications.



2. Press {Option} button 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.



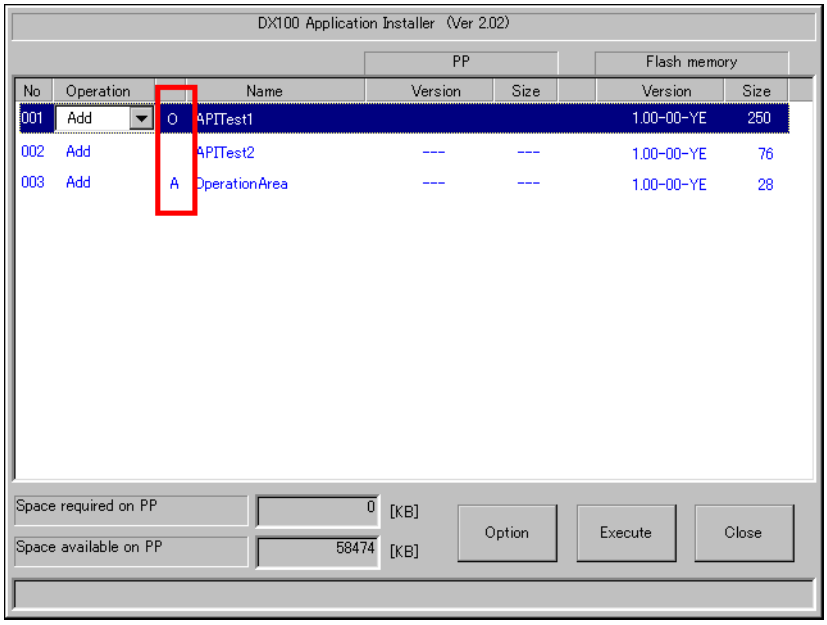
[Auto Run Application]

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.

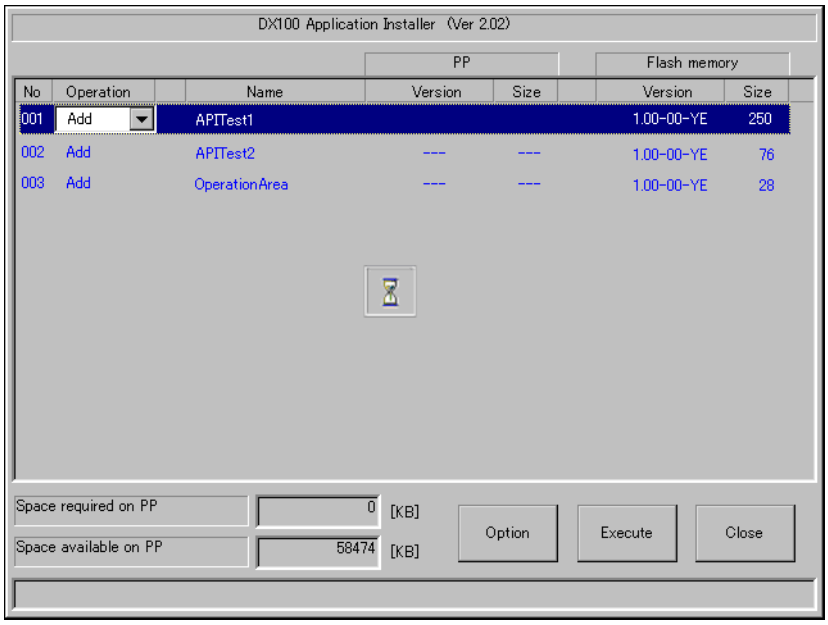
[One Touch Run Application]

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.

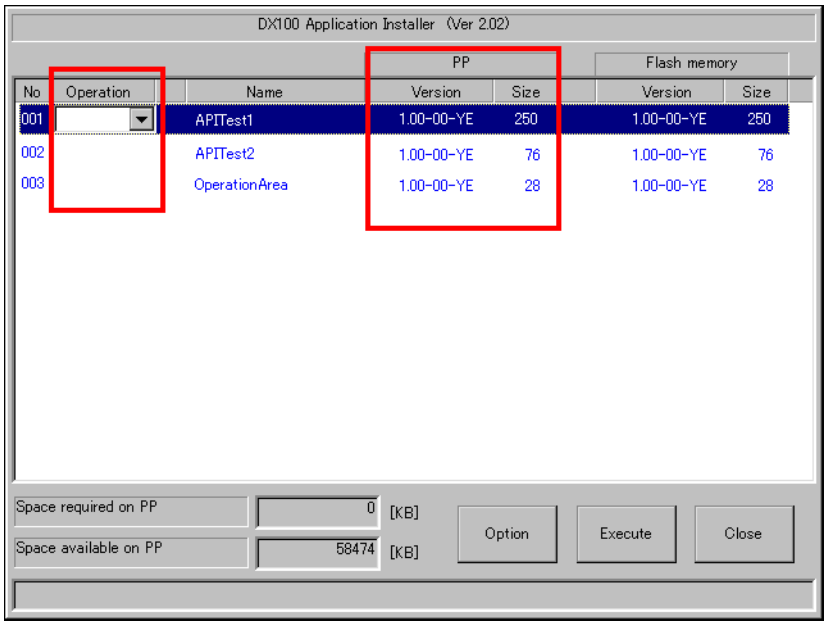
4. Press {Return} button after the above setting. The window returns to the previous one. The indication varies depending on the setting as follows.
   – “A” is indicated to the application specified as the Auto Run application at power ON.
   – “O” is indicated to the application specified as the One Touch Run with a button.
   – “*” is indicated to the application to which both functions are specified.



5. After all settings are completed, press {Execute} button. 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 DX100/DX200/FS100 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.)



SUPPLE-MENT

The installed application data is controlled by both DX100/DX200/FS100 and programming pendant. Therefore, for the data consistency, within the installed data, the inconsistent data between the DX100/DX200/FS100 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 DX100/DX200/FS100 and the programming pendant.

9 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.

9.1 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 DX100/DX200/FS100 may occur.

9.2 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 500 sec (within 300sec is recommended).

In case a process is execute which the created application continuously occupies the DX100/DX200/FS100, the YASKAWA standard programming pendant window does not work properly and communication faults (PP communication fault) with the DX100/DX200/FS100 may occur.

9.3 Successive Access to the DX100/DX200/FS100

It is prohibited to execute a process that the user-customized application continuously make access to the DX100/DX200/FS100. (Consecutive writing of the variables, for example.) The application is recommended to make a call to the API at intervals of more than 50 sec (100 msec is strongly recommended).

In case a process is execute which the created application continuously access to DX100/DX200/FS100, processing operation in the DX100/DX200/FS100 may be influenced as follows.

- Operations during the teach mode may slow down.
- Cycle times when play back operation may slow down.

9.4 Files Created by Application

Save the file created by the application to the CF (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.

9.5 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.

9.6 Application Test

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

9.6.1 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] key to move.
7. Change the mode to the play mode.
8. Indicate the cycle time.
9. Indicate the job stack.
10. Press {Start} button 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.

9.6.2 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 YIF board.
- Alarm during play back operation:
  Intentionally exceed the limit by a move instruction in which the P variable (cartesian value) is used.

9.6.3 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.

9.6.4 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.

9.6.5 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.

9.7 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.

10 Development of Application in C#

10.1 Preparation

10.1.1 Developing Environment (Developing PC Environment)

1. WindowsCE5.0 : Install
2. MS Visual Studio : Install
   - DX100/FS100:
     MS Visual Studio 2005

     Note: The debugging procedure is partially added in case Visual studio2005 ServicePack is installed.
   - DX200
     MS Visual Studio 2008
3. MS .NET Compact Framework : Install
   - DX100/FS100:
     MS .NET Compact Framework 2.0 SP2
   - DX200
     MS .NET Compact Framework 3.5
4. SDK : Install SDK for programming pendant
5. IP address setting for development PC

   Set the IP address of the developing PC as follows.

   IP Address : 10.0.0.3

   Sub-net Mask : 255.255.255.0

10.2 Development of Application for the DX100/DX200

There are following differences between the DX100/DX200 and the FS100 when developing applications. Refer to *Section 10.3 “Development of Application for the FS100” on page 10-10* developing applications for the FS100.

- Initializing process
  In the FS100 system, modification of the FS100 IP address and the programming pendant is possible. Therefore, it is required to receive them from the programming pendant main application as the parameter of the application, and deliver them as the argument of YPPConnectOpen() when the application starts up.

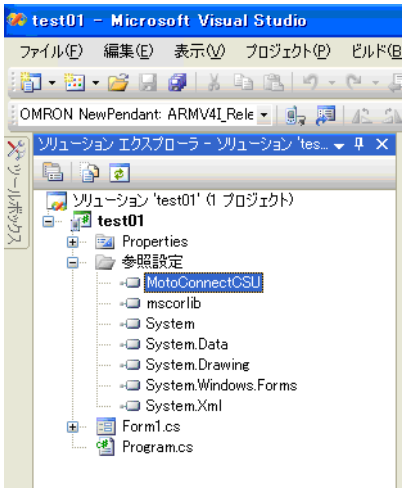
The library for C# (MotoConnectCSU.DLL) is common for both DX100, DX200 and FS100.

10.2.1 Creation of Software (PC for Development)

1. With Visual Studio, newly create a project for creating application for Windows CE 5.0.

10.2.2 Controller Communication Library

1. Copy the MotoConnect in the folder under the Develop folder to the application folder to be created.
2. Add MotoConnectCSU.dll to the project by “Reference setting”.



10.2.3 Library Initialization Process

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

[Code example]

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


    /// <summary>
    /// AppOption : Analize 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;


    /// <summary>
    /// MotoConnectCSU
    /// </summary>
    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 =
                newMotoConnectCS.MotoConnect(EndianData);
        // Completion of start upcommand 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.

Following are the important processing operations.

Initializing Process <1>

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

- Language code
- Window handle of the programming pendant main software

For the user-customized application, obtain and save the argument data.

[Code example]

```
  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);
    }
}

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

    /// <summary>
    /// AppOption : Analize 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.

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/receives messages to/from the window to which the window handle is sent.

[Code example]

```
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 up command sending process
        AppMsg.SendCreateCommand(Handle);

        this.Visible = true;
    }

        . . .
```

```
}
```

Closing Process

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

[Code example]

```
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.

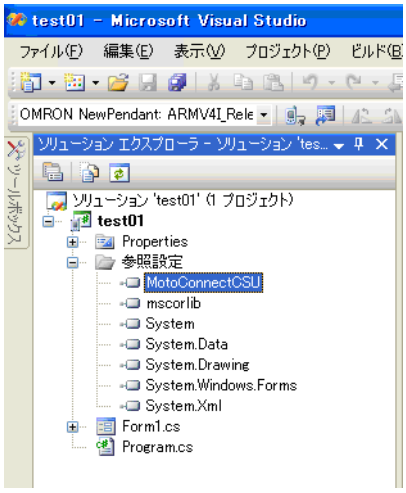
10.3 Development of Application for the FS100

10.3.1 Creation of Software (PC for Development)

With Visual studio2005, newly create a project for creating application for Windows CE 5.0.

10.3.2 Controller Communication Library

1. Copy the MotoConnect in the folder under the Develop folder to the application folder to be created.
2. Add MotoConnectCSU.dll to the project by “Reference setting”.



10.3.3 Library Initialization Process

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

[Code example]

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

    /// <summary>
    /// AppOption : Analize 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;

    /// <summary>
    /// MotoConnectCSU
    /// </summary>
    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,
                                AppOpt.GetYcpIPAddr, AppOpt.GetYppIPAddr);

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

    this.Visible = true;
}

        . . .

}
```

10.3.4 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.

Following are the important processing operations.

Initializing Process <1>

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

- Language code
- Window handle of the programming pendant main software

For the user-customized application, obtain and save the argument data.

[Code example]

```
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);
    }
}

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

```
    }

        /// <summary>
        /// AppOption : Analize 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.

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/receives messages to/from the window to which the window handle is sent.

[Code example]

```
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,
                        AppOpt.GetYcpIPAddr, AppOpt.GetYppIPAddr);

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

        this.Visible = true;
    }

        . . .

}
```

Closing Process

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

[Code example]

```
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.

10.4 Application Debugging

10.4.1 Parameter Setting

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

NOTE When releasing the debugging, do not fail to set S2C533=0 to return the parameter.

10.4.2 Debugging

Additional procedures are required as follows only when the versions except for Visual studio specified in *Section 10.1.1 "Developing Environment (Developing PC Environment)"* (including the versions of ServicePack) is installed. And this operation is necessary every time the power is turned ON.

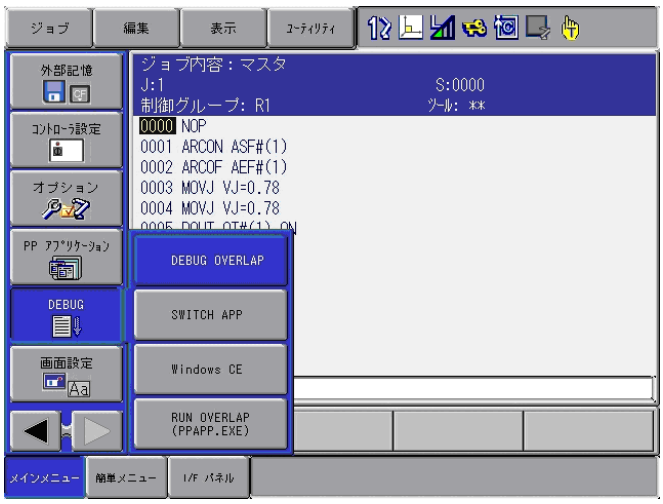
<Additional procedures>

NOTE Copy the following files to the \Window\ folder in the programming pendant. Based on the rules, those files are stored in \Program Files\Common Files\Microsoft Shared\CoreCon\1.0\Target\wce400\<CPU>.

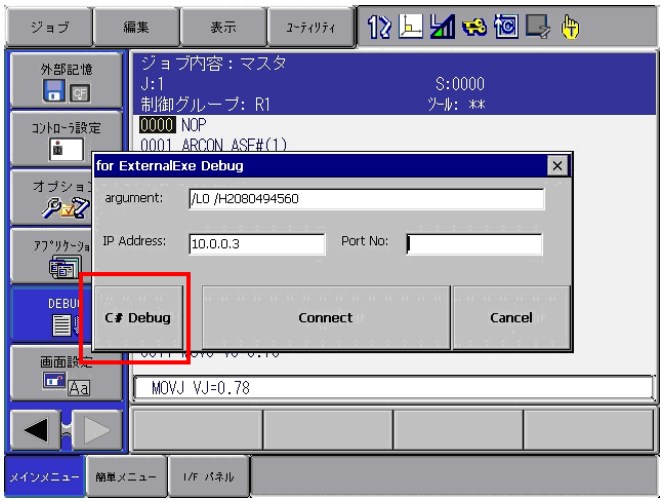
- Clientshutdown.exe
- ConmanClient2.exe
- CMaccept.exe
- eDbgTL.dll

Reference:
http://msdn.microsoft.com/ja-jp/library/ms228708(VS.80).aspx

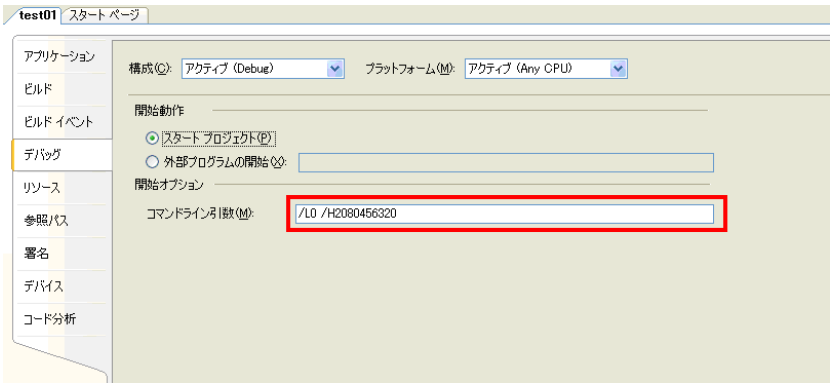
1. Select {DEBUG}- {DEBUG OVERLAP}.



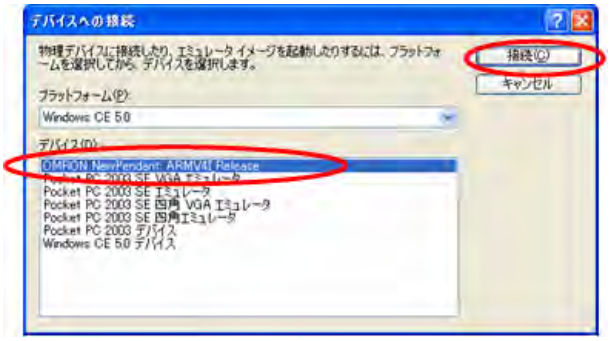
2. “for ExternalExe Debug” window appears.



3. Press [C# Debug] button. With this operation, the following processing will be executed and the pendant will be ready to connect a device.
   - Expansion of the memory size
     (Enable to download the user-customized application)
   - Execution of the device connection program
     (After execution, implement device connection within three minutes)
4. In Visual Studio on the desktop, press [Project] - [Property] to select [Debug] tab. On the window, also select "Debug" tab, and then, to "Program argument" section, input "argument:" which was indicated on the "for ExternalExe Debug".

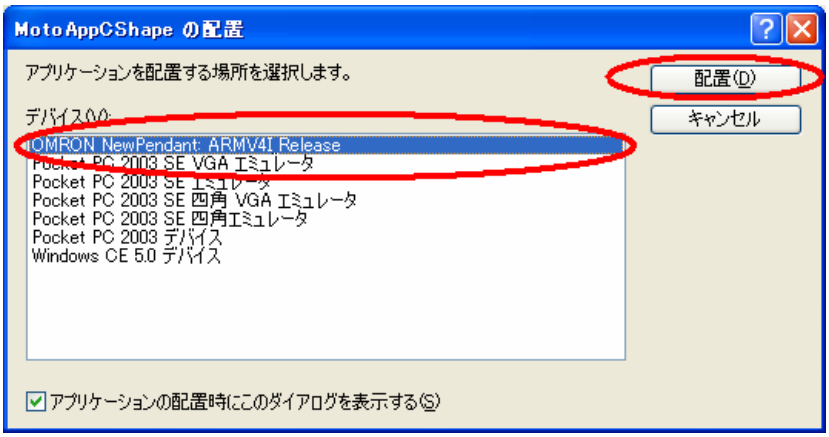


5. Press {Connect} button on "for EXternal Debug" window. It closes.
6. Connect a device via Visual Studio. Select [Tool] - [Connect to Device] on the pull-down menu. In "Connect to Device" dialog, select "OMRON NewPendant ARMV4I Release" and then push [Connect] button.



– When the following window is shown, debugging is ready.

7. Select {Menu} - {Debug} - {Start Debugging} to start the debugging operation.
8. If the following window is shown when debugging is started, select “OMRON NewPendant ARMV4I Release” and then press [Deploy] button.



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

Code name	Data configuration (Type)
EXE2YPP_CREATE_CMD	HWND hwnd : Window handle for message sending destination

<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

Code name	Data configuration (Type)
EXE2YPP_CLOSE_CMD	Not exist

<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.

11.1.3 Keymask

Code name	Data configuration (Type)
EXE2YPP_KEYMASK_CMD	EBYTE byte[8] : Keymask bit
	Not exist

<Description>

With this command, keys sent to the DX100/DX200/FS100 can be limited from the user-customized application. When setting “1” to the objective key, it is sent to the DX100/DX200/FS100.

<Key-Bit table>

	D7	D6	D5	D4	D3	D2	D1	D0
W0	X+/R+	ENTER	INSERT	FWD	SHIFT RIGHT	w+	Y+/B+	Z+/T+
W1	Y-/B-	X-/R-	MODIFY	TEST RUN	BWD	DELETE	Z-/T-	W-
W2	INTER LOCK	MOTION TYPE	Z+/U+	USAGE	Y+/L+	X+/S+	C+	INFORM LIST
W3	SHIFT LEFT	AUX	Z-/U-	EX AXIS	ROBOT	Y-/L-	X-/S-	C-
W4	.	0	7	ARROW DOWN	ARROW LEFT	1	2	4
W5	SERVO ON READY	5	-	3	8	FAST	SLOW	HIGH SPEED
W6	9	SELECT	AREA	CANCEL	ASSIST	PAGE	DIRECT OPEN	6
W7	Spare	Spare	COORD	SIMPLE MENU	MULTI	ARROW UP	ARROW RIGHT	MAIN MENU

Ex.

Set as follows when sending only [CANCEL] key to the DX100/DX200/ FS100.

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

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

11.1.4 Text Input Function Start

Code name	Data configuration (Type)	
EXE2YPP_KEYPAD_CMD	DWORD type	: key pad type (English/Japanese, etc.)
	DWORD maxlen	: limit of inputting character
	SCHAR buf[128]	: default character strings for keypad

<Description>

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

11.1.5 Switching Control

Code name	Data configuration (Type)	
EXE2YPP_CONTROL_CMD	DWORD target	1: programming pendant main software 2: user-customized application

<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

Code name	Data configuration (Type)	
EXE2YPP_GET_INFOAUTO_CMD	DWORD notifyingfo	Information to be notified (specified by bit)

<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

Code name	Data configuration (Type)
EXE2YPP_GET_INFO_CMD	Not exist

<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

Code name	Data configuration (Type)
EXE2YPP_GET_VERSION_CMD	Not exist

<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

Code name	Data configuration (Type)
YPPEXE_KEYPADRET_CMD	WORD okcancel 0=OK / 1=Cancel WCHAR buf[128] Input character string

<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

Code name	Data configuration (Type)
YPPEXE_CONTROL_CMD	DWORD target 1= Programming pendant main software 2=User-customized application

<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

Code name	Data configuration (Type)
YPPEXE_STATUS_CMD	DWORD enable[16] Valid information DWORD value[16] Status value

<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 a 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.

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 CF/ Touch operation invalid
9	: Teach/play mode
10	: Multi window mode
11-15	: (Unused)

Followings are the status value of each element.

Setting value	Meaning
0	R1
1	R2
2	R3
3	R4
4	R5
5	R6
6	R7
7	R8
8	B1
9	B2
10	B3
11	B4
12	B5
13	B6
14	B7
15	B8
16	S1
17	S2
18	S3
19	S4
20	S5
21	S6
22	S7
23	S8
24	S9
25	S10
26	S11
27	S12
28	S13
29	S14
30	S15
31	S16
32	S17
33	S18
34	S19
35	S20
36	S21
37	S22
38	S23
39	S24

1: Motion coordinate system

Setting value	Meaning
0	Link coordinate
1	Cartesian coordinate
2	Cylindrical coordinate
3	Tool coordinate
4	User coordinate
5	External reference point coordinate
6	Teaching line coordinate
7	(Unused)
8	(Unused)
9	(Unused)
10	(Unused)
11-74	User coordinate (1-64)

2: Manual speed

Setting value	Meaning
0	Inching
1	Low speed
2	Middle speed
3	HIgh speed

3: Security

Setting value	Meaning
0	Operation mode
1	Edit mode
2	Management mode
3	Safety mode *Only for DX200

4: Motion cycle

Setting value	Meaning
0	Continuous
1	1 cycle
2	Step

5: Executing status

Setting value	Meaning
0	Alarming
1	Stopped by emergency stop
2	Hold
3	Playback
4	Playback
5	Stopped

6: Page switch

Setting value	Meaning
0	Page does not switch
1	Page switches

7: Synchronized motion

Setting value	Meaning
0	Not synchronized
1	Synchronizing

8: Battery

Setting value	Meaning
0	Normal
1	Exhausted
2	Accessing to CF
3	Touch operation invalid
4	Exhausted and touch operation invalid

9: Teach/play mode

Setting value	Meaning
0	Initial status
1	Play mode
2	Teach mode
3	Play mode
4	Teach mode
5	Teach mode

10: Multi window indication mode

Setting value	Meaning
0	Single mode
1	Multi mode
2	Multi mode

Example

When changing the mode from teach to play

enable = {0.0.0.0.0.0.0.0.1.0.0.0.0.0.0.0}

value = {0.0.0.0.0.0.0.0.1.0.0.0.0.0.0.0}

When changing to return the mode from play to teach

enable = {0.0.0.0.0.0.0.0.1.0.0.0.0.0.0.0}

value = {0.0.0.0.0.0.0.0.2.0.0.0.0.0.0.0}

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

Code name	Data configuration (Type)	
YPPEXE_ALARM_CMD	DWORD status	1 = alarming / 2 = alarm released

<Description>

The programming pendant main software sends this command when alarm is occurred or released.

11.2.5 Message

Code name	Data configuration (Type)	
YPPEXE_MESSAGE_CMD	DWORD msgnum struct{ WORD code WCHAR buf[128] }*msgnum	Numbers of message Message code Message (volume of data for all messages)

<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

Code name	Data configuration (Type)	
YPPEXE_LANGUAGE_CMD	DWORD language	language code

<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

Code name	Data configuration (Type)	
YPPEXE_VERSION_CMD	WCHAR buff[16]	version information

<Description>

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

DX100/DX200/FS100 OPTIONS INSTRUCTIONS

FOR PROGRAMMING PENDANT CUSTOMIZATION FUNCTION

HEAD OFFICE
2-1 Kurosakishiroishi, Yahatanishi-ku, Kitakyushu 806-0004, Japan
Phone +81-93-645-7745 Fax +81-93-645-7746

YASKAWA America Inc. M Robotics Division
100 Automation Way, Miamisburg, OH 45342, U.S.A.
Phone +1-937-847-6200 Fax +1-937-847-6277

YASKAWA Nordic AB
Box 504 Verkstadsgatan 2, PO Box 504 SE-385 25 Torsås, Sweden
Phone +46-480-417-800 Fax +46-486-414-10

YASKAWA Europe GmbH Robotics Div.
Yaskawastrasse 1, 85391 Allershausen, Germany
Phone +49-8166-90-0 Fax +49-8166-90-103

YASKAWA Electric Korea Co., Ltd
9F, KYOBO Securities Bldg., 26-4, Yeoido-Dong Yeoungeungpo-ku, Seoul, Korea
Phone +82-2-784-7844 Fax +82-2-784-8495

YASKAWA Electric (Singapore) PTE Ltd.
151 Lorong Chuan, #04-02A, New Tech Park, Singapore 556741
Phone +65-6282-3003 Fax +65-6289-3003

YASKAWA Electric (Thailand) Co., Ltd.
252/246, 4th Floor. Muang Thai-Phatra Office Tower II Rachadaphisek Road,
Huaykwang Bangkok, 10320, Thailand
Phone +66-2-693-2200 Fax +66-2-693-4200

YASKAWA Shougang Robot Co. Ltd.
1015. Boxuenan Rd. Maluzhen, Jiading District, Shanghai, China
Phone +86-21-5950-3521 Fax +86-20-3878-0651

YASKAWA ELECTRIC CHINA Co., Ltd.
12F Carlton Building, No. 21-42 Huanghe Road, Shanghai 200003, China
Phone +86-21-5385-2200 Fax +86-21-5385-3299

YASKAWA Robotics India Ltd.
#426, Udyog Vihar, Phase- IV, Gurgaon, Haryana, India
Phone +91-124-475-8500 Fax +91-124-475-8542

YASKAWA
MOTOMAN ROBOTICS

MANUAL NO.
HW1481481 ①