MOTOCOM ES
OPERATION MANUAL

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

YASKAWA ELECTRIC CORPORATION

YASKAWA

MANDATORY

- **This manual explains the MOTOCOM ES. Read this manual carefully and be sure to understand its contents before operation.**
- **General items related to safety are listed in instruction manuals supplied with the manipulator. To ensure correct and safe operation, carefully read the instructions on safety 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.**
- **Software described in this manual is supplied against licensee only, with permission to use or copy under the conditions stated in the license. No part of this manual may be copied or reproduced in any form without written consent of YASKAWA.**

Notes for Safe Operation

Before using this product, read this manual and all the other related documents carefully to ensure knowledge about the product and safety, including all the cautions.
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.

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

Notation for Menus and Buttons

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

Item	Manual Designation
Menu	The menus displayed on screen are denoted with { }. ex. {TOOL}.
Button	The buttons, check boxes, radio buttons displayed on screen are denoted with [ ]. ex. [Close]; [Sync] check box; [Fast] radio button.

Description of the Operation Procedure

In the explanation of the operation procedure, the expression "Select • • • " means the following operations:

- To move the cursor to the object item and left-click on it with the mouse.
- To pick out the object item by the tab key and press the Enter key.
  (In case of selecting a menu, use arrow keys instead of the tab key to pick out the object item, then press the Enter key.)

Registered Trademark

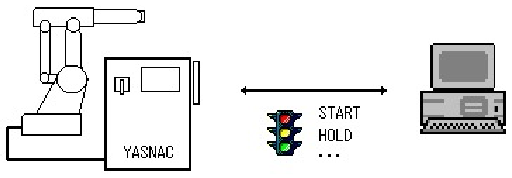
In this manual, names of companies, corporations, or products are trademarks, registered trademarks, or bland names for each company or corporation. The indications of (R) and TM are omitted.

1 INTRODUCTION

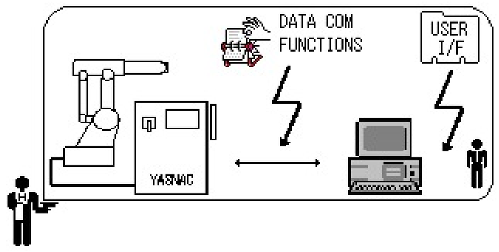
1.1 MOTOCOM ES

MOTOCOM ES is software for data transmission between a personal computer and YASKAWA industrial robot controller FS100, DX100. High-speed transmission can be achieved by connecting the FS100, DX100 to a LAN by using an Ethernet cable.
This software operates on a personal computer and has the following function.

Host Control Function
Can perform the following tasks easily according to the command from a personal computer.
- Reads robot status (current position, alarm, error, servo status, etc.) or controls the system (start, hold, job call, etc.)
- Reads or writes I/O signals.



Transmission Application Function
Supplies the following information so that the user can develop transmission applications between a robot and personal computer.
-Supplies data transmission functions between robot and personal computer. (MOTOCOMES.dll)
-Describes application procedures using sample programs including the above functions.



1.2 Hardware Requirements for MOTOCOM ES

OS	Microsoft Windows XP(32-bit) / Vista(32-bit) / 7(32-bit)*1
Reguired Memory	128 Mbyte or more
Hardware Disk Capacity for Installation	50 Mbyte or more
Display	Supported by MS-Windows
Robot Controller	FS100, DX100
Transmission Cable	Ethernet cable
Hardware Lock Key	Used under single user environment. For details, refer to the following section " 1.3 Hardware Lock Key ".

*1 MS-WindowsXP, MS-Windows Vista and MS-Windows 7 is a registerd trademark of Microsoft Coporation,USA.

NOTE

- The controller Data Transmission function, Ethernet function, Ethernet board, transmission cable, or the personal computer OS are not included in this package.
- To create a transmission application, a development tool such as Microsoft Visual C++ Ver6.0 or Visual Studio 2005 C# is required.

For softwares and devices, refer to the robot controller Operator's Manuals, Data Transmission Operator's Manual, Manual of Instructions for Binary Ethernet Server and manuals for MS-Windows, etc.

1.3 Hardware Lock Key

For proper operation, connect provided hardware lock key (USB type) to personal computer before using this software.

Check and execute <Check the computing environment> <Installation of driver> before connecting the key to USB port.

<Check the computing environment>
Multi-connection of USB type key is not available for one USB port because of hardware structure. Therefore, only one key should be connected to one USB port. When installing multiple offline software into one personal computer and multi-connectiong USB keys, use the personal computer which is provided same numbers of USB ports as the number of software to be installed.

<Installation of driver>

NOTE Please install the driver after detaching the all sentinel hardware key from the personal computer.

Execute "\SentinelDriver\Sentinel Protection Installer 7.6.3.exe" of installation CD-ROM. Refer to"\SentinelDriver\Manual\SafeNet_Sentinel_EndUser_Guide.pdf" for the details of installation.

NOTE

- Be sure to install the driver.
- When installing the driver, be sure to login in administrator mode in order to add files to system folder and input information in registry.
- If a key is connected to personal computer before installing the driver,the message concerning the driver is displayed. In this case, and detach the key from personal computer and then install the driver.
  If a key is connected to personal computer before installing the driver under Windows 95/98/NT4.0/2000/XP environment, Windows wizard ([Add New Hardware] Wizard) starts up. In this case, push [cancel], and detach the key from personal computer and then install the driver.
- When installing the driver under Windows NT4.0 environment, please install the driver located in the folder "\SentinelDriver\SSD5411\SSD5411-32bit.EXE" of installation CD-ROM.
  For the driver installation procedure, please consult the installation manuarl "\SentinelDriver\SSD5411\Manual\us\Readme.pdf".

Refer to " 6.2 Frequently-asked questions " for other countermeasures concerning hardware lock key.

2 SETUP

2.1 Installing MOTOCOM ES

When Motocom32 is installed, MOTOCOMES is installed also. For details on the way to setup, refer to the manual of MotoCom32.
When the setup is completed, [Host Control] and [MOTOCOMES Help] are registered under [MOTOCOMES] folder that appears by clicking the [Start] button in the task bar to select [All Program] , [Motoman] and then [MOTOCOM32].

NOTE To use Host Control of MOTOCOMES on Windows XP, It needs to install Microsoft .NET Framework 2.0 or more.

2.2 Environmental Settings for Use of Ethernet

The following configurations are required for Ethernet transmissions.

2.2.1 MOTOCOM ES Application Settings

■ parameter Setting

To communicate with the robot controller, the transmission parameter such as IP address must be set in each application.

2.2.2 Personal Computer Settings

Set the settings related to Ethernet transmissions, to the personal computer with the software installed.

■ Hardware settings

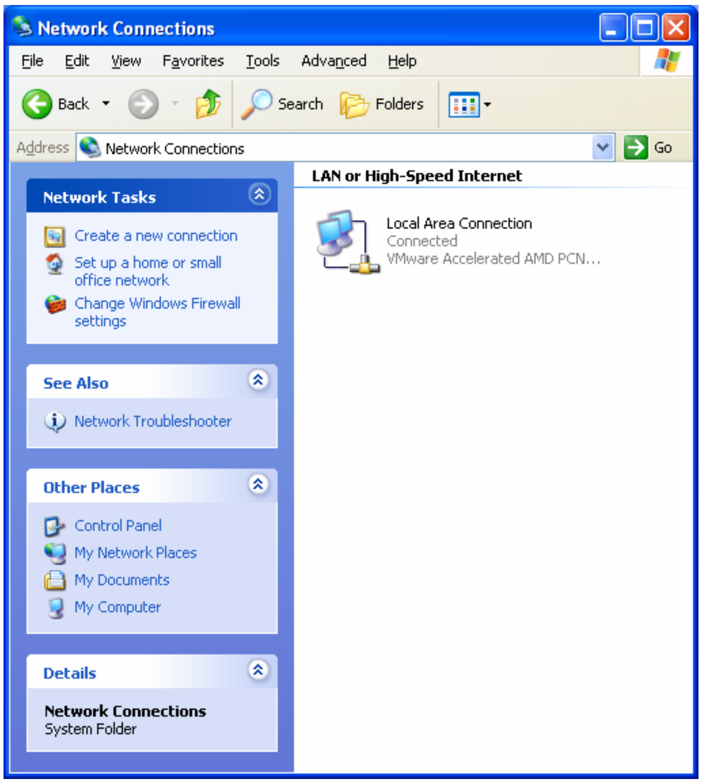
Before using the MOTOCOM ES, connect the Ethernet board to the personal computer and check if the Ethernet board operates correctly.
For connection methods, refer to the manual for the Ethernet board used.

■ Windows Network settings

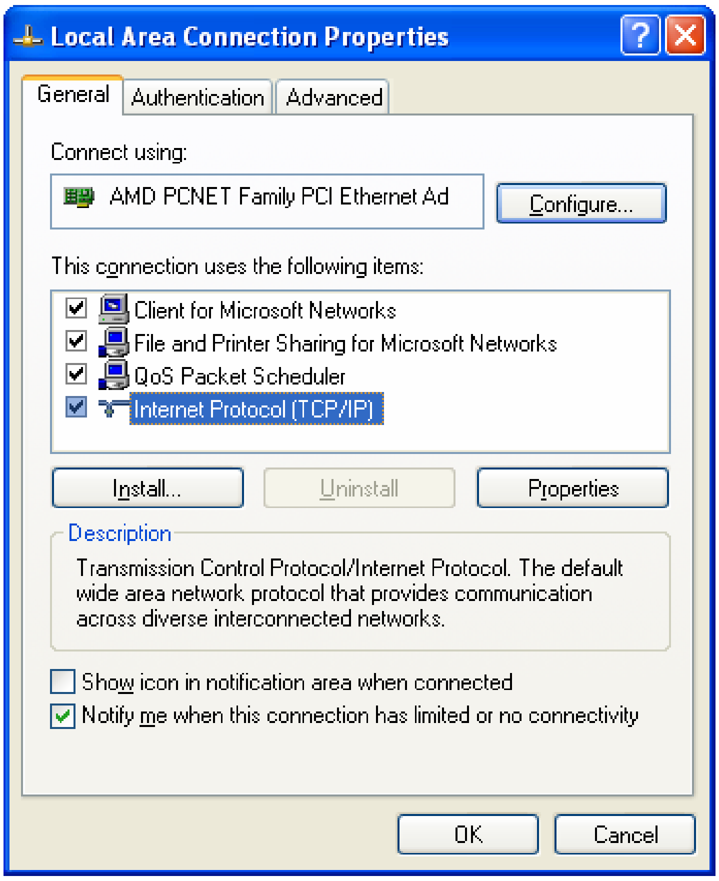
To communicate via the Ethernet, set the settings related to the Windows network. (The example below is based on Windows XP.)

1. Click the [Start] button in the task bar, select [Setting] and click [Control Panel]. If the [Control Panel] is "Category View", double-click the [Network and Internet Connections] category, and double-click the [Network Connections]. If the [Control Panel] is "Classic View", double-click the [Network Connections].

2. Select the network connection to use for data transmission, and select the "Properties" from the right click menu.



3. To Set the IP address and subnet mask for the personal computer, select [Internet Protocol (TCP/IP)] from the list and click the [Properties] button.



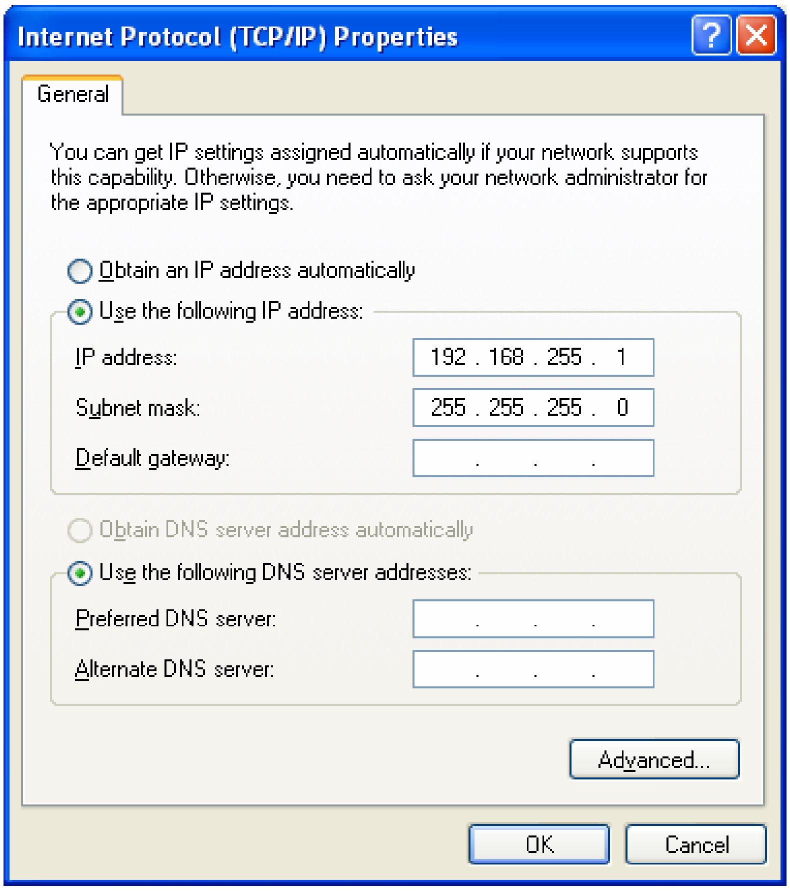
Connect using:
AMD PCNET Family PCI Ethernet Ad

This connection uses the following items:
- [✓] Client for Microsoft Networks
- [✓] File and Printer Sharing for Microsoft Networks
- [✓] QoS Packet Scheduler
- [✓] Internet Protocol (TCP/IP)

Description
Transmission Control Protocol/Internet Protocol. The default wide area network protocol that provides communication across diverse interconnected networks.

- [ ] Show icon in notification area when connected
- [✓] Notify me when this connection has limited or no connectivity

4. Input the value for the [IP address] and [Subnet mask] of the personal computer. For details of the settings of Default gateway and DNS server, refer to a Windows manual, to make proper settings for the application.


Internet Protocol (TCP/IP) Properties

General

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

○ Obtain an IP address automatically
● Use the following IP address:

IP address:	192 . 168 . 255 . 1
Subnet mask:	255 . 255 . 255 . 0
Default gateway:	. . .

○ Obtain DNS server address automatically
● Use the following DNS server addresses:

Preferred DNS server:	. . .
Alternate DNS server:	. . .

NOTE

The above values are examples only. When setting the IP address and subnet mask, input the correct numbers as advised by the network manager.
An incorrect setting such as assigning the same IP address to different personal computers may cause problems in communication.

2.2.3 Robot Controller Setting

■ Hardware settings

To communicate using Ethernet, for the FS100 use the Ethernet connector on the CPU201R board; for the DX100 use the Ethernet connector on the YCP01 board. To setup the IP address and submask, refer to the following manuals.

"FS100 Options: Instructions for Binary Ethernet Server"
"DX100 Options: Instructions for Binary Ethernet Server"

■ Communication parameter settings

Use the programming pendant in maintenance mode to set the communication parameters such as the IP address used by the controller. For details on the maintenance mode operations, refer to the following manuals.

Ethernet

Ethernet =	Used
IP address=	192.168.10.10(*)
Subnet mask=	255.255.255.0(*)
Default gateway=	192.168.10.1(*)
Server address=	0.0.0.0(*)

(*) The above values are examples only. Input the suitable values according to your network environment. There is no need to set the Server address. For details, refer to the following manuals.

"FS100 Options: Instructions for Binary Ethernet Server"
"DX100 Options: Instructions for Binary Ethernet Server"

■ Parameter settings

To establish communication between the robot controller and the personal computer, set the following parameters of the robot controller.

■ Command Remote Setting

In Order to transmit with the host control function, the remote command must be set to valid. Using the programming pendant, under [IN/OUT] a [PSEUDO INPUT] menu set the [CMD REMOTE SEL] to enabled or disabled.

For more details on these configurations, please refer to the following manual below.

"FS100 Options: Instructions for Data Transmission Function"
"DX100 Options: Instructions for Data Transmission Function"

2.2.4 Network Setting

To communicate with the robot controller using the Ethernet, the network must be set up correctly.
For details on how to setup the network, refer to the following manuals.

"FS100 Options: Instructions for Binary Ethernet Function"
"DX100 Options: Instructions for Ethernet Function"

2.3 Restriction

When using the MOTOCMES, pay attention to the following restrictions.

2.3.1 FS100, DX100 and Personal Computer Restrictions

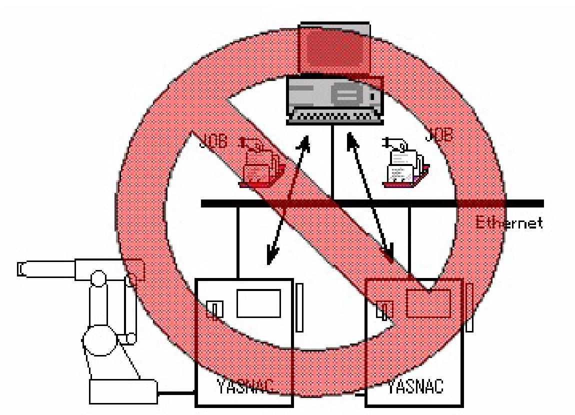
■ The port used for TCP/IP

The MOTOCOM ES uses UDP for the communication protocol. To communicate in UDP, the service identification numbers called "Port No" are used internally, while MOTOCOMES uses the port numbers 10040 and 10041 for the host control function. When these numbers overlap with the numbers used for other network devices, correct communication cannot be performed.
To use the MOTOCOM ES, be sure in advance that any network device in the same network does not use the above explained port numbers.

2.3.2 Personal Computer Restrictions

■ Same file access

The same file in the personal computer **cannot be accessed from different robot controllers simultaneously**.



2.3.3 FS100, DX100 Restrictions

■ CMOS batch storage

The BSC LIKE protocol and the FC1 protocol are available to communicate with external devices. The MOTOCOMES uses the BSC LIKE protocol for transmission. As the CMOS batch storage uses the FC1 protocol, CMOS batch storage is not available in the MOTOCOMES. For CMOS batch storage, use the CF / USB memory.

2.4 Execution of MOTOCOM ES Programs

To execute the MOTOCOMES program "Host Control", select the application to be executed from the start menu.

3 Operation Of Host Control Function

3.1 Host Control Function

The host control function consists of the following two functions, which can transmit according to the command of a personal computer.

Robot control function
I/O signal read/write function
File data transmission function
Macro function

For details, refer to " 3.3 Robot Control Function " " 3.4 Read/Write of I/O Signals " " 3.5 File Data Transmission Function " " 3.6 Macro function " in the following section.

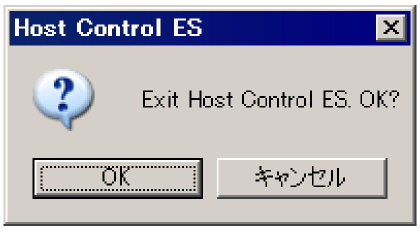
3.2 Startup and Exit

■ Startup

To start up the [Host Control ES], click the [Start: button in the task bar and point to [Program] and select [Motoman], [MotoComES], and then [Host Control].

■ Exit

Select the {Exit} command from the {File} menu, and the Host Control is ended.

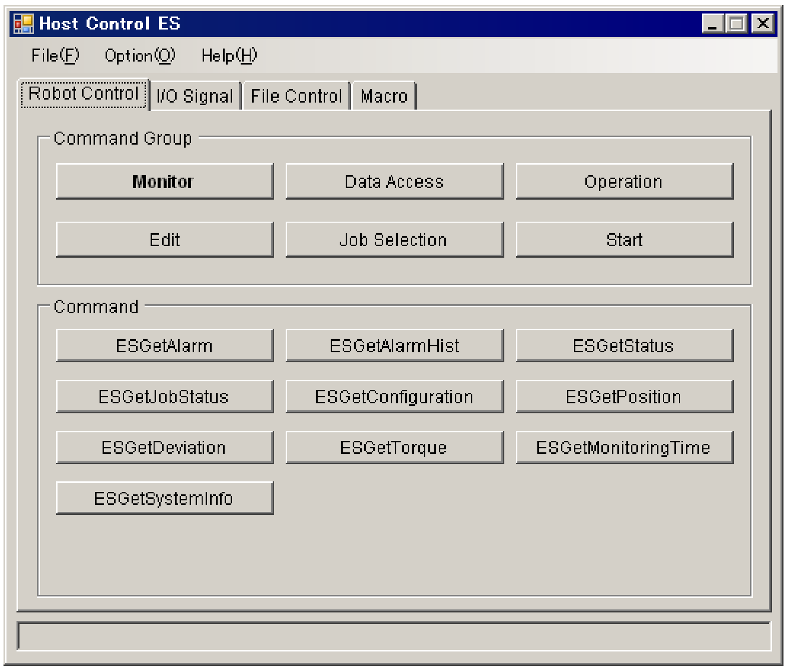


Exit Host Control ES. OK?

3.3 Robot Control Function

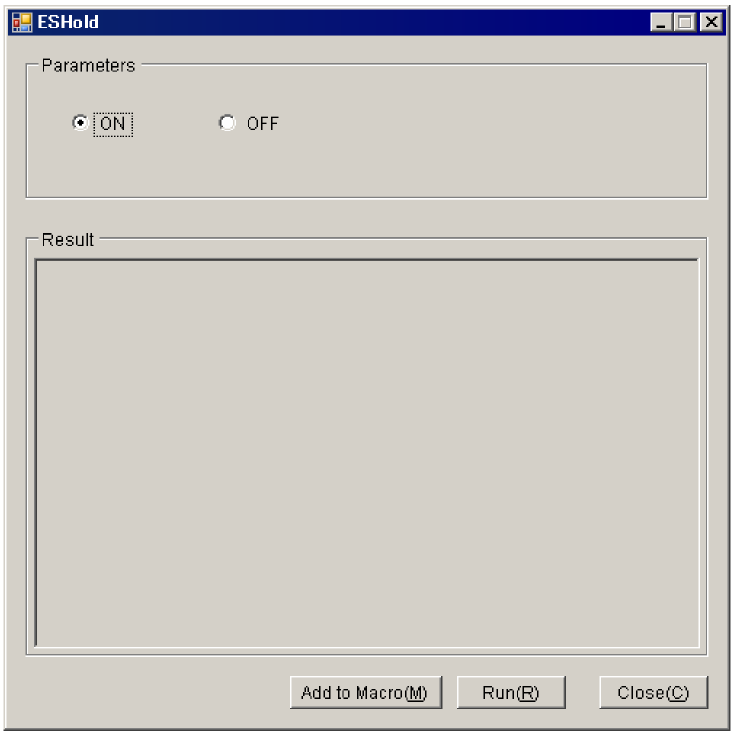
This function reads the robots status (current position, alarms, errors, servo, status, etc.) and controls the system (start, hold, job call, etc.). Each YASNAC transmission command can be executed individually.

1. Select [Host Control] and the [Host Control] Display appears.



2. Click the command button to display the list of the usable commands.

3. Click a command button to call up the display for that command.



Parameters: [✓ ON] OFF

4. Follow the instructions in the display to enter the reference parameters. (This is not necessary if there is no reference parameter.)
5. Click the [Execute] button to issue the command, and the response code and the response data from the controller appear.
   And, click the [Add to Macro] button, the command is added to Macro.
   Transmission commands are as follows. For details, refer to the robot controller Data Transmission Operator's Manual.

   "FS100 Options: Instructions for Data Transmission Function"
   “DX100 OPTIONS INSTRUCTIONS FOR DATA TRANSMISSION FUNCTION” .

		Command
Reading Status	Read, monitoring system	ESGetAlarm
		ESGetAlarmHist
		ESGetStatus
		ESGetJobStatus
		ESGetConfiguration
		ESGetPosition
		ESGetDeviation
		ESGetTorque
		ESGetMonitoringTime
		ESGetSystemInfo
		ESGetAlarmEx
		ESGetAlarmHistEx
	Read, data access system	ESGetVarData
		ESGetStrData
		ESGetPositionData
		ESGetBpexPositionData
		ESGetMultiData
Control of system	Operation system	ESReset
		ESCancel
		ESHold
		ESServo
		ESHlock
		ESCycle
		ESBDSP
	Editing system	ESSetVarData
		ESSetStrData
		ESSetPositionData
		ESSetBpexPositionData
		ESSetMultiData
	Job selection system	ESSelectJob
	Startup system	ESStartJob
		ESCartMove
		ESPulseMove

3.4 Read/Write of I/O Signals

Reads or writes the robot controller I/O signals.
The I/O signal read/write list and display are as follows.

■ List of I/O Signals that can be Read or Written

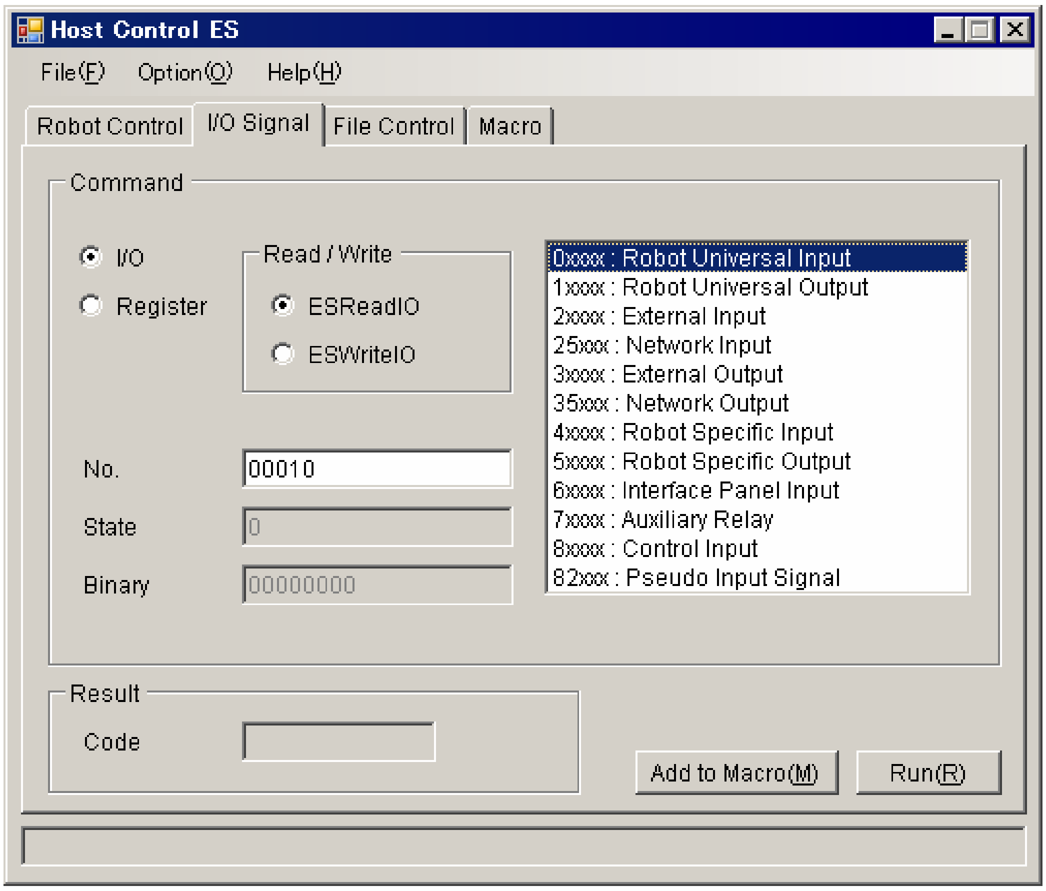
Signal	Range	Name	Read	Write
0xxx	00010-02567(2048)	Robot universal input	○	×
1xxx	10010-12567(2048)	Robot universal output	○	×
2xxx	20010-22567(2048)	External input	○	×
25xx	25010-27567(2048)	Network input	○	○
3xxx	30010-32567(2048)	External output	○	×
35xx	35010-37567(2048)	Network input	○	×
4xxx	40010-41607(1280)	Robot specific input (System)	○	×
5xxx	50010-52007 (1600)	Robot specific output (System)	○	×
6xxx	60010-60647(512)	I/F panel input	○	×
7xxx	70010-79997 (7992)	Auxiliary relay (System)	○	×
8xxx	80010-80647 (512)	Control status signal (System)	○	×
82xx	82010-82207(160)	Pseudo input signal (System)	○	×

■ List of Register that can be Read or Written

Range	Name	Read	Write
M000-M559 (560)	Robot universal register	○	○
M560-M599 (40)	Analog output register	○	×
M600-M639 (40)	Analog input register	○	×
M640-M999 (36)	System register	○	×

■ I/O signal read/write display

Check the [I/O] button, when reading/writing I/O signal. And check the [ESReadIO] button, when reading I/O signal. Choose the signal from the list, and input the number. Check the [ESWriteIO] button, when writing I/O signal. And choose the signal from the list, and input the number. Click the [Execute] button to issue the command, and the response code and response data from the controller appear. And, click the [Add to Macro] button, the command is added to Macro.



(● I/O) ○ Register

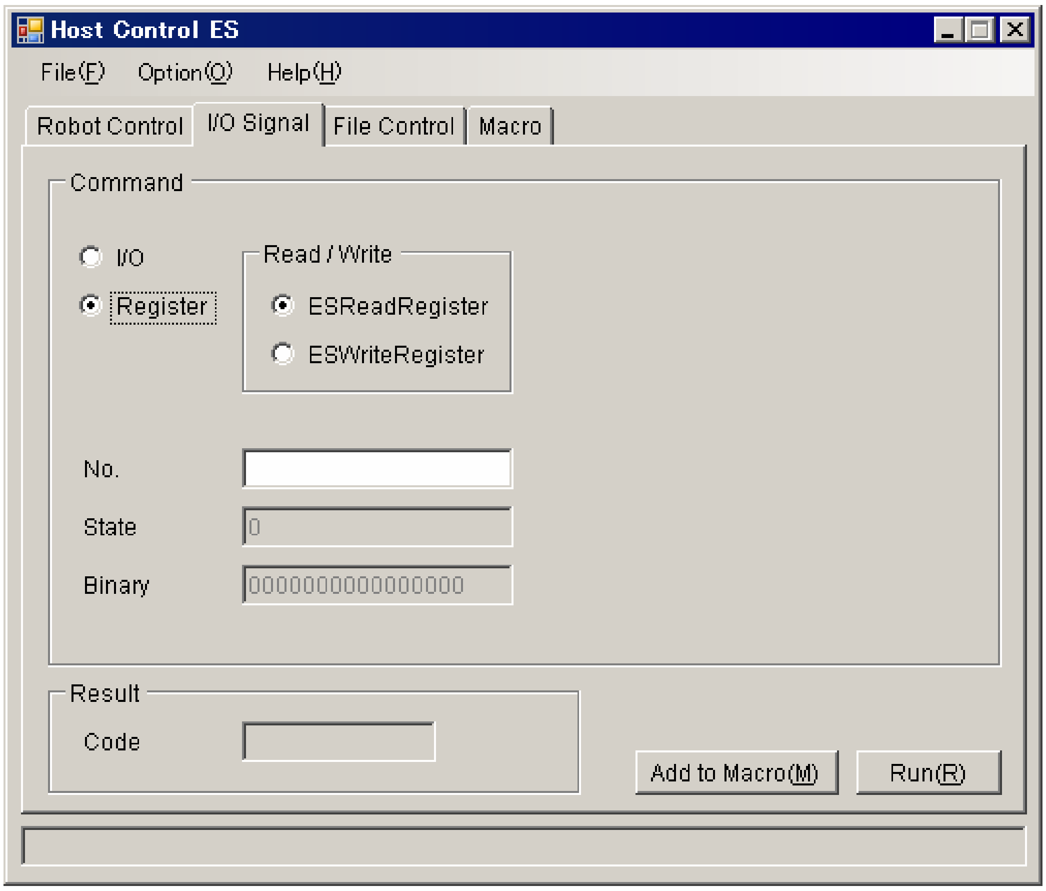
Read / Write: (● ESReadIO) ○ ESWriteIO

No.	00010
State	0
Binary	00000000
Code	

0xxxx : Robot Universal Input
1xxxx : Robot Universal Output
2xxxx : External Input
25xxx : Network Input
3xxxx : External Output
35xxx : Network Output
4xxxx : Robot Specific Input
5xxxx : Robot Specific Output
6xxxx : Interface Panel Input
7xxxx : Auxiliary Relay
8xxxx : Control Input
82xxx : Pseudo Input Signal

■ Register read/write display

Check the [Register] button, when reading/writing Register. And check the [ESReadRegister] button and input the number, when reading Register. Check the [ESWriteRegister] button and input the number, when writing Register. Click the [Execute] button to issue the command, and the response code and response data from the controller appear. And, click the [Add to Macro] button, the command is added to Macro.



Command

I/O [✓ Register]

Read / Write

[✓ ESReadRegister] ESWriteRegister

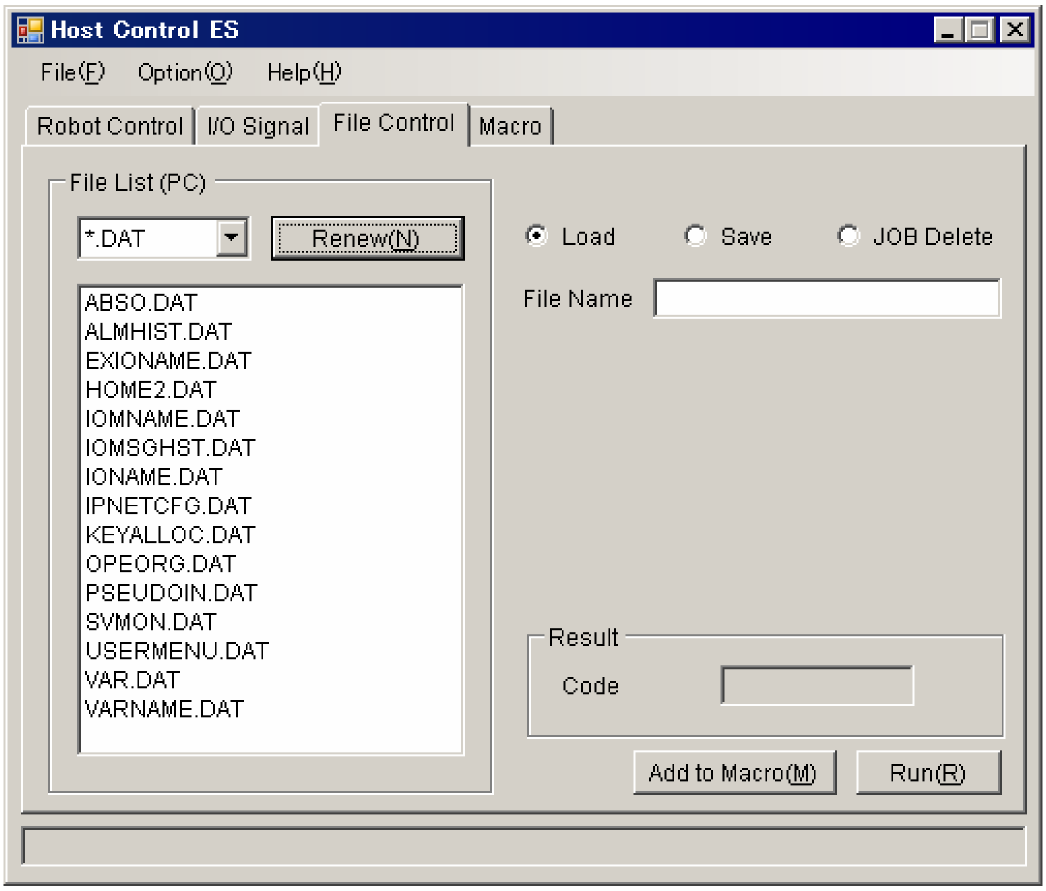
No.	
State	0
Binary	0000000000000000

Result

Code

3.5 File Data Transmission Function

This function transmits the file data and deletes the job. (The robot must be set to remote mode.)



File List (PC): *.DAT

ABSO.DAT
ALMHIST.DAT
EXIONAME.DAT
HOME2.DAT
IOMNAME.DAT
IOMSGHST.DAT
IONAME.DAT
IPNETCFG.DAT
KEYALLOC.DAT
OPEORG.DAT
PSEUDOIN.DAT
SVMON.DAT
USERMENU.DAT
VAR.DAT
VARNAME.DAT

[✓ Load] Save JOB Delete

File Name:

■ Fileload

Check the [Load] button, when sending the file data to the robot controller. Click the [Refresh] button, and the files are displayed, whose extension is equal to that of combo box. That files are in the folder specified by [Operation Environment] dialog. Choose the file name of the list, that name is displayed at [File name]. Click the [Execute] button to issue the command and send the file data of [File name], and the response code and response data from the controller appear. And, click the [Add to Macro] button, the command is added to Macro.

■ Filesave

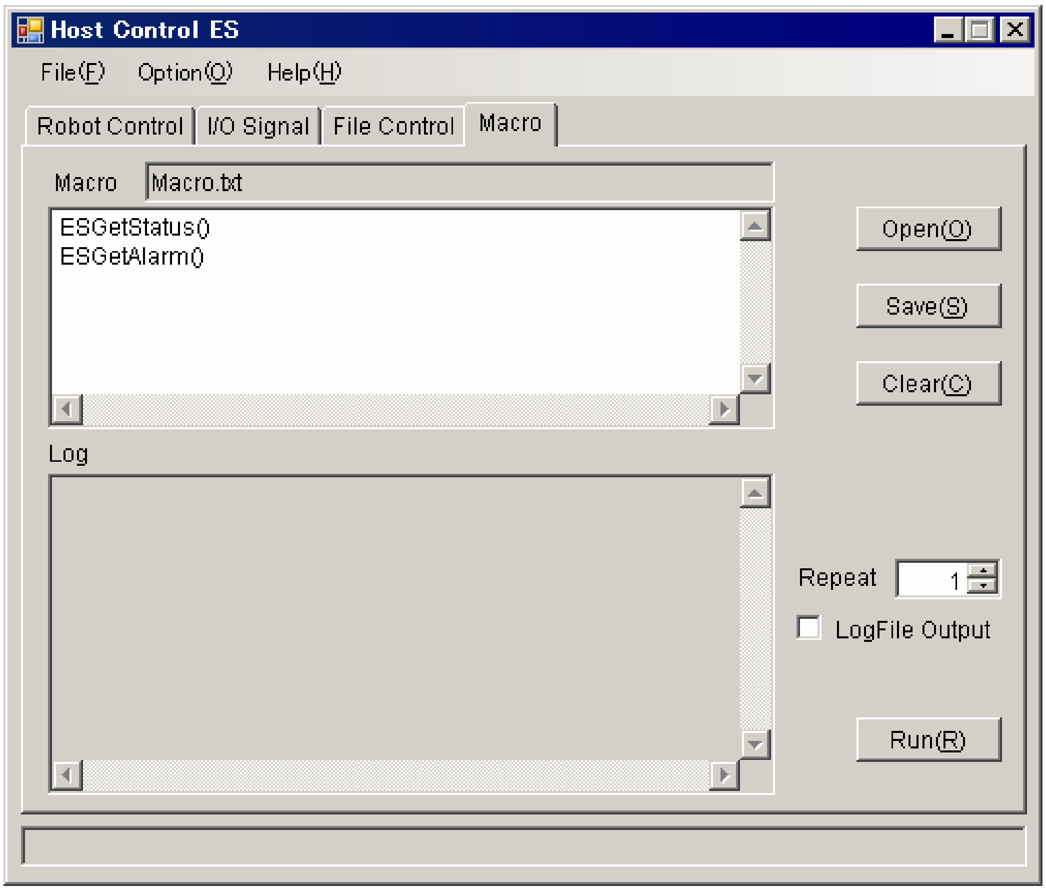
Check the [Save] button, when receiving the file data from the robot controller. Click the [Refresh] button, and the files are displayed, whose extension is equal to that of combo box. That files are at the controller. Choose the file name of the list, that name is displayed at [File name]. Click the [Execute] button to issue the command, and the response code and response data from the controller appear. Received files are in the folder specified by [Operation Environment] dialog. And, click the [Add to Macro] button, the command is added to Macro.

■ Delete Job

Check the [Delete Job] button, when deleting the file data of the robot controller. Click the [Refresh] button, and the files are displayed, whose extension is equal to that of combo box. That files are at the controller. Choose the file name of the list, that name is displayed at [File name]. Click the [Execute] button to issue the command, and the response code and response data from the controller appear. And, click the [Add to Macro] button, the command is added to Macro.

3.6 Macro function

Executes sequentially a number of commands registered as Macro. Click the [Open] button, the saved macro file can be opened. Click the [Save] button, the current macro can be saved. Click the [Clear] button, the current macro is deleted. The number of execution is set by [Repeat]. Check the [Log file output], when the log file is needed. The log file is saved at the folder specified by [Operation Environment] dialog. Click the [Execute] button to issue the command, and the response code and response data from the controller appear.



Macro: Macro.txt

```
ESGetStatus()
ESGetAlarm()
```

Repeat: 1

3.7 Environmental Settings

The environmental settings, define the operations of the host control.
Select {Option} - {Operation Environment}. Each item of the [Operation Environment] dialog box is set.

■ Setting transmission parameters

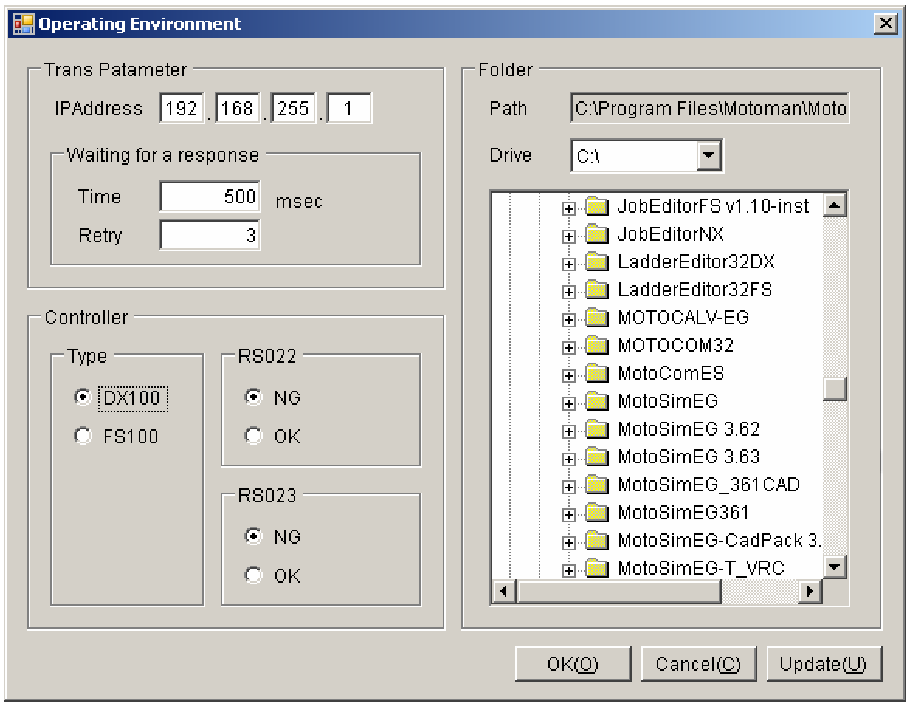
Enter the IP address assigned to the controller settings. And enter the time of waiting for a response and the number of retry.

■ Setting controller

Select the controller model for transmission. And set the parameter of the controller, "RS022" and "RS023".

■ Selecting a folder

Select the folder for communications.
Once the drive and folder are specified in this display, the file below the specified folder is to be transmitted when using a file command. And macro is saved at that folder.



Operating Environment

Trans Patameter
IPAddress 192 . 168 . 255 . 1
Waiting for a response
Time 500 msec
Retry 3

Controller
Type: [✓ DX100] FS100
RS022: [✓ NG] OK
RS023: [✓ NG] OK

Folder
Path C:\Program Files\Motoman\Moto
Drive C:\

JobEditorFS v1.10-inst
JobEditorNX
LadderEditor32DX
LadderEditor32FS
MOTOCALV-EG
MOTOCOM32
MotoComES
MotoSimEG
MotoSimEG 3.62
MotoSimEG 3.63
MotoSimEG_361CAD
MotoSimEG361
MotoSimEG-CadPack 3.
MotoSimEG-T_VRC

3.8 Version Information

Displays the Host Control version information.

4 CREATING A TRANSMISSION APPLICATION

4.1 Outline

This paragraph describes how to create an application so that the user can easily create a transmission application between the robot and the personal computer. This help explains how to create an application using the sample program (MS-Windows application development tool as the base "Visual C++" and "Visual C#") which employs a data transmission function (MS-Windows DLL file type: file name: MOTOCOMES.DLL).(Other languages can also be used.)

The program list of the sample program is in the "MOTOCOMES.DLL\Sample" folder below the MOTOCOMES installation directory.

NOTE
- Execute the sample program in the MOTOCOMES installation directory.
- YASKAWA is not responsible for anything that may result from using the sample program

4.2 Using Visual C++

4.2.1 Preparation

To create a transmission application, the following systems must be installed in the personal computer in advance.

(1) Microsoft Windows XP / Vista / *1

(2) Visual C++ Ver6.0 or more *2

*1 MS Windows XP / Vista / 7 is a registered trademark of Microsoft Corporation, U.S.A.

*2 Visual C++ is a registered trademark of Microsoft Corporation, U.S.A.

■ Creation of Skelton

Create a skelton using Visual C++ Ver.6.0 with the following procedure.

(1) Start up the Microsoft Development Studio and select "New" from the "File" menu to display the "New" display. Then click "Project Work Space" and then the [OK] button.

(2) Select the "Project" tab and then "MFC AppWizard (exe)."

(3) Enter the project name (in this example, input Test), and specify the folder where the project is to be created. Then click the [OK] button.

(4) Select "dialog base" as the type of the application to be created in "step 1," and click the [EXIT] button.

A source code to display a dialog box where only [OK] and [CANCEL] pushbuttons exist is created.

■ Definition of DLL Call

(1) Include "MOTOCOMES.h" attached to the MOTOCOMES application using the dialog class source file (TestDlg.cpp).

```
#include "MOTOCOMES.h"
```

(2) Copy the "MOTOCOMES.lib" file, the "MOTOCOMES.h" file and the data transmission function (Windows DLL file type, file name: MOTOCOMES.DLL and Motork.DLL, Motolkr.DLL) to the directory where the project exists.

(3) Click the "Build" and then the "Setting" buttons, and open the "link" tab in the "Set Project" dialog box. Specify the "MOTOCOMES.lib" file in the "Object/Library Module" setting column, and click the [OK] button.

The MOTOCMES functions can be used in the file where "MOTOCOMES.h" is incuded.

NOTE The library file (**file name: MOTOCOMES.lib**) and the included file (**file name: MOTOCOMES.h**) are in the MOTOCOMES installation directory.

4.2.2 How to Create a transmission application

This paragraph explains a simple program, as an example, which sends / receives a job (TEST.JBI) to / from the controller.

■ Editing with a Dialog Box

Edit the following with the created dialog box.
Open the IDD_TEST_DIALOG dialog box.

(1) Delete the [OK] push button and the [Cancel] push button which was created by default.
(2) Create a push button for sending, and name the caption "LOAD" and the ID "IDC_LOADFILE".
(3) Create a push button for receiving, and name the caption "SAVE" and the ID "IDC_SAVEFILE".

■ Addition of Functions and Variables

(1) Create a function "CTestDlg::OnLoadfile" for BN_CLICKED message in Class Wizard using the [LOAD] push button (IDC_LOADFILE).
(2) Create a function "CTestDlg::OnSavefile" for BN_CLICKED message in Class Wizard using the [SAVE] push button (IDC_ SAVE FILE).
(3) Write the code in each function.
    CTestDlg::OnLoadfile function
    CTestDlg::OnSavefile function

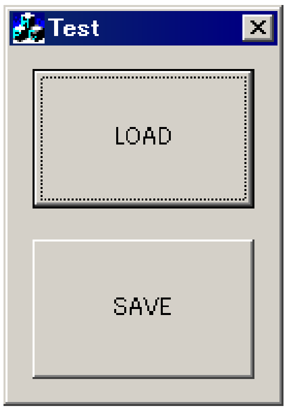
In each function, the storage passing of IP address is specified, when ESOpen() function is called. And, the storage passing of full path of file or folder is specified, when ESLoadFIle() / ESSaveFile() is called.
Please change according to customer's environment.

" CTestDlg::OnLoadfile () " to select the data part (program list) of the above function. Use "Copy" to copy this section to CTestDlg::OnLoadfile() function.
Repeat for CTestDlg::OnSavefile (" CTestDlg::OnSavefile () ").

■ Creation and Execution of EXE File

Execute "Build" in the Visual C++ Build menu to create a execution enabled module. By putting this module in the same directory as the job to be sent or received and executing it, the job can be sent or received.

NOTE The MOTOCOMES installation directory contains data transmission functions (Windows DLL file type, **file name: MOTOCOMES.DLL and Motolk.DLL Motolkr.DLL**). When executing an application, copy the functions to the directory where the module to be executed is created.



4.3 Using Visual C#

4.3.1 Preparation

To create a transmission application, the following systems must be installed in the personal computer in advance.

(1) Microsoft Windows XP/2000/ 7 *1

(2) Visual Studio 2005 or more *2

*1 MS Windows XP/2000/7 is a registered trademark of Microsoft Corporation, U.S.A.

*2 Visual C# is a registered trademark of, Microsoft Corporation U.S.A.

■ Creation of Project

Start up the Microsoft Visual Studio and select "New" from the "File" menu to display the "New" display. Then click "Visual C#" and "Windows application" and then the [OK] button.

■ Reference configuration of Library

To use "MOTOCOMES.DLL" in Visual Studio C#, "MOTOCOMES_CS.DLL" must be referenced.
Select "Add Reference" from "Project" menu to display the "Add Reference" display. Then select "Reference" tag, and select "MOTOCOMES_CS.DLL" in the "MOTOCOMES" folder, and "MOTOCOMES_CS.DLL" is added to the project.
To import the data types defined in the namespace of "MOTOCOMES_CS.DLL", descript the following using directive.

```
using MotoComES_CS
```

4.3.2 How to Create a transmission application

This paragraph explains a simple program, as an example, which sends / receives a job (TEST.JBI) to / from the controller.

■ Creation of Form Module

Create the following module.

(1) Form to be program display
On this form, create the following controls.

(2) Send button (control name: "CmdLoadFile", caption name: "LOAD")

(3) Receive button (control name: "CmdSaveFile"?caption name: "SAVE")

When the control is created, describe the event procedure for each button.

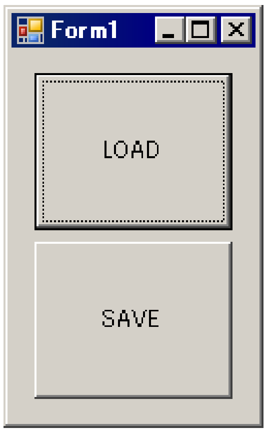
```
private void CmdLoadFile_Click( object sender, EventArgs e)
private void CmdSaveFile_Click( object sender, EventArgs e)
```

In each function, the storage passing of IP address is specified, when ESOpen() function is called. And, the storage passing of full path of file or folder is specified, when ESLoadFIle() / ESSaveFile() is called.
Please change according to customer's environment.

" Cmd_LoadFile_Click () " to select the data part (program list) of the above function. Use "Copy" to copy this section to CmdLoadFile_Click() function.
Repeat for CmdLoadSave_Click ("CmdLoadSave_Click()").

■ Creation and Execution of EXE File

Execute "Build" in the Visual Studio "Build" menu to create a execution enable module. By putting the job to be sent/received in the same folder where this module is, and executing this module, the job can be sent/received.



4.4 Each Function Program List

■ CTestDlg::OnLoadfile ()

```
void CTestDlg::OnLoadfile()
{
   //A variable for result
   long result = 0;
   //Handle
   HANDLE handle;

   //Executing ESOpen(1(DX100), IP Address, HANDLE) command
   result = ESOpen(1, "192.168.255.1", &handle);

   //Failure
   if (result != 0)
   {
      AfxMessageBox("ESOpen is failed.");
      return;
   }

   // Executing ESLoadFile(HANDLE, Full path of file) command
   result = ESLoadFile(handle, "C:\\TEMP\\TEST.JBI");

   //Failure
   if (result != 0)
   {
      AfxMessageBox("ESLoadFile is failed.");
      return;
   }

   //Executing ESClose(HANDLE) command
   result = ESClose(handle);

   //Failure
   if (result != 0)
   {
      AfxMessageBox("ESClose is failed.");
      return;
   }

   return;
}
```

SUPPLE-MENT

• Double underline indicates transmission functions belonging to the MOTOCOM ES.
• AfxMessageBox : VisualC++ function

■ CTestDlg::OnSavefile ()

```
void CTestDlg::OnSavefile()
{
   //A variable for result
   long result = 0;
   //Handle
   HANDLE handle;

   //Executing ESOpen(1(DX100), IP Address, HANDLE) command
   result = ESOpen(1, "192.168.255.1", &handle);

   //Failure
   if (result != 0)
   {
      AfxMessageBox("ESOpen is failed.");
      return;
   }

   //Executing ESSaveFile(HANDLE, Path of folder to save, Job name) command
   result = ESSaveFile(handle, "C:\\TEMP", "TEST.JBI");

   //Failure
   if (result != 0)
   {
      AfxMessageBox("ESSaveFile is failed.");
      return;
   }

   //Executing ESClose(HANDLE) command
   result = ESClose(handle);

   //Failure
   if (result != 0)
   {
      AfxMessageBox("ESClose is failed.");
      return;
   }

   return;
}
```

SUPPLE-MENT

- Double underline indicates transmission functions belonging to the MOTOCOM ES.
- AfxMessageBox : VisualC++ function

■ Cmd_LoadFile_Click ()

```
private void Cmd_LoadFile_Click(object sender, EventArgs e)
{
   //A variable for result
   long result = 0;
   //Handle
   IntPtr handle = new IntPtr();
   //A variable for Counting of string length
   int iByteCount = 0;

   //Convert String of IP address to Byte array
   iByteCount = MotoComES._ECode.GetByteCount("192.168.255.1") + 1;
   byte[] bIPAdd = MotoComES.StringToByteArray("192.168.255.1", iByteCount);

   //Executing ESOpen(1(DX100), IP Address, HANDLE) command
   result = MotoComES.ESOpen(1, ref bIPAdd[ 0 ], ref handle);

   //Failure
   if (result != 0)
   {
      MessageBox.Show("ESOpen is failed.");
      return;
   }

   //Convert String of Full path of File to Byte array
   iByteCount = MotoComES._ECode.GetByteCount("C:\\TEMP\\TEST.JBI") + 1;
   byte[] fPath = MotoComES.StringToByteArray("C:\\TEMP\\TEST.JBI", iByteCount);

   //Executing ESLoadFile(HANDLE, Full path of File) command
   result = MotoComES.ESLoadFile(handle, ref fPath[0]);

   //Failure
   if (result != 0)
   {
      MessageBox.Show("ESLoadFile is failed.");
      return;
   }

   //Executing ESClose(HANDLE) command
   result = MotoComES.ESClose(handle);

   //Failure
   if (result != 0)
   {
      MessageBox.Show("ESClose is failed.");
      return;
   }
```

```
    }
```

SUPPLE-MENT

- Double underline indicates transmission functions belonging to the MOTOCOM ES.
- Single underline indicates function defined by MOTOCOMES_CS.DLL.
  MotoComES._ECode.GetByteCount() : Function of MOTOCOMES_CS
  Return the length of string.
  MotoComES.StringToByteArray() : Function of MOTOCOMES_CS
  Convert String data to Byte array.
- MessageBox.Show() : Function of Visual Studio C#
  Display the message in the dialog box, and wait for clicking the button.

■ Cmd_SaveFile_Click ()

```
private void Cmd_SaveFile_Click(object sender, EventArgs e)
{
    //A variable for result
    long result = 0;
    //Handle
    IntPtr handle = new IntPtr();
    //A variable for Counting of string length
    int iByteCount = 0;

    //Convert String of IP address to Byte array
    iByteCount = MotoComES._ECode.GetByteCount("192.168.255.1") + 1;
    byte[] bIPAdd = MotoComES.StringToByteArray("192.168.255.1", iByteCount);

    //Executing ESOpen(1(DX100), IP Address, HANDLE) command
    result = MotoComES.ESOpen(1, ref bIPAdd[0], ref handle);

    //Failure
    if (result != 0)
    {
        MessageBox.Show("ESOpen is failed.");
        return;
    }

    //Convert String of Folder path to Byte array
    iByteCount = MotoComES._ECode.GetByteCount("C:\\TEMP") + 1;
    byte[] sPath = MotoComES.StringToByteArray("C:\\TEMP", iByteCount);

    //Convert String of File name to Byte array
    iByteCount = MotoComES._ECode.GetByteCount("TEST.JBI") + 1;
    byte[] fName = MotoComES.StringToByteArray("TEST.JBI", iByteCount);

    //Executing ESSaveFile(HANDLE, Path of folder to save, Job name) command
    result = MotoComES.ESSaveFile(handle, ref sPath[0], ref fName[0]);
```

```
        //Failure
        if (result != 0)
        {
            MessageBox.Show("ESSaveFile is failed.");
            return;
        }

        //Executing ESClose(HANDLE) command
        result = MotoComES.ESClose(handle);

        //Failure
        if (result != 0)
        {
            MessageBox.Show("ESClose is failed.");
            return;
        }
    }
```

SUPPLE-MENT

- Double underline indicates transmission functions belonging to the MOTOCOM ES.
- Single underline indicates function defined by MOTOCOMES_CS.DLL.
  - MotoComES._ECode.GetByteCount() : Function of MOTOCOMES_CS
    Return the length of string.
  - MotoComES.StringToByteArray() : Function of MOTOCOMES_CS
    Convert String data to Byte array.
- MessageBox.Show() : Function of Visual Studio C#
  Display the message in the dialog box, and wait for clicking the button.

5 COMMUNICATION TRANSMISSION

5.1 Outline

MOTOCOMES.DLL is a transmission library that controls the data transmission function of the FS100, DX100 on a personal computer. This library is composed in the form of Microsoft Windows DLL (Dynamic Link Library)..

NOTE MOTOCOMES.DLL is located below the MOTOCOMES installation directory. When a transmission application is created, copy this file to the same directory as the application. MOTOCOMES.H and MOTOCOMES.LIB files are provided in the MOTOCOMES installation directory. Use these files when a transmission application is created in C++ language.

Transmission library has the following functions.

- Robot control function
- I/O signal read/write function
- File data transmission functions
- Other functions

5.2 Robot Control Function

Reads the robot status (current position, alarm, error, servo, status, etc.) and controls the system (start, hold, job call, etc.)
The following fuctions are available.

Status Read	**System Control**
ESGetAlarm	ESReset
ESGetAlarmHist	ESCancel
ESGetStatus	ESHold
ESGetJobStatus	ESServo
ESGetConfiguration	ESHlock
ESGetPosition	ESCycle
ESGetDeviation	ESBDSP
ESGetTorque	ESSetVarData1
ESGetMonitoringTime	ESSetVarData2
ESGetSystemInfo	ESSetStrData
ESGetVarData1	ESSetPositionData
ESGetVarData2	ESSetBpexPositionData
ESGetStrData	ESSetVarDataMB
ESGetPositionData	ESSetVarDataMI
ESGetBpexPositionData	ESSetVarDataMD
ESGetVarDataMB	ESSetVarDataMR
ESGetVarDataMI	ESSetStrDataM
ESGetVarDataMD	ESSetPositionDataM
ESGetVarDataMR	ESSetBpexPositionDataM
ESGetStrDataM	ESSelectJob
ESGetPositionDataM	ESStartJob
ESGetBpexPositionDataM	ESCartMove
ESGetAlarmEx	ESPulseMove
ESGetAlarmHistEx	

■ ESGetAlarm

Reads a current error data.

FORMAT

LONG ESGetAlarm(HANDLE handle, ESAlarmList* alarmList);

ARGUMENTS

[in] handle	Target handle value
[out] alarmList	Alarm list storage pointer

ESAlarmList
Alarm list structure

FORMAT

```
#define Length_of_AlarmList (4)

typedef struct
{
   ESAlarmData data[Length_of_AlarmList];
} ESAlarmList;
```

MEMBER <data[Length_of_AlarmList]> Alarm data (Max number = 4)

ESAlarmData Alarm data structure

FORMAT :

```
#define Length_of_Time (16)
#define Length_of_Name (32)

typedef struct
{
   LONG  alarmCode;
   LONG  alarmData;
   LONG  alarmType;
   CHAR  alarmTime[Length_of_Time+1];
   CHAR  alarmName[Length_of_Name+1];
} ESAlarmData;
```

MEMBER :

<alarmCode>	Alarm code
<alarmData>	Alarm data
<alarmrType>	Alarm data type
<alarmTime[Length_of_Time+1]>	Time of alarm occurrence (Max size = 16)
<alarmName[Length_of_Name+1]>	Alarm Name (Max size = 32)

RETURN VALUE

0	: Normal completion
Others	: Error codes

REFERENCE

"ESGetAlarmHist" "ESGetStatus" "ESGetAlarmEx" "ESGetAlarmHistEx"

■ ESGetAlarmHist

Reads an alarm history of the specified alarm number.

FORMAT

LONG ESGetAlarmHist(HANDLE handle, LONG alarmHistNo, ESAlarmData* alarmData);

ARGUMENTS

[in] handle	Target handle value
[in] alarmHistNo	Alarm number

Value	Explanation
1～100	Major alarm
1001～1100	Minor alarm
2001～2100	User alarm（system）
3001～3100	User alarm（user）
4001～4100	off-line alarm

[out] alarmData	Alarm data storage pointer

ESAlarmData

Alarm data structure

FORMAT

```
#define Length_of_Time (16)
#define Length_of_Name (32)

typedef struct
{
    LONG  alarmCode;
    LONG  alarmData;
    LONG  alarmType;
    CHAR  alarmTime[Length_of_Time+1];
    CHAR  alarmName[Length_of_Name+1];
} ESAlarmData;
```

MEMBER

<alarmCode> Alarm code

<alarmData> Alarm data

<alarmrType> Alarm data type

<alarmTime[Length_of_Time+1]> Time of alarm occurrence (Max size =16)

<alarmName[Length_of_Name+1]> Alarm Name (Max size = 32)

RETURN VALUE

0	: Normal completion
0xA001	: Out of alarm number range
Others	: Error codes

REFERENCE

"ESGetAlarm" "ESGetStatus" "ESGetAlarmEx" "ESGetAlarmHistEx"

■ ESGetStatus

Reads the current status of controller.

FORMAT

LONG ESGetStatus(HANDLE handle, ESStatusData* statusData);

ARGUMENTS

[in] handle	Target handle value
[out] statusData	Status data storage pointer

ESStatusData

Status data structure

FORMAT

```
typedef struct
{
    LONG  status1;
    LONG  status2;
} ESStatusData;
```

MEMBER <status1> status1

bitValue	Explanation
D00	Step
D01	1-cycle
D02	Auto operation
D03	Operating
D04	Operation at safe speed
D05	Teach
D06	Play
D07	Command remote

<status2> status2

bitValue	Explanation
D00	
D01	Hold (Programming pendant hold)
D02	Hold (External hold)
D03	Hold (Command hold)
D04	Alarm occurred
D05	Error occurred
D06	Servo ON
D07	

RETURN VALUE

0 : Normal completion
Others : Error codes

REFERENCE

"ESGetAlarm" "ESGetAlarmHist" "ESGetAlarmEx" "ESGetAlarmHistEx"

■ ESGetJobStatus

Reads a current job information of the specified task.

FORMAT

LONG ESGetJobStatus(HANDLE handle, LONG taskNo, ESJobStatusData jobStatus-Data);

ARGUMENTS

[in] handle	Target handle value
[in] taskNo	Task number

Value	Explanation	Value	Explanation
1	Master task	9	Sub task 8
2	Sub task 1	10	Sub task 9
3	Sub task 2	11	Sub task 10
4	Sub task 3	12	Sub task 11
5	Sub task 4	13	Sub task 12
6	Sub task 5	14	Sub task 13
7	Sub task 6	15	Sub task 14
8	Sub task 7	16	Sub task 15

[out] jobStatusData	Job status data storage pointer

ESJobStatusData
Job status data structure

FORMAT

```
#define Length_of_Name (32)

typedef struct
{
    CHAR  jobName[Length_of_Name+1];
    LONG  lineNo;
    LONG  stepNo;
    LONG  speedOverride;
} ESJobStatusData;
```

MEMBER

<jobName[Length_of_Name+1]> Job Name (Max size = 32)

<lineNo> Line number

<stepNo> Step Number

<speedOverRide> Speed override value

RETURN VALUE

0	: Normal completion
0xA001	: No task
Others	: Error codes

REFERENCE

"ESSelectJob"

■ ESGetConfiguration

Reads a current axes configuration of the specified control group.

FORMAT

LONG ESGetConfiguration(HANDLE handle, LONG ctrlGrp, ESConfigurationData* configData);

ARGUMENTS

[in] handle	Target handle value
[in] ctrlGrp	Control group

Value	Explanation	Value	Explanation
1	R1 (Robot 1) (Pulse)	101	R1 (Robot 1) (Coordinate)
2	R2 (Robot 2) (Pulse)	102	R2 (Robot 2) (Coordinate)
3	R3 (Robot 3) (Pulse)	103	R3 (Robot 3) (Coordinate)
4	R4 (Robot 4) (Pulse)	104	R4 (Robot 4) (Coordinate)
5	R5 (Robot 5) (Pulse)	105	R5 (Robot 5) (Coordinate)
6	R6 (Robot 6) (Pulse)	106	R6 (Robot 6) (Coordinate)
7	R7 (Robot 7) (Pulse)	107	R7 (Robot 7) (Coordinate)
8	R8 (Robot 8) (Pulse)	108	R8 (Robot 8) (Coordinate)
11	B1 (Base 1) (Pulse)	111	B1 (Base 1) (Coordinate)
12	B2 (Base 2) (Pulse)	112	B2 (Base 2) (Coordinate)
13	B3 (Base 3) (Pulse)	113	B3 (Base 3) (Coordinate)
14	B4 (Base 4) (Pulse)	114	B4 (Base 4) (Coordinate)
15	B5 (Base 5) (Pulse)	115	B5 (Base 5) (Coordinate)
16	B6 (Base 6) (Pulse)	116	B6 (Base 6) (Coordinate)
17	B7 (Base 7) (Pulse)	117	B7 (Base 7) (Coordinate)
18	B8 (Base 8) (Pulse)	118	B8 (Base 8) (Coordinate)
21	S1 (Station 1) (Pulse)		
22	S2 (Station 2) (Pulse)		
23	S3 (Station 3) (Pulse)		
24	S4 (Station 4) (Pulse)		
25	S5 (Station 5) (Pulse)		
26	S6 (Station 6) (Pulse)		
27	S7 (Station 7) (Pulse)		
28	S8 (Station 8) (Pulse)		
29	S9 (Station 9) (Pulse)		
30	S10 (Station 10)(Pulse)		
31	S11 (Station 11)(Pulse)		
32	S12 (Station 12)(Pulse)		
33	S13 (Station 13)(Pulse)		
34	S14 (Station 14)(Pulse)		
35	S15 (Station 15)(Pulse)		
36	S16 (Station 16)(Pulse)		
37	S17 (Station 17)(Pulse)		
38	S18 (Station 18)(Pulse)		
39	S19 (Station 19)(Pulse)		
40	S20 (Station 20)(Pulse)		
41	S21 (Station 21)(Pulse)		
42	S22 (Station 22)(Pulse)		
43	S23 (Station 23)(Pulse)		
44	S24 (Station 24)(Pulse)		

[out] configData Axis configuration data storage pointer

ESConfigurationData
Axis configuration data structure

FORMAT
```
#define Number_of_Axis (8)

typedef struct
{
   CHAR  configurations[Number_of_Axis];
} ESConfigurationData;
```

MEMBER <configuration[Number_of_Axis]> Axis configuration data
(Max number = 8)

Array	R* : Pulse	B*/S* : Pulse	R*/B* : Coordinate
configurations[0]	1st axis"S"	1st axis"1"	"X"
configurations[1]	2nd axis"L"	2nd axis"2"	"Y"
configurations[2]	3rd axis"U"	3td axis"3"	"Z"
configurations[3]	4th axis"R"	4th axis"4"	"Rx" (R only)
configurations[4]	5th axis"B"	5th axis"5"	"Ry" (R only)
configurations[5]	6th axis"T"	6th axis"6"	"Rz" (R only)
configurations[6]	7th axis"E"	7th axis"7"	"Re" (R only)
configurations[7]	8th axis"8"	8th axis"8"	

RETURN VALUE

0 : Normal completion
0xA001 : No control group
Others : Error codes

■ ESGetPosition

Reads a current robot position of the specified control group.

FORMAT

```
LONG  ESGetPosition( HANDLE handle, LONG ctrlGrp, ESPositionData* positionData );
```

ARGUMENTS

[in] handle	Target handle value
[in] ctrlGrp	control group

Value	Explanation	Value	Explanation
1	R1 (Robot 1) (Pulse)	101	R1 (Robot 1) (Coordinate)
2	R2 (Robot 2) (Pulse)	102	R2 (Robot 2) (Coordinate)
3	R3 (Robot 3) (Pulse)	103	R3 (Robot 3) (Coordinate)
4	R4 (Robot 4) (Pulse)	104	R4 (Robot 4) (Coordinate)
5	R5 (Robot 5) (Pulse)	105	R5 (Robot 5) (Coordinate)
6	R6 (Robot 6) (Pulse)	106	R6 (Robot 6) (Coordinate)
7	R7 (Robot 7) (Pulse)	107	R7 (Robot 7) (Coordinate)
8	R8 (Robot 8) (Pulse)	108	R8 (Robot 8) (Coordinate)
11	B1 (Base 1) (Pulse)		
12	B2 (Base 2) (Pulse)		
13	B3 (Base 3) (Pulse)		
14	B4 (Base 4) (Pulse)		
15	B5 (Base 5) (Pulse)		
16	B6 (Base 6) (Pulse)		
17	B7 (Base 7) (Pulse)		
18	B8 (Base 8) (Pulse)		
21	S1 (Station 1) (Pulse)		
22	S2 (Station 2) (Pulse)		
23	S3 (Station 3) (Pulse)		
24	S4 (Station 4) (Pulse)		
25	S5 (Station 5) (Pulse)		
26	S6 (Station 6) (Pulse)		
27	S7 (Station 7) (Pulse)		
28	S8 (Station 8) (Pulse)		
29	S9 (Station 9) (Pulse)		
30	S10 (Station 10)(Pulse)		
31	S11 (Station 11)(Pulse)		
32	S12 (Station 12)(Pulse)		
33	S13 (Station 13)(Pulse)		
34	S14 (Station 14)(Pulse)		
35	S15 (Station 15)(Pulse)		
36	S16 (Station 16)(Pulse)		
37	S17 (Station 17)(Pulse)		
38	S18 (Station 18)(Pulse)		
39	S19 (Station 19)(Pulse)		
40	S20 (Station 20)(Pulse)		
41	S21 (Station 21)(Pulse)		
42	S22 (Station 22)(Pulse)		
43	S23 (Station 23)(Pulse)		
44	S24 (Station 24)(Pulse)		

[out] positionData Robot position data storage pointer

ESPositionData
Robot position data structure

FORMAT

```
typedef struct
{
   LONG  dataType;
   LONG  fig;
   LONG  toolNo;
   LONG  userFrameNo;
   LONG  exFig;
   ESAxisData axesData;
} ESPositionData;
```

MEMBER

<dataType> data type

Value	Explanation
0	Pulse
16	Coordinate （base）

<fig> Figure

bitValue	Explanation
D00	0:Front side 1:Back side
D01	0:Elbow above 1:Elbow under
D02	0:Flip 1:No-flip
D03	0:R<180deg 1:R>=180deg
D04	0:T<180deg 1:T>=180deg
D05	0:S<180deg 1:S>=180deg
D06-D07	Reserved

<toolNo> tool number

<userFrameNo> user frame number

<exFig> extended figure

<axesData> Axes data
ESAxisData
Axis data structure

FORMAT : #define Number_of_Axis (8)

```
typedef struct
{
   DOUBLE    axis[Number_of_Axis];
 } ESAxisData;
```

MEMBER : <axis[Number_of_Axis]> Axis data of robot (Size = 8)

Array	Pulse type	Coordinate type
axis[0]	1st axis Pulse	X-coordinate (unit: mm)
axis[1]	2nd axis Pulse	Y-coordinate (unit: mm)
axis[2]	3rd axis Pulse	Z-corrdinate (unit: mm)
axis[3]	4th axis Pulse	Rx angle (unit: deg)
axis[4]	5th axis Pulse	Ry angle (unit: deg)
axis[5]	6th axis Pulse	Rz angle (unit: deg)
axis[6]	7th axis Pulse	Re angle (unit: deg)
axis[7]	8th axis Pulse	

RETURN VALUE

0	: Normal completion
0xA001	: No control group
Others	: Error codes

■ ESGetDeviation

Reads a current deviation of the specified control group.

FORMAT

```
LONG  ESGetDeviation( HANDLE handle, LONG ctrlGrp, ESAxisData* deviationData );
```

ARGUMENTS

[in] handle	Target handle value
[in] ctrlGrp	Control group

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

[out] deviationData Axis data storage pointer

ESPositionData
Axis data structure

FORMAT
```
#define Number_of_Axis (8)

typedef struct
{
   DOUBLE    axis[Number_of_Axis];
} ESAxisData;
```

MEMBER <axis[Number_of_Axis]> Axis data of robot (Size = 8)

Array	Explanation
axis[0]	1st axis Pulse
axis[1]	2nd axis Pulse
axis[2]	3rd axis Pulse
axis[3]	4th axis Pulse
axis[4]	5th axis Pulse
axis[5]	6th axis Pulse
axis[6]	7th axis Pulse
axis[7]	8th axis Pulse

RETURN VALUE

0	: Normal completion
0xA001	: No control group
Others	: Error codes

■ ESGetTorque

Reads a current torque of the specified control group.

FORMAT

```
LONG  ESGetTorque( HANDLE handle, LONG ctrlGrp, ESAxisData* torqueData );
```

ARGUMENTS

[in] handle	Target handle value
[in] ctrlGrp	Control group

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

[out] torqueData Axis data storage pointer

ESPositionData
Axis data structure

FORMAT

```
#define Number_of_Axis (8)

typedef struct
{
   DOUBLE    axis[Number_of_Axis];
} ESAxisData;
```

MEMBER <axis[Number_of_Axis]> Axis data of robot (Size = 8)

Array	Explanation
axis[0]	1st axis (unit: 0.01%)
axis[1]	2nd axis (unit: 0.01%)
axis[2]	3rd axis (unit: 0.01%)
axis[3]	4th axis (unit: 0.01%)
axis[4]	5th axis (unit: 0.01%)
axis[5]	6th axis (unit: 0.01%)
axis[6]	7th axis (unit: 0.01%)
axis[7]	8th axis (unit: 0.01%)

RETURN VALUE

0	: Normal completion
0xA001	: No control group
Others	: Error codes

■ ESGetMonitoringTime

Reads a current monitoring time.

FORMAT

LONG ESGetMonitoringTime(HANDLE handle, LONG timeType, ESMonitoringTime-Data* timeData);

ARGUMENTS

[in] handle	Target handle value
[in] timeType	Monitoring time type

Value	Explanation
1	Control power time
10	Servo power time（TOTAL）
11 to 18	Servo power time（R1 to R8）
21 to 44	Servo power time（S1 to S24）
110	Playback time（TOTAL）
111 to 118	Playback time（R1 to R8）
121 to 144	Playback time（S1 to S24）
210	Moving time（TOTAL）
211 to 218	Moving time（R1 to R8）
221 to 244	Moving time（S1 to S24）
301 to 308	Operating time（APPLI1 to APPLI8）

[out] timeData	Monitoring time data storage pointer

ESMonitoringTimeData
Monitoring time data structure

FORMAT

```
#define Length_of_Time (16)
#define Length_of_ElapseTime (12)

typedef struct
{
   CHAR  startTime[Length_of_Time+1];
   CHAR  elapseTime[Length_of_ElapseTime+1];
} ESMonitoringTimeData;
```

MEMBER

<startTime[Length_of_Time+1]> Started time(Max size = 16)

<elapseTime[Length_of_Time+1]> Elapsed time(Max size = 12)

RETURN VALUE

0	: Normal completion
0xA001	: Out of type range
0xC800	: No Monitoring time
Others	: Error codes

■ ESGetSystemInfo

Read a current system information.

FORMAT

LONG ESGetSystemInfo(HANDLE handle, LONG systemType, ESSystemInfoData* info-Data);

ARGUMENTS

[in] handle	Target handle value
[in] systemType	System information type

Value	Explanation
11 to 18	Model information （R1 to R8）
21 to 44	Model information （S1 to S24）
101 to 108	Application information （R1 to R8）

[out] infoData	System information data storage pointer

ESSystemInfoData

System information data structure

FORMAT

```
#define Length_of_SystemVer (24)
#define Length_of_RobotName (16)
#define Length_of_ParamNo (8)

typedef struct
{
   CHAR  systemVersion[Length_of_SystemVer+1];
   CHAR  name[Length_of_RobotName+1];
   CHAR  parameterNo[Length_of_ParamNo+1];
} ESSystemInfoeData;
```

MEMBER

<systemVersion[Length_of_SystemVer+1]> System version (Max size = 24)

<name[Length_of_RobotName+1]> Model/Application name (Max size = 16)

<parameterNo[Length_of_ParamNo+1]> Parameter number (Max size = 8)

RETURN VALUE

0	: Normal completion
0xB006	: No application
0xB007	: No model
Others	: Error codes

■ ESGetVarData1

Reads a variable (B,I,D,R).

FORMAT

```
LONG  ESGetVarData1( HANDLE handle, LONG type, LONG number, DOUBLE* data );
```

ARGUMENTS

[in] handle	Target handle value
[in] type	Variable data type

Value	Explanation
1	Byte (B)
2	Integer (I)
3	Double (D)
4	Real (R)

[in] number	Variable number

Value	Explanation
1 to 100	RS022=0
0 to 99	RS022=1

NOTE Check the "RS022" parameter before using this function. Set the value as above. If the "RS022"parameter is zero, the variable number needs to added 1.

[out] data	Variable data storage pointer

RETURN VALUE

0	: Normal completion
0xA001	: Out of variable number range
Others	: Error codes

REMARKS

Restrictions

Check the "RS023" parameter before using this function. Use "ESGetVarData1" or "ESGetVarData2" as to the "RS023" parameter.

RS023	Function
0	ESGetVarData1
1	ESGetVarData2

REFERENCE

"ESGetVarData2" "ESGetVarDataMB" "ESGetVarDataMI" "ESGetVarDataMD" "ESGetVarDataMR" "ESSetVarData1" "ESSetVarData2" "ESSetVarDataMB" "ESSetVarDataMI" "ESSetVarDataMD" "ESSetVarDataMR"

■ ESGetVarData2

Reads a variable (B,I,D,R).

FORMAT

```
LONG  ESGetVarData2( HANDLE handle, LONG type, LONG number, DOUBLE* data );
```

ARGUMENTS

[in] handle	Target handle value
[in] type	variable data type

Value	Explanation
1	Byte (B)
2	Integer (I)
3	Double (D)
4	Real (R)

NOTE Check the "RS022" parameter before using this function. Set the value as above. If the "RS022"parameter is zero, the variable number needs to added 1.

[in] number	Variable number

Value	Explanation
1 to 100	RS022=0
0 to 99	RS022=1

[out] data	Variable data storage pointer

RETURN VALUE

0	: Normal completion
0xA001	: Out of variable number range
Others	: Error codes

REMARKS

Restrictions
Check the "RS023" parameter before using this function. Use "ESGetVarData1" or "ESGetVarData2" as to the "RS023" parameter.

RS023	Function
0	ESGetVarData1
1	ESGetVarData2

REFERENCE

"ESGetVarData1" "ESGetVarDataMB" "ESGetVarDataMI" "ESGetVarDataMD" "ESGetVarDataMR" "ESSetVarData1" "ESSetVarData2" "ESSetVarDataMB" "ESSetVarDataMI" "ESSetVarDataMD" "ESSetVarDataMR"

■ ESGetStrData

Reads a string variable.

FORMAT

LONG ESGetStrData(HANDLE handle, LONG number, CHAR* cp);

ARGUMENTS

[in] handle	Target handle value
[in] number	Variable number

Value	Explanation
1 to 100	RS022=0
0 to 99	RS022=1

NOTE Check the "RS022" parameter before using this function. Set the value as above. If the "RS022"parameter is zero, the variable number needs to added 1.

[out] cp	String variable data storage pointer (Max size = 16)

RETURN VALUE

0	: Normal completion
0xA001	: Out of variable number range
Others	: Error codes

REFERENCE

"ESGetStrDataM" "ESSetStrData" "ESSetStrDataM"

■ ESGetPositionData

Reads a robot position variable.

FORMAT

LONG ESGetPositonData(HANDLE handle, LONG number, ESPositionData* position-Data);

ARGUMENTS

[in] handle	Target handle value
[in] number	Variable number

Value	Explanation
1 to 128	RS022=0
0 to 127	RS022=1

NOTE Check the "RS022" parameter before using this function. Set the value as above. If the "RS022"parameter is zero, the variable number needs to added 1.

[out] positionData	Robot position data storage pointer

ESPositionData
Robot position data structure

FORMAT
```
typedef struct
{
   LONG  dataType;
   LONG  fig;
   LONG  toolNo;
   LONG  userFrameNo;
   LONG  exFig;
   ESAxisData axesData;
} ESPositionData;
```

MEMBER <dataType> Data type

Value	Explanation
0	Pulse
16	Coordinate（Base-coordinate）
17	Coordinate（Robot-coordinate）
18	Coordinate（Tool-coordinate）
19	Coordinate（User-coordinate）
20	Coordinate（Master tool-coordinate）

<fig>figure

bitValue	Explanation
D00	0:Front side 1:Back side
D01	0:Elbow above 1:Elbow under
D02	0:Flip 1:No-flip
D03	0:R<180deg 1:R>=180deg
D04	0:T<180deg 1:T>=180deg
D05	0:S<180deg 1:S>=180deg
D06-D07	Reserved

<toolNo> Tool number

<userFrameNo> User frame Number

<exFig> extended figure

<axesData> Axes data
ESAxisData
Axis data structure

FORMAT : #define Number_of_Axis (8)

```
typedef struct
{
    DOUBLE    axis[Number_of_Axis];
} ESAxisData;
```

MEMBER : <axis[Number_of_Axis]> Axis data of robot (Size = 8)

Array	Pulse type	Coordinate type
axis[0]	1st axis Pulse	X-coordinate (unit: mm)
axis[1]	2nd axis Pulse	Y-coordinate (unit: mm)
axis[2]	3rd axis Pulse	Z-corrdinate (unit: mm)
axis[3]	4th axis Pulse	Rx angle (unit: deg)
axis[4]	5th axis Pulse	Ry angle (unit: deg)
axis[5]	6th axis Pulse	Rz angle (unit: deg)
axis[6]	7th axis Pulse	Re angle (unit: deg)
axis[7]	8th axis Pulse	

RETURN VALUE

0	: Normal completion
0xA001	: Out of variable number range
0xB001	: No data
Others	: Error codes

REFERENCE

"ESGetPositionDataM" "ESSetPositionData" "ESSetPositionDataM"

■ ESGetBpexPositionData

Reads a base/external-axis position variable.

FORMAT

LONG ESGetBpexPositonData(HANDLE handle, LONG type, LONG number, ESBpexPositionData* positionData);

ARGUMENTS

[in] handle	Target handle value
[in] tpye	Variable data type

Value	Explanation
1	Base
2	External axis

[in] number	Variable number

Value	Explanation
1 to 128	RS022=0
0 to 127	RS022=1

NOTE
Check the "RS022" parameter before using this function. Set the value as above. If the "RS022"parameter is zero, the variable number needs to added 1.

[out] positionData	Base/External-axis position data storage pointer

ESBpexPositionData
Base/External-axis position data structure

FORMAT

```
typedef struct
{
   LONG  dataType;
   ESAxisData  axesData;
} ESBpexPositionData;
```

MEMBER <dataType> Data type

Value	Explanation
0	Pulse
16	Coordinate（Base only）

<axesData> Axes data
ESAxisData
Axis data structure

FORMAT : #define Number_of_Axis (8)

```
typedef struct
{
    DOUBLE    axis[Number_of_Axis];
} ESAxisData;
```

MEMBER : <axis[Number_of_Axis]> Axis data of base/external-axis (Size = 8)

Array	Pulse type	Coordinate type
axis[0]	1st axis Pulse	X-coordinateValue (unit: mm)
axis[1]	2nd axis Pulse	Y-coordinateValue (unit: mm)
axis[2]	3rd axis Pulse	Z-coordinateValue (unit: mm)
axis[3]	4th axis Pulse	
axis[4]	5th axis Pulse	
axis[5]	6th axis Pulse	
axis[6]	7th axis Pulse	
axis[7]	8th axis Pulse	

RETURN VALUE

0	: Normal completion
0xA001	: Out of variable number range
0xB001	: No data
Others	: Error codes

REFERENCE

"ESGetBpexPositionDataM" "ESSetBpexPositionData" "ESSetBpexPositionDataM"

■ ESGetVarDataMB

Reads variables (B) continuously from the specified number.

FORMAT

LONG ESGetVarDataMB(HANDLE handle, LONG varno, LONG number, ESMultiByte-Data* varData);

ARGUMENTS

[in] handle	Target handle value
[in] varno	Variable number

Value	Explanation
1 to 100	RS022=0
0 to 99	RS022=1

NOTE Check the "RS022" parameter before using this function. Set the value as above. If the "RS022"parameter is zero, the variable number needs to added 1.

[in] number	Number of variables

NOTE It needs that Number of variables is even.

[out] varData	Multi-data storage pointer

ESMultiByteData
Multi-data structure (1byte)

FORMAT
```
#define Length_of_Multi_1 (474)

typedef struct
{
   CHAR    data[Length_of_Multi_1];
} ESMultiByteData;
```

MEMBER <data[Length_of_Multi_1]> Data (1byte) (Max number = 474)

RETURN VALUE

0 : Normal completion
0xA001 : Out of variable number range
0xB004 : Including out of variable number range
Others : Error codes

REFERENCE

"ESGetVarData1" "ESGetVarData2" "ESGetVarDataMI" "ESGetVarDataMD"

"ESGetVarDataMR" "ESSetVarData1" "ESSetVarData2" "ESSetVarDataMB" "ESSetVarDataMI" "ESSetVarDataMD" "ESSetVarDataMR"

■ ESGetVarDataMI

Reads variables (I) continuously from the specified number.

FORMAT

LONG ESGetVarDataMI(HANDLE handle, LONG varno, LONG number, ESMultiShort-Data* varData);

ARGUMENTS

[in] handle	Target handle value
[in] varno	Variable number

Value	Explanation
1 to 100	RS022=0
0 to 99	RS022=1

NOTE

Check the "RS022" parameter before using this function. Set the value as above. If the "RS022"parameter is zero, the variable number needs to added 1.

[in] number	Number of variables
[out] varData	Multi-data storage pointer

ESMultiShortData
Multi-data structure (2byte)

FORMAT
```
#define Length_of_Multi_2 (237)

typedef struct
{
   SHORT    data[Length_of_Multi_2];
} ESMultiShortData;
```

MEMBER <data[Length_of_Multi_2]> Data (2byte) (Max number = 237)

RETURN VALUE

0 : Normal completion
0xA001 : Out of variable number range
0xB004 : Including out of variable number range
Others : Error codes

REFERENCE

"ESGetVarData1" "ESGetVarData2" "ESGetVarDataMB" "ESGetVarDataMD" "ESGetVarDataMR" "ESSetVarData1" "ESSetVarData2" "ESSetVarDataMB" "ESSetVarDataMI" "ESSetVarDataMD" "ESSetVarDataMR"

■ ESGetVarDataMD

Reads variables (D) continuously from the specified number.

FORMAT

LONG ESGetVarDataMD(HANDLE handle, LONG varno, LONG number, ESMultiLongData* varData);

ARGUMENTS

[in] handle	Target handle value
[in] varno	Variable number

Value	Explanation
1 to 100	RS022=0
0 to 99	RS022=1

NOTE Check the "RS022" parameter before using this function. Set the value as above. If the "RS022"parameter is zero, the variable number needs to added 1.

[in] number	Number of variables
[out] varData	Multi-data storage pointer

ESMultiLongData
Multi-data structure (4byte)

FORMAT

```
#define Length_of_Multi_4 (118)

typedef struct
{
   LONG    data[Length_of_Multi_4];
} ESMultiLongData;
```

MEMBER <data[Length_of_Multi_4]> (Max number = 118)

RETURN VALUE

0	: Normal completion
0xA001	: Out of variable number range
0xB004	: Including out of variable number range
Others	: Error codes

REFERENCE

"ESGetVarData1" "ESGetVarData2" "ESGetVarDataMB" "ESGetVarDataMI" "ESGetVarDataMR" "ESSetVarData1" "ESSetVarData2" "ESSetVarDataMB" "ESSetVarDataMI" "ESSetVarDataMD" "ESSetVarDataMR"

■ ESGetVarDataMR

Reads variables (R) continuously from the specified number.

FORMAT

LONG ESGetVarDataMR(HANDLE handle, LONG varno, LONG number, ESMultiReal-Data* varData);

ARGUMENTS

[in] handle	Target handle value
[in] varno	Variable number

Value	Explanation
1 to 100	RS022=0
0 to 99	RS022=1

NOTE Check the "RS022" parameter before using this function. Set the value as above. If the "RS022"parameter is zero, the variable number needs to added 1.

[in] number	Number of variables
[out] varData	Multi-data storage pointer

ESMultiRealData
Multi-data structure (Real number)

FORMAT
```
#define Length_of_Multi_4 (118)

typedef struct
{
   DOUBLE    data[Length_of_Multi_4];
} ESMultiRealData;
```

MEMBER <data[Length_of_Multi_4]> Data (Real number) (Max number = 118)

RETURN VALUE

0 : Normal completion
0xA001 : Out of variable number range
0xB004 : Including out of variable number range
Others : Error codes

REFERENCE

"ESGetVarData1" "ESGetVarData2" "ESGetVarDataMB" "ESGetVarDataMI" "ESGetVarDataMD" "ESSetVarData1" "ESSetVarData2" "ESSetVarDataMB" "ESSetVarDataMI" "ESSetVarDataMD" "ESSetVarDataMR"

■ ESGetStrDataM

Reads string variables continuously from the specified number.

FORMAT

```
LONG  ESGetStrDataM(HANDLE handle, LONG varno, LONG number, ESMultiStrData* varData );
```

ARGUMENTS

[in] handle	Target handle value
[in] varno	Variable number

Value	Explanation
1 to 100	RS022=0
0 to 99	RS022=1

NOTE Check the "RS022" parameter before using this function. Set the value as above. If the "RS022"parameter is zero, the variable number needs to added 1.

[in] number	Number of variables
[out] varData	Multi-data storage pointer

ESMultiStrData
Multi-data structure (String)

FORMAT

```
#define Length_of_Multi_Str (29)
#define Length_of_String (16)

typedef struct
{
    CHAR    data[Length_of_Multi_Str][Length_of_String+1];
} ESMultiStrData;
```

MEMBER <data[Length_of_Multi_Str][Length_of_String+1]>
String data (Max number =29/Max length = 16)

RETURN VALUE

0	: Normal completion
0xA001	: Out of variable number range
0xB004	: Including out of variable number range
Others	: Error codes

REFERENCE

"ESGetStrData" "ESSetStrData" "ESSetStrDataM"

■ ESGetPositionDataM

Reads robot position variables continuously from the specified number.

FORMAT

LONG ESGetPositonDataM(HANDLE handle, LONG varno, LONG number, ESMultiPositionData* positionData);

ARGUMENTS

[in] handle	Target handle value
[in] varno	Variable number

Value	Explanation
1-128	RS022=0
0-127	RS022=1

NOTE Check the "RS022" parameter before using this function. Set the value as above. If the "RS022"parameter is zero, the variable number needs to added 1.

[in] number	Number of variables
[out] positionData	Multi-data storage pointer

ESMultiPositionData
Multi-data structure (Robot position)

FORMAT
```
#define Length_of_Multi_Pos (9)

typedef struct
{
    ESPositionData    data[Length_of_Multi_Pos];
} ESMultiPositionData;
```

MEMBER <data[Length_of_Multi_Pos]> Robot position data (Max number = 9)
ESPositionData
Robot position structure

```
FORMAT : typedef struct
         {
            LONG  dataType;
            LONG  fig;
            LONG  toolNo;
            LONG  userFrameNo;
            LONG  exFig;
            ESAxisData axesData;
          } ESPositionData;
```

MEMBER : <dataType> Data type

Value	Explanation
0	Pulse
16	Coordinate（Base-coordinate）
17	Coordinate（Robot-coordinate）
18	Coordinate（Tool-coordinate）
19	Coordinate（User-coordinate）
20	Coordinate（Master tool-coordinate）

<fig> figure

bitValue	Explanation
D00	0:Front side　1:Back side
D01	0:Elbow above　1:Elbow under
D02	0:Flip　1:No-flip
D03	0:R<180deg　1:R>=180deg
D04	0:T<180deg　1:T>=180deg
D05	0:S<180deg　1:S>=180deg
D06-D07	Reserved

<toolNo> Tool number

<userFrameNo> User frame number

<exFig> extended figure

<axesData> Axes data
ESAxisData
Axis data structure

```
FORMAT : #define Number_of_Axis (8)

         typedef struct
         {
            DOUBLE    axis[Number_of_Axis];
          } ESAxisData;
```

MEMBER : <axis[Number_of_Axis]> Axis data of robot
(Size = 8)

Array	Pulse type	Coordinate type
axis[0]	1st axis Pulse	X-coordinate (unit: mm)
axis[1]	2nd axis Pulse	Y-coordinate (unit: mm)
axis[2]	3rd axis Pulse	Z-corrdinate (unit: mm)
axis[3]	4th axis Pulse	Rx angle (unit: deg)
axis[4]	5th axis Pulse	Ry angle (unit: deg)
axis[5]	6th axis Pulse	Rz angle (unit: deg)
axis[6]	7th axis Pulse	Re angle (unit: deg)
axis[7]	8th axis Pulse	

RETURN VALUE

0	: Normal completion
0xA001	: Out of variable number range
0xB001	: Including no data
0xB004	: Including out of variable number range
Others	: Error codes

REFERENCE

"ESGetPositionData" "ESSetPositionData" "ESSetPositionDataM"

■ ESGetBpexPositionDataM

Reads base/external-axis position variables continuously from the specified number.

FORMAT

LONG ESGetBpexPositonDataM(HANDLE handle, LONG type, LONG varno, LONG number,
ESMultiBpexPositionData* positionData);

ARGUMENTS

[in] handle	Target handle value
[in] type	Variable data type

Value	Explanation
1	Base
2	External axis

[in] varno	Variable number

Value	Explanation
1 to 128	RS022=0
0 to 127	RS022=1

NOTE Check the "RS022" parameter before using this function. Set the value as above. If the "RS022"parameter is zero, the variable number needs to added 1.

[in] number	Number of variables
[out] positionData	Multi-data storage pointer

ESMultiBpexPositionData
Multi-data structure (Base/External axis)

FORMAT
```
#define Length_of_Multi_Bpex (13)

typedef struct
{
   ESBpexPositionData    data[Length_of_Multi_Bpex];
} ESMultiBpexPositionData;
```

MEMBER <data[Length_of_Multi_Bpex]> Base/External-axis position data (max number = 13)

ESBpexPositionData
Base/External-axis position data structure

```
FORMAT : typedef struct
         {
            LONG  dataType;
            ESAxisData axesData;
         } ESBpexPositionData;
```

MEMBER : <dataType> data type

Value	Explanation
0	Pulse
16	Coordinate （Base only）

<axesData> Axes data
ESAxisData
axis data structure

```
FORMAT : #define Number_of_Axis (8)

         typedef struct
         {
            DOUBLE    axis[Number_of_Axis];
         } ESAxisData;
```

MEMBER : <axis[Number_of_Axis]> Axis data of base/ external-axis (Size = 8)

Array	Pulse type	Coordinate type
axis[0]	1st axis Pulse	X-coordinateValue (unit: mm)
axis[1]	2nd axis Pulse	Y-coordinateValue (unit: mm)
axis[2]	3rd axis Pulse	Z-coordinateValue (unit: mm)
axis[3]	4th axis Pulse	
axis[4]	5th axis Pulse	
axis[5]	6th axis Pulse	
axis[6]	7th axis Pulse	
axis[7]	8th axis Pulse	

RETURN VALUE

0	: Normal completion
0xA001	: Out of variable number range
0xB001	: Including no data
0xB004	: Including out of variable number range
Others	: Error codes

REFERENCE

"ESGetBpexPositionData" "ESSetBpexPositionData" "ESSetBpexPositionDataM"

■ ESGetAlarmEx

Reads a current error data. (for applying the sub code character strings)

FORMAT

LONG ESGetAlarmEx(HANDLE handle, ESAlarmListEx* alarmList);

ARGUMENTS

[in] handle	Target handle value
[out] alarmList	Alarm list storage pointer (for applying the sub code character strings)

ESAlarmListEx
  Alarm list structure (for applying the sub code character strings)

FORMAT
```
#define Length_of_AlarmList (4)

typedef struct
{
   ESAlarmDataEx    data[Length_of_AlarmList];
} ESAlarmListEx;
```

MEMBER
```
< data[Length_of_AlarmList]>  Alarm data (for applying the sub code
                                         character strings,
                                         Max number = 4)
  ESAlarmDataEx
    Alarm data structure (for applying the sub code character strings)

    FORMAT : typedef struct
             {
                ESAlarmData  alarmData;
                ESSubcodeData  subcodeData;
             } ESAlarmDataEx;

    MEMBER : <alarmData>  Alarm data
             ESAlarmData
               Alarm data structure

               FORMAT : #define Length_of_Time (16)
                        #define Length_of_Name (32)
                         typedef struct
                         {
                            LONG  alarmCode;
                            LONG  alarmData;
                            LONG  alarmType;
                            CHAR  alarmTime[Length_of_Time+1];
                            CHAR  alarmName[Length_of_Name+1];
                         } ESAlarmData;
```

MEMBER : <alarmCode> Alarm code

<alarmData> Alarm data

<alarmType> Alarm data type

<alarmTime[Length_of_Time+1]>
Time of alarm occurrence (Max size = 16)

<alarmName[Length_of_Name+1]>
Alarm name (Max size = 32)

<subcodeData> Sub code data
ESSubcodeData
Alarm sub code character strings data structure

```
FORMAT : #define Length_of_Subcode_AddInfo (16)
         #define Length_of_Subcode_StrData (96)
         typedef struct
         {
            CHAR alarmAddInfo
                    [Length_of_Subcode_AddInfo+1];
            CHAR  alarmStrData
                    [Length_of_Subcode_StringData+1];
            CHAR  alarmHighlightData
                    [Length_of_Subcode_StringData+1];
         } ESSubcodeData;
```

MEMBER :

<alarmAddInfo[Length_of_Subcode_AddInfo+1]>
Sub code data additional information character strings (Max size = 16)

<alarmStrData
[Length_of_Subcode_StringData+1]>
Sub code data character strings
(Max size = 96)

<alarmHighLightData
[Length_of_Subcode_StringData+1]>
Sub code data character strings reverse display information (Max size = 96)

RETURN VALUE

0 : Normal completion
Others : Error codes

REFERENCE

"ESGetAlarm" "ESGetAlarmHist" "ESGetStatus" "ESGetAlarmHistEx"

■ ESGetAlarmHistEx

Reads an alarm history of specified alarm number. (for applying the sub code character strings)

FORMAT

LONG ESGetAlarmHistEx(HANDLE handle, LONG alarmHistNo, ESAlarmDataEx* alarm-Data);

ARGUMENTS

[in] handle	Target handle value
[in] alarmHistNo	Alarm number

Value	Explanation
1～100	Major alarm
1001～1100	Minor alarm
2001～2100	User alarm （system）
3001～3100	User alarm （user）
4001～4100	off-line alarm

[out] alarmData	Alarm data storage pointer (for applying the sub code character strings)

ESAlarmDataEx
Alarm data structure (for applying the sub code character strings)

FORMAT
```
typedef struct
{
   ESAlarmData  alarmData;
   ESSubcodeData  subcodeData;
} ESAlarmDataEx;
```

MEMBER
<alarmData> Alarm data
ESAlarmData
Alarm data structure

```
FORMAT : #define Length_of_Time (16)
         #define Length_of_Name (32)

         typedef struct
         {
            LONG  alarmCode;
            LONG  alarmData;
            LONG  alarmType;
            CHAR  alarmTime[Length_of_Time+1];
            CHAR  alarmName[Length_of_Name+1];
         } ESAlarmData;
```

```
MEMBER : <alarmCode>  Alarm code

         <alarmData>  Alarm data

         <alarmType>  Alarm data type

         <alarmTime[Length_of_Time+1]>
                Time of alarm occurrence (Max size = 16)

         <alarmName[Length_of_Name+1]>
                Alarm Name (Max size = 32)
```

<subcodeData> Sub code data
ESSubcodeData
Alarm sub code character strings data structure

```
FORMAT : #define Length_of_Subcode_AddInfo (16)
         #define Length_of_Subcode_StringData (96)

         typedef struct
         {
            CHAR  alarmAddInfo[Length_of_Subcode_AddInfo+1];
            CHAR  alarmStrData[Length_of_Subcode_StringData+1];
            CHAR  alarmHighlightData
                      [Length_of_Subcode_StringData+1];
         } ESSubcodeData;

MEMBER : <alarmAddInfo[Length_of_Subcode_AddInfo+1]>
            Sub code data additional information character strings
            (Max size = 16)

         <alarmStrData[Length_of_Subcode_StringData+1]>
            Sub code data character strings (Max size = 96)

         <alarmHighLightData[Length_of_Subcode_StringData+1]>
            Sub code data character strings reverse display
            information (Max size = 96)
```

RETURN VALUE

0 : Normal completion
0xA001: Out of alarm number range
Others : Error codes

REFERENCE

"ESGetAlarm" "ESGetAlarmHist" "ESGetStatus" "ESGetAlarmEx"

■ ESReset

Resets alarm.

FORMAT

LONG ESReset(HANDLE handle);

ARGUMENTS

[in] handle	Target handle value

RETURN VALUE

0	: Normal completion
Others	: Error codes

REFERENCE

"ESCancel"

■ ESCancel

Cancels error.

FORMAT

```
LONG  ESCancel( HANDLE handle );
```

ARGUMENTS

[in] handle	Target handle value

RETURN VALUE

0	: Normal completion
Others	: Error codes

REFERENCE

"ESReset"

■ ESHold

Sets hold on/off.

FORMAT

```
LONG  ESHold( HANDLE handle, LONG onOff );
```

ARGUMENTS

[in] handle	Target handle value
[in] onOff	Hold status

Value	Explanation
1	Hold ON
2	Hold OFF

RETURN VALUE

0 : Normal completion
Others : Error codes

REFERENCE

"ESServo" "ESHlock"

■ ESServo

Sets servo on/off.

FORMAT

```
LONG  ESServo( HANDLE handle, LONG onOff );
```

ARGUMENTS

[in] handle	Target handle value
[in] onOff	Servo status

Value	Explanation
1	Servo ON
2	Servo OFF

RETURN VALUE

0	: Normal completion
Others	: Error codes

REFERENCE

"ESHold" "ESHlock"

■ ESHlock

Sets interlock on/off

FORMAT

LONG ESHlock(HANDLE handle, LONG onOff);

ARGUMENTS

[in] handle	Target handle value
[in] onOff	Interlock status

Value	Explanation
1	Hlock ON
2	Hlock OFF

RETURN VALUE

0 : Normal completion
Others : Error codes

REFERENCE

"ESHold" "ESServo"

■ ESCycle

Sets cycle mode.

FORMAT

```
LONG  ESCycle( HANDLE handle, LONG cycle );
```

ARGUMENTS

[in] handle	Target handle value
[in] cycle	Cycle mode

Value	Explanation
1	Step
2	Cycle
3	Auto oparation

RETURN VALUE

0	: Normal completion
Others	: Error codes

■ ESBDSP

Displays string on the programming pendant.

FORMAT

LONG ESBDSP(HANDLE handle, CHAR* message);

ARGUMENTS

[in] handle	Target handle value
[in] message	String storage pointer(Max length = 30)

RETURN VALUE

0	: Normal completion
Others	: Error codes

■ ESSetVarData1

Sets a variable (B,I,D,R).

FORMAT

LONG ESSetVarData1(HANDLE handle, LONG type, LONG number, DOUBLE data);

ARGUMENTS

[in] handle Target handle value

[in] type variable data type

Value	Explanation
1	Byte (B)
2	Integer (I)
3	Double (D)
4	Real (R)

[in] number Variable number

Value	Explanation
1 to 100	RS022=0
0 to 99	RS022=1

NOTE Check the "RS022" parameter before using this function. Set the value as above. If the "RS022"parameter is zero, the variable number needs to added 1.

[in] data Variable number

Data type	Range
Byte (B)	0 to 255
Integer (I)	-32768 to 32767
Double (D)	-2147483648 to 2147483647
Real (R)	-3.4E+38 to 3.4E+38

RETURN VALUE

0 : Normal completion
0xA001 : Out of variable number range
Others : Error codes

REMARKS

Restrictions
Check the "RS023" parameter before using this function. Use "ESSetVarData1" or "ESSetVarData2" as to the "RS023" parameter.

RS023	Function
0	ESGetVarData1
1	ESGetVarData2

REFERENCE

"ESGetVarData1" "ESGetVarData2" "ESGetVarDataMB" "ESGetVarDataMI" "ESGetVarDataMD" "ESGetVarDataMR" "ESSetVarData2" "ESSetVarDataMB" "ESSetVarDataMI" "ESSetVarDataMD" "ESSetVarDataMR"

■ ESSetVarData2

Sets a variable (B,I,D,R).

FORMAT

LONG ESSetVarData2(HANDLE handle, LONG type, LONG number, DOUBLE data);

ARGUMENTS

[in] handle	Target handle value
[in] type	variable data type

Value	Explanation
1	Byte (B)
2	Integer (I)
3	Double (D)
4	Real (R)

[in] number	Variable number

Value	Explanation
1 to 100	RS022=0
0 to 99	RS022=1

NOTE Check the "RS022" parameter before using this function. Set the value as above. If the "RS022"parameter is zero, the variable number needs to added 1.

[in] data	Variable data

Data type	Range
Byte (B)	0 to 255
Integer (I)	-32768 to 32767
Double (D)	-2147483648 to 2147483647
Real (R)	-3.4E+38 to 3.4E+38

RETURN VALUE

0 : Normal completion
0xA001 : Out of variable number range
Others : Error codes

REMARKS

Restrictions
Check the "RS023" parameter before using this function. Use "ESSetVarData1" or "ESSetVarData2" as to the "RS023" parameter.

RS023	Function
0	ESGetVarData1
1	ESGetVarData2

REFERENCE

"ESGetVarData1" "ESGetVarData2" "ESGetVarDataMB" "ESGetVarDataMI" "ESGetVarDataMD" "ESGetVarDataMR" "ESSetVarData1" "ESSetVarDataMB" "ESSetVarDataMI" "ESSetVarDataMD" "ESSetVarDataMR"

■ ESSetStrData

Sets a string variable.

FORMAT

```
LONG  ESSetStrData( HANDLE handle, LONG number, CHAR* cp );
```

ARGUMENTS

[in] handle	Target handle value
[in] number	Variable number

Value	Explanation
1 to 100	RS022=0
0 to 99	RS022=1

NOTE Check the "RS022" parameter before using this function. Set the value as above. If the "RS022"parameter is zero, the variable number needs to added 1.

[in] cp	String variable data storage pointer (Max size = 16)

RETURN VALUE

0	: Normal completion
0xA001	: Out of variable number range
Others	: Error codes

REFERENCE

"ESGetStrData" "ESGetStrDataM" "ESSetStrDataM"

■ ESSetPositionData

Sets a robot position variable.

FORMAT

LONG ESSetPositionData(HANDLE handle, LONG number, ESPositionData position-Data);

ARGUMENTS

[in] handle	Target handle value
[in] number	Variable number

Value	Explanation
1 to 128	RS022=0
0 to 127	RS022=1

NOTE Check the "RS022" parameter before using this function. Set the value as above. If the "RS022"parameter is zero, the variable number needs to added 1.

[in] positionData	Robot position data structure

ESPositionData
Robot position data structure

FORMAT
```
typedef struct
{
   LONG  dataType;
   LONG  fig;
   LONG  toolNo;
   LONG  userFrameNo;
   LONG  exFig;
   ESAxisData axesData;
} ESPositionData;
```

MEMBER <dataType> Data type

Value	Explanation
0	Pulse
16	Coordinate（Base-coordinate）
17	Coordinate（Robot-coordinate）
18	Coordinate（Tool-coordinate）
19	Coordinate（User-coordinate）
20	Coordinate（Master tool-coordinate）

MEMBER <fig>Figure

bitValue	Explanation	
D00	0:Front side	1:Back side
D01	0:Elbow above	1:Elbow under
D02	0:Flip	1:No-flip
D03	0:R<180deg	1:R>=180deg
D04	0:T<180deg	1:T>=180deg
D05	0:S<180deg	1:S>=180deg
D06-D07	Reserved	

<toolNo> Tool Number

<userFrameNo> User frame number

<exFig> Extended figure

<axesData> Axes data
ESAxisData
Axis data structure

FORMAT : #define Number_of_Axis (8)

```
typedef struct
{
    DOUBLE    axis[Number_of_Axis];
} ESAxisData;
```

MEMBER : <axis[Number_of_Axis]> Axis data of robot (Size = 8)

Array	Pulse type	Coordinate type
axis[0]	1st axis Pulse	X-coordinate (unit: mm)
axis[1]	2nd axis Pulse	Y-coordinate (unit: mm)
axis[2]	3rd axis Pulse	Z-corrdinate (unit: mm)
axis[3]	4th axis Pulse	Rx angle (unit: deg)
axis[4]	5th axis Pulse	Ry angle (unit: deg)
axis[5]	6th axis Pulse	Rz angle (unit: deg)
axis[6]	7th axis Pulse	Re angle (unit: deg)
axis[7]	8th axis Pulse	

RETURN VALUE

0	: Normal completion
0xA001	: Out of variable number range
Others	: Error codes

REFERENCE

"ESGetPositionData" "ESGetPositionDataM" "ESSetPositionDataM"

■ ESSetBpexPositionData

Sets a base/external-axis position variable.

FORMAT

LONG ESSetBpexPositionData(HANDLE handle, LONG type, LONG number, ESBpexPositionData positionData);

ARGUMENTS

[in] handle	Target handle value
[in] type	Variable data type

Value	Explanation
1	Base
2	External axis

[in] number	Variable number

Value	Explanation
1 to 128	RS022=0
0 to 127	RS022=1

NOTE Check the "RS022" parameter before using this function. Set the value as above. If the "RS022"parameter is zero, the variable number needs to added 1.

[in] positionData	Base/External-axis position data data structure

ESBpexPositionData
Base/External-axis position data structure

FORMAT
```
typedef struct
{
   LONG  dataType;
   ESAxisData  axesData;
} ESBpexPositionData;
```

MEMBER <dataType> Data type

Value	Explanation
0	Pulse
16	Coordinate（base）

<axesData> Axes data
ESAxisData
Axis data structure

FORMAT : #define Number_of_Axis (8)

```
typedef struct
{
   DOUBLE    axis[Number_of_Axis];
} ESAxisData;
```

MEMBER : <axis[Number_of_Axis]> Axis data of base/external-axis (Size = 8)

Array	Pulse type	Coordinate type
axis[0]	1st axis Pulse	X-coordinateValue (unit: mm)
axis[1]	2nd axis Pulse	Y-coordinateValue (unit: mm)
axis[2]	3rd axis Pulse	Z-coordinateValue (unit: mm)
axis[3]	4th axis Pulse	
axis[4]	5th axis Pulse	
axis[5]	6th axis Pulse	
axis[6]	7th axis Pulse	
axis[7]	8th axis Pulse	

RETURN VALUE

0	: Normal completion
0xA001	: Out of variable number range
Others	: Error codes

REFERENCE

"ESGetBpexPositionData" "ESGetBpexPositionDataM" "ESSetBpexPositionDataM"

■ ESSetVarDataMB

Sets variables (B) continuously from the specified number.

FORMAT

LONG ESSetVarDataMB(HANDLE handle, LONG varno, LONG number, ESMultiByte-Data varData);

ARGUMENTS

[in] handle	Target handle value
[in] varno	Variable number

Value	Explanation
1 to 100	RS022=0
0 to 99	RS022=1

NOTE Check the "RS022" parameter before using this function. Set the value as above. If the "RS022"parameter is zero, the variable number needs to added 1.

[in] number	Number of variables

NOTE It needs that Number of variables is even.

[in] varData	Multi-data structure

ESMultiByteData
Multi-data structure (1byte)

FORMAT
```
t#define Length_of_Multi_1 (474)

typedef struct
{
   CHAR   data[Length_of_Multi_1];
} ESMultiByteData;
```

MEMBER <data[Length_of_Multi_1]> Data (1byte) (Max number = 474)

RETURN VALUE

0 : Normal completion
0xA001 : Out of variable number range
0xB004 : Including out of variable number range
Others : Error codes

REFERENCE

"ESGetVarData1" "ESGetVarData2" "ESGetVarDataMB" "ESGetVarDataMI"

"ESGetVarDataMD" "ESGetVarDataMR" "ESSetVarData1" "ESSetVarData2" "ESSetVarDataMI" "ESSetVarDataMD" "ESSetVarDataMR"

■ ESSetVarDataMI

Sets variables (I) continuously from the specified number.

FORMAT

LONG ESSetVarDataMI(HANDLE handle, LONG varno, LONG number, ESMultiShortData varData);

ARGUMENTS

[in] handle	Target handle value
[in] varno	Variable number

Value	Explanation
1 to 100	RS022=0
0 to 99	RS022=1

NOTE Check the "RS022" parameter before using this function. Set the value as above. If the "RS022"parameter is zero, the variable number needs to added 1.

[in] number	Number of variables
[in] varData	Multi-data structure

ESMultiShortData

Multi-data structure (2byte)

FORMAT
```
#define Length_of_Multi_2 (237)

typedef struct
{
   SHORT    data[Length_of_Multi_2];
} ESMultiShortData;
```

MEMBER <data[Length_of_Multi_2]> Data (2byte) (Max number = 237)

RETURN VALUE

0 : Normal completion
0xA001 : Out of variable number range
0xB004 : Including out of variable number range
Others : Error codes

REFERENCE

"ESGetVarData1" "ESGetVarData2" "ESGetVarDataMB" "ESGetVarDataMI" "ESGetVarDataMD" "ESGetVarDataMR" "ESSetVarData1" "ESSetVarData2" "ESSetVarDataMB" "ESSetVarDataMD" "ESSetVarDataMR"

■ ESSetVarDataMD

Sets variables (D) continuously from the specified number.

FORMAT

LONG ESSetVarDataMD(HANDLE handle, LONG varno, LONG number, ESMultiLong-Data varData);

ARGUMENTS

[in] handle	Target handle value
[in] varno	Variable number

Value	Explanation
1 to 100	RS022=0
0 to 99	RS022=1

NOTE Check the "RS022" parameter before using this function. Set the value as above. If the "RS022"parameter is zero, the variable number needs to added 1.

[in] number	Number of variables
[in] varData	Multi-data structure

ESMultiLongData
Multi-data structure (4byte)

FORMAT
```
#define Length_of_Multi_4 (118)

typedef struct
{
    LONG    data[Length_of_Multi_4];
} ESMultiLongData;
```

MEMBER <data[Length_of_Multi_4]> Data (4byte) (Max number = 118)

RETURN VALUE

0	: Normal completion
0xA001	: Out of variable number range
0xB004	: Including out of variable number range
Others	: Error codes

REFERENCE

"ESGetVarData1" "ESGetVarData2" "ESGetVarDataMB" "ESGetVarDataMI" "ESGetVarDataMD" "ESGetVarDataMR" "ESSetVarData1" "ESSetVarData2" "ESSetVarDataMB" "ESSetVarDataMI" "ESSetVarDataMR"

■ ESSetVarDataMR

Sets variables (R) continuously from the specified number.

FORMAT

LONG ESSetVarDataMR(HANDLE handle, LONG varno, LONG number, ESMultiRealData varData);

ARGUMENTS

[in] handle	Target handle value
[in] varno	Variable number

Value	Explanation
1 to 100	RS022=0
0 to 99	RS022=1

NOTE Check the "RS022" parameter before using this function. Set the value as above. If the "RS022"parameter is zero, the variable number needs to added 1.

[in] number	Number of variables
[in] varData	Multi-data structure

ESMultiRealData
Multi-data structure (Real number)

FORMAT
```
#define Length_of_Multi_4 (118)

typedef struct
{
   DOUBLE    data[Length_of_Multi_4];
} ESMultiRealData;
```

MEMBER <data[Length_of_Multi_4]> Data (Real number) (Max number = 118)

RETURN VALUE

0 : Normal completion
0xA001 : zut of variable number range
0xB004 : Including out of variable number range
Others : Error codes

REFERENCE

"ESGetVarData1" "ESGetVarData2" "ESGetVarDataMB" "ESGetVarDataMI" "ESGetVarDataMD" "ESGetVarDataMR" "ESSetVarData1" "ESSetVarData2" "ESSetVarDataMB" "ESSetVarDataMI" "ESSetVarDataMD"

■ ESSetStrDataM

Sets string variables continuously from the specified number.

FORMAT

LONG ESSetStrDataM(HANDLE handle, LONG varno, LONG number, ESMultiStrData varData);

ARGUMENTS

[in] handle	Target handle value
[in] varno	Variable number

Value	Explanation
1 to 100	RS022=0
0 to 99	RS022=1

NOTE Check the "RS022" parameter before using this function. Set the value as above. If the "RS022"parameter is zero, the variable number needs to added 1.

[in] number	Number of variables
[in] varData	Multi-data structure

ESMultiStrData
Multi-data structure (String)

FORMAT
```
#define Length_of_Multi_Str (29)
#define Length_of_String (16)

typedef struct
{
   CHAR    data[Length_of_Multi_Str][Length_of_String+1];
} ESMultiStrData;
```

MEMBER <data[Length_of_Multi_Str][Length_of_String+1]> String data (Max number =29/ Max length = 16)

RETURN VALUE

0 : Normal completion
0xA001 : Out of variable number range
0xB004 : Including out of variable number range
Others : Error codes

REFERENCE

"ESGetStrData" "ESSetStrData" "ESSetStrDataM"

■ ESSetPositionDataM

Sets robot position variables continuously from the specified number.

FORMAT

```
LONG  ESSetPositionDataM( HANDLE handle, LONG varno, LONG number,
ESMultiPositionData positionData );
```

ARGUMENTS

[in] handle	Target handle value
[in] varno	Variable number

Value	Explanation
1 to 128	RS022=0
0 to 127	RS022=1

NOTE
Check the "RS022" parameter before using this function. Set the value as above. If the "RS022"parameter is zero, the variable number needs to added 1.

[in] number	Number of variables
[in] positionData	Multi-data structure

ESMultiPositionData
Multi-data structure (Robot position)

FORMAT

```
#define Length_of_Multi_Pos (9)

typedef struct
{
   ESPositionData    data[Length_of_Multi_Pos];
} ESMultiPositionData;
```

MEMBER <data[Length_of_Multi_Pos]> Robot position data (Max number = 9)
ESPositionData
Robot position data structure

```
FORMAT : typedef struct
             {
                LONG  dataType;
                LONG  fig;
                LONG  toolNo;
                LONG  userFrameNo;
                LONG  exFig;
                ESAxisData axesData;
             } ESPositionData;
```

MEMBER : <dataType> Data type

Value	Explanation
0	Pulse
16	Coordinate（Base-coordinate）
17	Coordinate（Robot-coordinate）
18	Coordinate（Tool-coordinate）
19	Coordinate（User-coordinate）
20	Coordinate（Master tool-coordinate）

<fig>Figure

bitValue	Explanation
D00	0:Front side 1:Back side
D01	0:Elbow above 1:Elbow under
D02	0:Flip 1:No-flip
D03	0:R<180deg 1:R>=180deg
D04	0:T<180deg 1:T>=180deg
D05	0:S<180deg 1:S>=180deg
D06-D07	Reserved

<toolNo> Tool number

<userFrameNo>   User frame number

<exFig> Extended figure

<axesData> Axes data
ESAxisData
Axis data structure

FORMAT : #define Number_of_Axis (8)

```
typedef struct
{
   DOUBLE    axis[Number_of_Axis];
} ESAxisData;
```

MEMBER : <axis[Number_of_Axis]> Axis data of robot
(Size = 8)

Array	Pulse type	Coordinate type
axis[0]	1st axis Pulse	X-coordinate (unit: mm)
axis[1]	2nd axis Pulse	Y-coordinate (unit: mm)
axis[2]	3rd axis Pulse	Z-corrdinate (unit: mm)
axis[3]	4th axis Pulse	Rx angle (unit: deg)
axis[4]	5th axis Pulse	Ry angle (unit: deg)
axis[5]	6th axis Pulse	Rz angle (unit: deg)
axis[6]	7th axis Pulse	Re angle (unit: deg)
axis[7]	8th axis Pulse	

RETURN VALUE

0	: Normal completion
0xA001	: Out of variable number range
0xB004	: Including out of variable number range
Others	: Error codes

REFERENCE

"ESGetPositionData" "ESSetPositionData" "ESGetPositionDataM"

■ ESSetBpexPositionDataM

Sets base/external-axis position variables continuously from the specified number.

FORMAT

LONG ESSetBpexPositionDataM(HANDLE handle, LONG type, LONG varno, LONG number,
ESMultiBpexPositionData positionData);

ARGUMENTS

[in] handle	Target handle value
[in] type	variable data type

Value	Explanation
1	Base
2	External axis

[in] varno	Variable number

Value	Explanation
1 to 128	RS022=0
0 to 127	RS022=1

NOTE Check the "RS022" parameter before using this function. Set the value as above. If the "RS022"parameter is zero, the variable number needs to added 1.

[in] number	Number of variables
[in] positionData	Multi-data structure

ESMultiBpexPositionData
Multi-data structure (Base/External-axis position)

FORMAT

```
#define Length_of_Multi_Bpex (13)

typedef struct
{
   ESBpexPositionDatadata[Length_of_Multi_Bpex];
} ESMultiBpexPositionData;
```

MEMBER <data[Length_of_Multi_Bpex]> Base/External-axis position data (max number = 13)

ESBpexPositionData
Base/External-axis position data structure

```
FORMAT : typedef struct
         {
             LONG  dataType;
             ESAxisData axesData;
          } ESBpexPositionData;
```

MEMBER : <dataType> Data type

Value	Explanation
0	Pulse
16	Coordinate（Base only）

<axesData> Axes data
ESAxisData
Axis data structure

```
FORMAT : #define Number_of_Axis (8)

         typedef struct
         {
            DOUBLE    axis[Number_of_Axis];
         } ESAxisData;
```

MEMBER : <axis[Number_of_Axis]> Axis data of base/external-axis (Size = 8)

Array	Pulse type	Coordinate type
axis[0]	1st axis Pulse	X-coordinateValue (unit: mm)
axis[1]	2nd axis Pulse	Y-coordinateValue (unit: mm)
axis[2]	3rd axis Pulse	Z-coordinateValue (unit: mm)
axis[3]	4th axis Pulse	
axis[4]	5th axis Pulse	
axis[5]	6th axis Pulse	
axis[6]	7th axis Pulse	
axis[7]	8th axis Pulse	

RETURN VALUE

0	: Normal completion
0xA001	: Out of variable number range
0xB004	: Including out of variable number range
Others	: Error codes

REFERENCE

"ESGetBpexPositionData" "ESSetBpexPositionData" "ESGetBpexPositionDataM"

■ ESSelectJob

Sets the specified job as a current job or a master job.

FORMAT

```
LONG  ESSelectJob( HANDLE handle, LONG jobType, LONG lineNo, CHAR* jobName );
```

ARGUMENTS

[in] handle	Target handle value
[in] jobType	Job type

Value	Explanation
1	Current job
10	Master job (Master)
11	Master job (Sub 1)
12	Master job (Sub 2)
13	Master job (Sub 3)
14	Master job (Sub 4)
15	Master job (Sub 5)
16	Master job (Sub 6)
17	Master job (Sub 7)
18	Master job (Sub 8)
19	Master job (Sub 9)
20	Master job (Sub 10)
21	Master job (Sub 11)
22	Master job (Sub 12)
23	Master job (Sub 13)
24	Master job (Sub 14)
25	Master job (Sub 15)

[in] lineNo	Line number (0 to 9999)
[in] jobName	Job name string data storage pointer (Max size = 32)

RETURN VALUE

0	: Normal completion
0xA001	: Out of variable number range
0xC800	: Value error
Others	: Error codes

REFERENCE

"ESGetJobStatus"

■ ESStartJob

Starts a job.

FORMAT

```
LONG  ESStartJob( HANDLE handle );
```

ARGUMENTS

[in] handle	Target handle value

ARGUMENTS

0	: Normal completion
Others	: Error codes

■ ESCartMove

Moves the robot to specified position. (Type Cartesian coordinates)

FORMAT

LONG ESCartMove(HANDLE handle, LONG moveType, ESCartMoveData moveData);

ARGUMENTS

[in] handle	Target handle value
[in] moveType	Move type

Value	Explanation
1	Link absolute position operation (MOVJ)
2	Straight absolute position operation (MOVL)
3	Straight increment value operation (IMOV)

[in] moveData	Move data structure

ESCartMoveData
Move data structure (type Cartesian coordinates)

FORMAT
```
typedef struct
{
   ESMoveData    moveData;
   ESCartPosData    robotPos;
   ESBaseData    basePos;
   ESStationData    stationPos;
} ESCartMoveData;
```

MEMBER <moveData> Move information data structure
ESMoveData
Move information data structure

FORMAT :
```
typedef struct
{
   LONG  robotNo;
   LONG  stationNo;
   LONG  speedType;
   DOUBLE  speed;
} ESMoveData;
```

MEMBER : <robotNo> Robot number (0, 1 to 8)

<stationNo> Station number (0, 1 to 24)

<speedType> Classification in speed

Value	Explanation
0	% (Link operation MOVJ)
1	V (Cartesian operation MOVL、IMOV)
2	VR (Cartesian operation MOVL、IMOV)

<speed> Specifying the speed

Unit	Explanation
%	% (Link operation MOVJ)
mm/s	V (Cartesian operation MOVL、IMOV)
°/s	VR (Cartesian operation MOVL、IMOV)

<robotData> Robot position data structure
ESCartPosData
Robot position data structure

```
FORMAT : typedef struct
            {
               LONG  dataType;
               LONG  fig;
               LONG  toolNo;
               LONG  userFrameNo;
               LONG  exFig;
               ESAxisData axesData;
            } ESCartPosData;
```

MEMBER : <dataType> Data type

Value	Explanation
16	Base-coordinate
17	Robot-coordinate
18	Tool-coordinate
19	User-coordinate

<fig> Figure

bitValue	Explanation
D00	0:Front side 1:Back side
D01	0:Elbow above 1:Elbow under
D02	0:Flip 1:No-flip
D03	0:R<180deg 1:R>=180deg
D04	0:T<180deg 1:T>=180deg
D05	0:S<180deg 1:S>=180deg
D06-D07	Reserved

<toolNo> Tool number (0 to 63)

<userFrameNo> User frame number (1 to 63)

<exFig> Extra figure

<axesData> Axes data of robot
ESAxisData
Axis data structure

FORMAT : #define Number_of_Axis (8)

```
typedef struct
{
   DOUBLE    axis[Number_of_Axis];
} ESAxisData;
```

MEMBER : <axis[Number_of_Axis]> Axis data of robot (Size = 8)

Array	Coordinate type
axis[0]	X-coordinate (unit: mm)
axis[1]	Y-coordinate (unit: mm)
axis[2]	Z-corrdinate (unit: mm)
axis[3]	Rx angle (unit: deg)
axis[4]	Ry angle (unit: deg)
axis[5]	Rz angle (unit: deg)
axis[6]	Re angle (unit: deg)
axis[7]	

<baseData> Base position data structure
ESBaseData
Base position data structure

FORMAT : #define Number_of_BaseAxis (3)

```
typedef struct
{
   DOUBLE    axis[Number_of_BaseAxis];
} ESBaseData;
```

MEMBER : <axis[Number_of_BaseAxis]> Axis data of base (Size = 3)

Array	Coordinate value
axis[0]	1st axis position (unit: mm)
axis[1]	2nd axis position (unit: mm)
axis[2]	3rd axis position (unit: mm)

<stationData> Station position data structure
ESStationData
Station position data structure

FORMAT : #define Number_of_StationAxis (6)

```
typedef struct
{
   DOUBLE    axis[Number_of_StationAxis];
} ESStationData;
```

MEMBER : <axis[Number_of_StationAxis]> Axis data of station (Size = 6)

Array	Pulse value
axis[0]	1st axis Pulse
axis[1]	2nd axis Pulse
axis[2]	3rd axis Pulse
axis[3]	4th axis Pulse
axis[4]	5th axis Pulse
axis[5]	6th axis Pulse

NOTE

- It is not able to operate the robot and the station at the same time. Setting the both operation at the same time receives the control group setting error (0xB008) from the DX100 or FS100.
- To move the base axes only, specify the robot number at the specifying control group, and input the current value to the following coordinate values.
  - X-coordinate value (unit: micron)
  - Y-coordinate value (unit: micron)
  - Z-coordinate value (unit: micron)
  - Rx angle value (unit: 0.0001deg)
  - Ry angle value (unit: 0.0001deg)
  - Rz angle value (unit: 0.0001deg)

RETURN VALUE

0	: Normal completion
0xB008	: Control group setting error
Others	: Error codes

REFERENCE

"ESPulseMove"

■ ESPulseMove

Moves the robot to specified position. (Type Pulse)

FORMAT

LONG ESPulseMove(HANDLE handle, LONG moveType, ESPulseMoveData moveData);

ARGUMENTS

[in] handle	Target handle value
[in] moveType	Move type

Value	Explanation
1	Link absolute position operation (MOVJ)
2	Straight absolute position operation (MOVL)

[in] moveData	Move data structure

ESPulseMoveData
Move data structure (Pulse)

FORMAT
```
typedef struct
{
   ESMoveData    moveData;
   ESPulsePosData    robotData;
   LONG    toolNo;
   ESBaseData    baseData;
   ESStationData    stationData;
} ESPulseMoveData;
```

MEMBER
<moveData> Move information data structure
ESMoveData
Move information data structure

```
FORMAT : typedef struct
         {
            LONG  robotNo;
            LONG  stationNo;
            LONG  speedType;
            DOUBLE  speed;
         } ESPulseMoveData;
```

MEMBER : <robotNo> Robot number (0, 1 to 8)

<stationNo> Station number (0, 1 to 24)

<speedType> Classification in speed

Value	Explanation
0	% (Link operation MOVJ)
1	V (Cartesian operation MOVL)
2	VR (Cartesian operation MOVL)

<speed> Specifying the speed

Unit	Explanation
%	% (Link operation MOVJ)
mm/s	V (Cartesian operation MOVL)
°/s	VR (Cartesian operation MOVL)

<robotData> Robot position pulse data structure
ESPulsePosData
Robot position pulse data structure

FORMAT : #define Number_of_Axis (8)

```
typedef struct
{
   DOUBLE    axis[Number_of_Axis];
} ESPulsePosData;
```

MEMBER : <axis[Number_of_Axis]> Axis data of robot (Size = 8)

Array	Explanation
axis[0]	1st axis Pulse
axis[1]	2nd axis Pulse
axis[2]	3rd axis Pulse
axis[3]	4th axis Pulse
axis[4]	5th axis Pulse
axis[5]	6th axis Pulse
axis[6]	7th axis Pulse
axis[7]	8th axis Pulse

<baseData> Base position pulse data structure
ESBaseData
Base position pulse data structure

FORMAT : #define Number_of_BaseAxis (3)

```
typedef struct
{
   DOUBLE    axis[Number_of_BaseAxis];
} ESBaseData;
```

MEMBER : <axis[Number_of_BaseAxis]> Axis data of base (Size = 3)

Array	Pulse value
axis[0]	1st axis Pulse
axis[1]	2nd axis Pulse
axis[2]	3rd axis Pulse

<stationData> Station position pulse data structure
ESStationData
Station position pulse data structure

```
FORMAT : #define Number_of_StationAxis (6)

         typedef struct
         {
            DOUBLE    axis[Number_of_StationAxis];
         } ESStationData;
```

MEMBER : <axis[Number_of_StationAxis]> Axis data of station (Size = 6)

Array	Pulse value
axis[0]	1st axis Pulse
axis[1]	2nd axis Pulse
axis[2]	3rd axis Pulse
axis[3]	4th axis Pulse
axis[4]	5th axis Pulse
axis[5]	6th axis Pulse

<toolNo> Tool number (0 to 63)

NOTE

- It is not able to operate the robot and the station at the same time. Setting the both operation at the same time receives the control group setting error (0xB008) from the DX100 or FS100.
- To move the base axes only, specify the robot number at the specifying control group, and input the current value to the following coordinate values.
  1st axis Pulse
  2nd axis Pulse
  3rd axis Pulse
  4th axis Pulse
  5th axis Pulse
  6th axis Pulse
  7th axis Pulse
  8th axis Pulse

RETURN VALUE

0	: Normal completion
0xB008	: Control group setting error
Others	: Error codes

REFERENCE

"ESCartMove"

5.3 I/O Signal Read/Write Function

Reads or writes the I/O signals.
The following functions are available.

- ESReadIO1
- ESReadIO2
- ESWriteIO1
- ESWriteIO2
- ESReadRegister
- ESWriteRegister
- ESReadIOM
- ESWriteIOM
- ESReadRegisterM
- ESWriteRegisterM

■ ESReadIO1

Reads I/O signals.

FORMAT

LONG ESReadIO1(HANDLE handle, LONG ioNumber, SHORT* ioData);

ARGUMENTS

[in] handle	Target handle value
[in] ioNumber	I/O address (Sets a signal divided by 10.)

Value	Explanation	
1 to 256	Robot universal input	#00010 to #02567(2048)
1001 to 1256	Robot universal output	#10010 to #12567(2048)
2001 to 2256	External input	#20010 to #22567(2048)
3001 to 3256	External output	#30010 to #32567(2048)
4001 to 4160	Robot specific input	#40010 to #41607(1280)
5001 to 5200	Robot specific output	#50010 to #52007(1600)
6001 to 6064	I/F panel input	#60010 to #60647(512)
7001 to 7999	Auxiliary relay	#70010 to #79997(7992)
8001 to 8064	Control status signal	#80010 to #80647(512)
8201 to 8220	Pseudo input signal	#82010 to #82207(160)
2501 to 2756	Network input	#25010 to #27567(2048)
3501 to 3756	Net work output	#35010 to #37567(2048)

[out] ioData	I/O signals Data storage pointer

RETURN VALUE

0	: Normal completion
0xA001	: Out of variable number range
Others	: Error codes

REMARKS

Restrictions
Check the "RS023" parameter before using this function. Use "ESReadIO1" or "ESReadIO2" as to the "RS023" parameter.

RS023	Function
0	ESReadIO1
1	ESReadIO2

REFERENCE

"ESReadIO2" "ESWriteIO1" "ESWriteIO2" "ESReadIOM" "ESWriteIOM"

■ ESReadIO2

Reads I/O signals.

FORMAT

LONG ESReadIO2(HANDLE handle, LONG ioNumber, SHORT* ioData);

ARGUMENTS

[in] handle	Target handle value
[in] ioNumber	I/O address (Sets a signal divided by 10.)

Value	Explanation	
1 to 256	Robot universal input	#00010 to #02567(2048)
1001 to 1256	Robot universal output	#10010 to #12567(2048)
2001 to 2256	External input	#20010 to #22567(2048)
3001 to 3256	External output	#30010 to #32567(2048)
4001 to 4160	Robot specific input	#40010 to #41607(1280)
5001 to 5200	Robot specific output	#50010 to #52007(1600)
6001 to 6064	I/F panel input	#60010 to #60647(512)
7001 to 7999	Auxiliary relay	#70010 to #79997(7992)
8001 to 8064	Control status signal	#80010 to #80647(512)
8201 to 8220	Pseudo input signal	#82010 to #82207(160)
2501 to 2756	Network input	#25010 to #27567(2048)
3501 to 3756	Net work output	#35010 to #37567(2048)

[out] ioData	I/O signals Data storage pointer

RETURN VALUE

0 : Normal completion
0xA001 : Out of variable number range
Others : Error codes

REMARKS

Restrictions
Check the "RS023" parameter before using this function. Use "ESReadIO1" or "ESReadIO2" as to the "RS023" parameter

RS023	Function
0	ESReadIO1
1	ESReadIO2

REFERENCE

"ESReadIO1" "ESWriteIO1" "ESWriteIO2" "ESReadIOM" "ESWriteIOM"

■ ESWriteIO1

Sets I/O signals.

FORMAT

LONG ESWriteIO1(HANDLE handle, LONG ioNumber, SHORT ioData);

ARGUMENTS

[in] handle	Target handle value
[in] ioNumber	I/O address (Sets a signal divided by 10.)

Value	Explanation
2501 to 2756	Network input #25010 to #27567(2048)

[out] ioData	I/O signals Data(0 to 255)

RETURN VALUE

0	: Normal completion
0xA001	: Out of variable number range
Others	: Error codes

REMARKS

Restrictions
Check the "RS023" parameter before using this function. Use "ESWriteIO1" or "ESWriteIO2" as to the "RS023" parameter.

RS023	Function
0	ESWriteIO1
1	ESWriteIO2

REFERENCE

"ESReadIO1" "ESReadIO2" "ESWriteIO2" "ESReadIOM" "ESWriteIOM"

■ ESWriteIO2

Sets I/O signals.

FORMAT

```
LONG  ESWriteIO2( HANDLE handle, LONG ioNumber, SHORT ioData );
```

ARGUMENTS

[in] handle	Target handle value
[in] ioNumber	I/O address (Sets a signal divided by 10.)

Value	Explanation	
2501 to 2756	Network input	#25010 to #27567(2048)

[out] ioData	I/O signals Data (0 to 255)

RETURN VALUE

0 : Normal completion
0xA001 : Out of variable number range
Others : Error codes

REMARKS

Restrictions
Check the "RS023" parameter before using this function. Use "ESWriteIO1" or "ESWriteIO2" as to the "RS023" parameter.

RS023	Function
0	ESWriteIO1
1	ESWriteIO2

REFERENCE

"ESReadIO1" "ESReadIO2" "ESWriteIO1" "ESReadIOM" "ESWriteIOM"

■ ESReadRegister

Reads a register data.

FORMAT

LONG ESReadRegister(HANDLE handle, LONG regNumber, UNSIGNED SHORT* regData);

ARGUMENTS

[in] handle	Target handle value
[in] regNumber	Register number

Value	Explanation
1 to 1000	RS022=0
0 to 999	RS022=1

NOTE Check the "RS022" parameter before using this function. Set the value as above. If the "RS022"parameter is zero, the variable number needs to added 1.

[out] regData	Register data storage pointer

RETURN VALUE

0	: Normal completion
0xA001	: Out of variable number range
Others	: Error codes

REFERENCE

"ESReadRegister" "ESReadRegisterM" "ESWriteRegisterM"

■ ESWriteRegister

Sets a register data.

FORMAT

LONG ESSetRegister(HANDLE handle, LONG regNumber, UNSIGNED SHORT regData);

ARGUMENTS

[in] handle	Target handle value
[in] regNumber	Register number

Value	Explanation
1 to 560	RS022=0
0 to 559	RS022=1

NOTE Check the "RS022" parameter before using this function. Set the value as above. If the "RS022"parameter is zero, the variable number needs to added 1.

[out] regData	Register data (0 to 65535)

RETURN VALUE

0	: Normal completion
0xA001	: Out of variable number range
Others	: Error codes

REFERENCE

"ESReadRegister" "ESReadRegisterM" "ESWriteRegisterM"

■ ESReadIOM

Reads I/O signals continuously from the specified I/O address.

FORMAT

LONG ESReadIOM(HANDLE handle, LONG ioNumber, LONG number, ESMultiByte-Data* ioData);

ARGUMENTS

[in] handle	Target handle value
[in] ioNumber	I/O address (Sets a signal divided by 10.)

Value	Explanation
1 to 256	Robot universal input #00010 to #02567(2048)
1001 to 1256	Robot universal output #10010 to #12567(2048)
2001 to 2256	External input #20010 to #22567(2048)
3001 to 3256	External output #30010 to #32567(2048)
4001 to 4160	Robot specific input #40010 to #41607(1280)
5001 to 5200	Robot specific output #50010 to #52007(1600)
6001 to 6064	I/F panel input #60010 to #60647(512)
7001 to 7999	Auxiliary relay #70010 to #79997(7992)
8001 to 8064	Control status signal #80010 to #80647(512)
8201 to 8220	Pseudo input signal #82010 to #82207(160)
2501 to 2756	Network input #25010 to #27567(2048)
3501 to 3756	Net work output #35010 to #37567(2048)

[in] number Number of I/O signals group

NOTE It needs that Number of I/O signals group is even.

[out] ioData Multi-data storage pointer

ESMultiByteData
Multi-data structure (1byte)

FORMAT
```
#define Length_of_Multi_1 (474)

typedef struct
{
    CHAR    data[Length_of_Multi_1];
} ESMultiByteData;
```

MEMBER <data[Length_of_Multi_1]> Data (1byte) (Max number = 474)

RETURN VALUE

0	: Normal completion
0xA001	: Out of variable number range
0xB004	: Including out of variable number range
Others	: Error codes

REFERENCE

"ESReadIO1" "ESReadIO2" "ESWriteIO1" "ESWriteIO2" "ESWriteIOM"

■ ESWriteIOM

Sets I/O signals continuously from the specified I/O address.

FORMAT

LONG ESWriteIOM(HANDLE handle, LONG ioNumber, LONG number, ESMultiByteData ioData);

ARGUMENTS

[in] handle	Target handle value
[in] ioNumber	I/O address (Sets a signal divided by 10.)

Value	Explanation	
2501 to 2756	Network input	#25010 to #27567(2048)

[in] number	Number of I/O signals group

NOTE It needs that Number of I/O signals group is even.

[in] ioData	Multi-data structure

ESMultiByteData
Multi-data structure (1byte)

FORMAT
```
#define Length_of_Multi_1 (474)

typedef struct
{
   CHAR    data[Length_of_Multi_1];
} ESMultiByteData;
```

MEMBER <data[Length_of_Multi_1]> Data (1byte) (Max number = 474)

RETURN VALUE

0	: Normal completion
0xA001	: Out of variable number range
0xB004	: Including out of variable number range
Others	: Error codes

REFERENCE

"ESReadIO1" "ESReadIO2" "ESWriteIO1" "ESWriteIO2" "ESReadIOM"

■ ESReadRegisterM

Reads register data continuously from the specified number.

FORMAT

LONG ESReadRegisterM(HANDLE handle, LONG regNumber, LONG number, ESMultiUShortData* regData);

ARGUMENTS

[in] handle	Target handle value
[in] regNumber	Register number

Value	Explanation
1 to 1000	RS022=0
0 to 999	RS022=1

NOTE Check the "RS022" parameter before using this function. Set the value as above. If the "RS022"parameter is zero, the variable number needs to added 1.

[in] number	Number of register
[out] regData	Multi-data storage pointer

ESMultiUShortData
Multi-data structure (2byte, unsigned)

FORMAT

```
#define Length_of_Multi_2 (237)

typedef struct
{
    UNSIGNED SHORT    data[Length_of_Multi_2];
} ESMultiUShortData;
```

MEMBER <data[Length_of_Multi_2]> Data (2byte, unsigned) (Max number = 237)

RETURN VALUE

0	: Normal completion
0xA001	: Out of variable number range
0xB004	: Including out of variable number range
Others	: Error codes

REFERENCE

"ESReadRegister" "ESWriteRegister" "ESWriteRegisterM"

■ ESWriteRegisterM

Sets register data continuously from the specified number.

FORMAT

LONG ESWriteRegisterM(HANDLE handle, LONG regNumber, LONG number, ESMultiUShortData regData);

ARGUMENTS

[in] handle	Target handle value
[in] regNumber	Register number

Value	Explanation
1 to 560	RS022=0
0 to 559	RS022=1

NOTE Check the "RS022" parameter before using this function. Set the value as above. If the "RS022"parameter is zero, the variable number needs to added 1.

[in] number	Number of register
[in] regData	Multi-data structure

ESMultiUShortData
Multi-data stcture (2byte, unsigned)

FORMAT

```
#define Length_of_Multi_2 (237)

typedef struct
{
   UNSIGNED SHORT    data[Length_of_Multi_2];
} ESMultiUShortData;
```

MEMBER <data[Length_of_Multi_2]> Data (2byte, unsigned) (Max size = 237)

RETURN VALUE

0	: Normal completion
0xA001	: Out of variable number range
0xB004	: Including out of variable number range
Others	: Error codes

REFERENCE

"ESReadRegister" "ESWriteRegister" "ESReadRegisterM"

5.4 File Data Transmission Function

Loads and saves the files containing job, condition data, system information, etc.
The following functions are availble.

- ESDeleteJob
- ESLoadFile
- ESSaveFile
- ESFileListFirst
- ESFileListNext

■ ESDeleteJob

Deletes the specified job.

FORMAT

```
LONG  ESDeleteJob( HANDLE handle, CHAR* jobName );
```

ARGUMENTS

[in] handle	Target handle value
[in] jobName	Job name string data storage pointer (Max size = 32)

RETURN VALUE

0	: Normal completion
0xE2B3	: No file
Others	: Error codes

REMARKS

Restrictions
Change to the remote mode before executing this function.

■ ESLoadFile

Sends the specified file to the robot controller.

FORMAT

```
LONG  ESLoadFile( HANDLE handle, CHAR* filePath );
```

ARGUMENTS

[in] handle	Target handle value
[in] filePath	Full path storage pointer of sent file

RETURN VALUE

0	: Normal completion
Others	: Error codes

REFERENCE

"ESSaveFile"

■ ESSaveFile

Receives the specified file from the robot controller.

FORMAT

```
LONG  ESSaveFile( HANDLE handle, CHAR* savePath, CHAR* fileName );
```

ARGUMENTS

[in] handle	Target handle value
[in] savePath	Full path storage pointer to save the received file
[in] fileName	File name string data storage pointer

RETURN VALUE

0	: Normal completion
0xE2B3	: No file
Others	: Error codes

REFERENCE

"ESLoadFile"

■ ESFileListFirst

Refreshes the file list and reads the first file name of the list.

FORMAT

```
LONG  ESFileListFirst( HANDLE handle, LONG fileType, CHAR* fileName );
```

ARGUMENTS

[in] handle	Target handle value
[in] fileType	File type

Value	Explanation
1	*.JBI
2	*.DAT
3	*.CND
4	*.PRM
5	*.SYS
6	*.LST

[out] fileName	File name string data storage pointer

RETURN VALUE

0 : Normal completion
0xE2A7 : No file list
Others : Error codes

REFERENCE

"ESFileListNext"

■ ESFileListNext

Reads the file name of the list.

FORMAT

```
LONG  ESFileListNext( HANDLE handle, CHAR* fileName );
```

ARGUMENTS

[in] handle	Target handle value
[out] fileName	File name string data storage pointer

RETURN VALUE

0	: Normal completion
0xFFFF	: No file list
Others	: Error codes

REMARKS

Call Condition
The ESFileListFirst function must be called up and the file list must be refreshed before executing this function.

REFERENCE

"ESFileListFirst"

5.5 Other Functions

The following functions are also available.

ESOpen
ESClose
ESSetTimeOut

■ ESOpen

Opens the connection and Gets a communication handler.

FORMAT

```
LONG  ESOpen(LONG controllerType, CHAR* ipAddress, HANDLE*handle );
```

ARGUMENTS

[in] controllerType   Controller Type

Value	Explanation
1	DX100
2	FS100

[in] ipAddress   IP address

[out] handle   Communication handler

RETURN VALUE

0	: Normal completion
0x9000	: Connection Error
Others	: Error codes

REFERENCE

"ESClose"

■ ESClose

Close the connection of the specified communication handler.

FORMAT

```
LONG  ESClose( HANDLEhandle );
```

ARGUMENTS

[in] handle	Communication handler

RETURN VALUE

0	: Normal completion
Others	: Error codes

REFERENCE

"ESOpen"

■ ESSetTimeOut

Sets a communication contol timer or retry counter.

FORMAT

```
LONG  ESSetTimeOut ( HANDLE handle, LONG timeOut, LONG retry )
```

ARGUMENTS

[in] handle	Target handle value
[in] timeOut	Time out?ms?
[in] retry	Number of retry

RETURN VALUE

0	: Normal completion
Others	: Error codes

REMARKS

Initial Value

timeout	500(msec)
retry	3

NOTE This function is used to change the parameters of MOTOCOMES on the personal computer.
To change the robot controller transmission parameters (control timers, retry counter), use the programming pendant of the robot controller.

6 Appendix

6.1 Procedure to replace MOTOCOM32 with MOTOCOMES

The interface functions of MOTOCOMES DLL are incompatible with those of MOTOCOM32 DLL.
The main differences and procedure to replace MOTOCOM32 with MOTOCOMES are as follows.

6.1.1 Differences

■ Procedure to connect

MOTOCOMES can connect to controller by Ethernet only. So procedure to connect is simplified.
MOTOCOM32 (case of Ethernet)
    BscOpen (gets a communication handler)
    BscSetEther (set the parameters for Ethernet connection)
    BscConnect (connect to controller)

MOTOCOM ES
    ESOpen (connect to controller using connection parameters)

■ Communication handler

The type of communication handler is "short" for MOTOCOM32 and "HANDLE" for MOTOCOMES.

■ Unsupported function

The following functions are not supported by MOTOCOMES.
- Read/Write a user coordinate data
- Read the names of jobs related to the parent job
- Relative job conversion
- Check operations of manipulator
- Select a mode

6.1.2 Procedure to replace

A procedure is explained by an example in the case to create an application in C++.

■ Files to need to create an application

	MOTOCOM32	MOTOCOMES
Definition of functions (include file)	MOTOCOM.h	MOTOCOMES.h
Library to link	MOTOCOM32.lib	MOTOCOMES.lib
DLL file	MOTOCO32.dll	MOTOCOMES.dll

■ Procedure to connect

MOTOCOM32 needs 3 steps to connect to controller as follows.

1. Set the current directory and get a communication handler.
2. Set the communication parameters.
3. Connect to controller.

The type of communication handler is "short".

Case of Ethernet connection:

```
short nCid;
short rc = 0;
char cur_dir[_MAX_DIR];
char *IPAddress="255.255.255.255";

_getcwd( cur_dir, _MAX_DIR );
nCid = BscOpen( cur_dir, PACKETETHERNET );
rc = BscSetEther( nCid, IPAddress, FuncMode, GetSafeHwnd() );
rc = BscConnect( nCid );
```

Meanwhile, MOTOCOMES make procedure to connect by one function. Setting the current directory is not needed. The type of communication handler is "HANDLE".

```
HANDLE handle;
long rc = 0;
char *IPAddress="255.255.255.255";

rc = ESOpen( 1, IPAddress, &handle );
```

	MOTOCOM32	MOTOCOMES
Functions to connect	BscOpen BscSetCom (serial) BscSetEther (Ethernet) BscSetEServer (ethernet server) BscConnect	ESOpen

■ Procedure to disconnect

For disconnect to controller, each call the function to disconnect.

	MOTOCOM32	MOTOCOMES
Functions to disconnect	BscClose	ESClose

6.1.3 Correspondence of interface functions

The function of MOTOCOM32 and the function of MOTOCOMES which offers an equivalent function are shown below. The function that is put together two or more functions and that is divided into two or more functions are contained in these.

The new function of MOTOCOMES is not described.

MOTOCOM32	MOTOCOMES
File Data Transmission Function	
BscDownload BscDownloadEx	ESLoadFile
BscUpload BscUploadEx	ESSaveFile
Robot Control Function	
Status Read	
BscFindFirst	ESFileListFirst
BscFindNext	ESFileListNext
BscFindFirstMaster	No corresponding function
BscFindNextMaster	No corresponding function
BscGetCtrlGroup BscGetCtrlGroupXrc BscGetCtrlGroupDX BscIsCtrlGroup BscIsCtrlGroupXrc BscIsCtrlGroupDX BscIsTaskInf BscIsTaskInfXrc	Target control group and task can be set by each function.
BscGetError BscGetError2 BscReadAlarmS BscGetFirstAlarm BscGetNextAlarm BscGetFirstAlarmS BscGetNextAlarmS	ESGetAlarm (alarm only) ESGetAlarmEx (for applying the sub code character strings)
BscGetStatus	ESGetStatus
BscGetUFrame	No corresponding function
BscGetVarData BscGetVarData2 BscHostGetVarData BscGetVarDataEx	ESGetVarData1 (B,I,D,R) ESGetVarData2 (B,I,D,R) ESGetStrData (S) ESGetPositionData (P) ESGetBpexPositionData (BP,EX)
BscHostGetVarDataM	ESGetVarDataMB (B) ESGetVarDataMI (I) ESGetVarDataMD (D) ESGetVarDataMR (R)
BscIsAlarm	ESGetStatus
BscIsCycle	ESGetStatus
BscIsError	ESGetStatus
BscIsHold	ESGetStatus
BscIsJobLine	ESGetJobStatus
BscIsJobName	ESGetJobStatus
BscIsJobStep	ESGetJobStatus
BscIsLoc BscGetPulsePos	ESGetPosition
BscIsPlayMode	ESGetStatus
BscIsTeachMode	ESGetStatus
BscIsRemoteMode	ESGetStatus

BscIsRobotPos BscGetCartPos	ESGetPosition(Base coordinate only)
BscIsServo	ESGetStatus
BscJobWait	No corresponding function
System Control	
BscCancel	ESCancel
BscChangeTask	Target task can be set by each function.
BscContinueJob	ESStartJob
BscConvertJobP2R BscConvertJobR2P	No corresponding function
BscDeleteJob	ESDeleteJob
BscHoldOff BscHoldOn	ESHold
BscPutVarData BscPutVarData2 BscHostPutVarData BscPutVarDataEx	ESSetVarData1 (B,I,D,R) ESSetVarData2 (B,I,D,R) ESSetStrData (S) ESSetPositionData (P) ESSetBpexPositionData (BP,EX)
BscHostPutVarDataM	ESSetVarDataMB (B) ESSetVarDataMI (I) ESSetVarDataMD (D) ESSetVarDataMR (R)
BscImov BscImovEx BscImovEx2 BscMov BscMovEx BscMovEx2 BcsMovj BscMovjEx BscMovl BscMovlEx BscPMov BscPmovEx BscPmovj BscPMovjEx BscPMovl BscPMovlEx	ESCartMove ESPulseMove (It is not able to operate the robot and the station at the same time.)
BscMDSP	ESBDSP
BscOPLock BscOPUnLock	ESHlock
BscPutUFrame BscPutUFrameEx2	No corresponding function
BscStartJob	ESStartJob
BscSelectJob	ESSelectJob
BscSelectMode	No corresponding function
BscSelLoopCycle BscSelOneCycle BscSelStepCycle	ESCycle
BscSetLineNumber	ESSelectJob
BscSetMasterJob	ESSelectJob
BscReset	ESReset
BscSetCtrlGroup BscSetCtrlGroupXrc BscSetCtrlGroupDX	Target control group and task can be set by each function.
BscServoOff BscServoOn	ESServo
I/O Signal Read/Write Function	

BscReadIO BscReadIO2	ESReadIO1 ESReadIO2 ESReadIOM
BscWriteIO BscWriteIO2	ESWriteIO1 ESWriteIO2 ESWriteIOM

6.2 Frequently-asked questions

■ When the driver has been installed with USB type key connected to a personal computer

1. With the USB type key attached to a personal computer, delete the item registered as"USB Token" in Device Manager.
2. Uninstall the driver (Sentinel System Driver) with "Add/Remove Programs".
3. Install the driver with key detached from personal computer.

(For installation, refer to " 1.3 Hardware Lock Key " in the following section.)

■ When the previous version key driver has been installed after install-ing the key driver

Although it is rare, there may be some trouble in this case.
Uninstall the driver(Sentinel System Driver) with "Add / Remove Programs".

(For installation, refer to " 1.3 Hardware Lock Key " in the following section.)

YASKAWA

MOTOCOM ES
OPERATION MANUAL

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

YASKAWA America Inc. MOTOMAN Robotics Division
805 Liberty Lane, West Carrollton, OH 45449, U.S.A.
Phone +1-937-847-6200 Fax +1-937-847-6277

YASKAWA Nordic AB
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.
Kammerfeldstr. 1, 80591 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

Shougang MOTOMAN Robot Co. Ltd.
No.7, Yongchang-North Road, Beijing E&T Development Area, China 100176
Phone +86-10-6788-0548 Fax +86-10-6788-0548-813

YASKAWA ELECTRIC (SHANGHAI) Co., Ltd.
No.18Xizang Zhong Road, 17F, Harbour Ring Plaza, Shanghai 200001, CHINA
Phone +86-21-5385-0655 Fax +86-21-5385-2770

YASKAWA Robotics India Ltd.
#426, Udyog Vihar, Phase- IV, Gurgaon, Haryana, India
Phone +91-124-475-8500 Fax +91-124-414-8016

YASKAWA ELECTRIC CORPORATION

YASKAWA

© Printed in Japan May 2008 08-03