MOTOCOM ES
OPERATOR’S MANUAL

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

Please have the following information available when contacting Yaskawa Customer Support:

- System
- Primary Application
- Software Version (Located on Programming Pendant by selecting: (Main Menu) - (System Info) - (Version))
- Robot Serial Number (Located on robot data plate)
- Robot Sales Order Number (Located on controller data plate)

Part Number: 175525-1CD
Revision: 1
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 the Chapter 1: Safety of Controller Instructions. To ensure correct and safe operation, carefully read Controller Instructions before reading this manual.

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.

---

This instruction manual is applicable to both FS100 (a controller for small-sized manipulators) and FS100L (a controller for large and medium-sized manipulators). The description of “FS100” refers to both “FS100” and “FS100L” in this manual unless otherwise specified.
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 “DANGER”, “WARNING”, “CAUTION”, or "NOTICE".

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

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

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

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

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

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

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

<table>
<thead>
<tr>
<th>Item</th>
<th>Manual Designation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Menu</td>
<td>The menus displayed on screen are denoted with {}. ex. {TOOL}.</td>
</tr>
<tr>
<td>Button</td>
<td>The buttons, check boxes, radio buttons displayed on screen are denoted with [ ].</td>
</tr>
<tr>
<td></td>
<td>ex. [Close]; [Sync] check box; [Fast] radio button.</td>
</tr>
</tbody>
</table>

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 ................................................. 9
1.2 Hardware Requirements for MOTOCOM ES ........ 10
1.3 Hardware Lock Key ................................. 11

2 SETUP
2.1 Installing MOTOCOM ES ................................. 12
2.2 Environmental Settings for Use of Ethernet .......... 13
   2.2.1 MOTOCOM ES Application Settings .............. 13
       ■ parameter Setting .................................. 13
   2.2.2 Personal Computer Settings ................. 13
       ■ Hardware settings ................................ 13
       ■ Windows Network settings ..................... 13
   2.2.3 Robot Controller Setting ...................... 15
       ■ Hardware settings ................................ 15
       ■ Communication parameter settings .......... 16
       ■ Parameter settings ................................. 16
       ■ Command Remote Setting ...................... 16
   2.2.4 Network Setting ................................. 16
2.3 Restriction ............................................. 17
   2.3.1 YRC1000, DX200, DX100, FS100 and Personal Computer Restrictions ............................ 17
       ■ The port used for TCP/IP ......................... 17
   2.3.2 Personal Computer Restrictions .............. 17
       ■ Same file access .................................. 17
   2.3.3 YRC1000, DX200, DX100, FS100 Restrictions ............................................ 17
       ■ CMOS batch storage ............................... 17
2.4 Execution of MOTOCOM ES Programs .......... 18

3 Operation Of Host Control Function
3.1 Host Control Function ................................. 19
3.2 Startup and Exit ..................................... 19
   ■ Startup .............................................. 19
   ■ Exit .................................................. 19
3.3 Robot Control Function .............................. 20
3.4 Read/Write of I/O Signals ......................... 22
   ■ List of I/O Signals that can be Read or Written . 22
   ■ List of Register that can be Read or Written .... 23
   ■ I/O signal read/write display ................... 24
   ■ Register read/write display .................... 25
3.5 File Data Transmission Function
- Fileload ........................................... 26
- Filesave ........................................... 26
- Delete Job ........................................... 27
3.6 Macro function ........................................... 27
3.7 Environmental Settings
- Setting transmission parameters ................. 28
- Setting controller ................................... 28
- Selecting a folder .................................... 28
3.8 Version Information ........................................... 28

4 CREATING A TRANSMISSION APPLICATION
4.1 Outline ........................................... 29
4.2 Using Visual C++ ........................................... 30
  4.2.1 Preparation ........................................... 30
   - Creation of Skelton ................................... 30
   - Definition of DLL Call ................................... 30
  4.2.2 How to Create a transmission application ................. 31
   - Editing with a Dialog Box ......................... 31
   - Addition of Functions and Variables ......... 31
   - Creation and Execution of EXE File .......... 32
4.3 Using Visual C# ........................................... 33
  4.3.1 Preparation ........................................... 33
   - Creation of Project ................................... 33
   - Reference configuration of Library .......... 33
  4.3.2 How to Create a transmission application ................. 33
   - Creation of Form Module ......................... 33
   - Creation and Execution of EXE File .......... 34
  4.3.3 Structure ........................................... 35
4.4 Each Function Program List ................. 36
  - CTestDlg::OnLoadfile () ......................... 36
  - CTestDlg::OnSavefile () ......................... 37
  - Cmd_LoadFile_Click () ......................... 38
  - Cmd_SaveFile_Click () ......................... 39

5 COMMUNICATION TRANSMISSION
5.1 Outline ........................................... 41
5.2 Robot Control Function ................. 42
  - ESGetAlarm ........................................... 43
  - ESGetAlarmHist ........................................... 45
  - ESGetStatus ........................................... 47
  - ESGetJobStatus ........................................... 48
5.3 I/O Signal Read/Write Function
5.4 File Data Transmission Function

- ESDeleteJob ................................................. 141
- ESSaveFile ................................................. 143
- ESWriteRegisterM ........................................... 139
- ESRecipeRegister ............................................ 138
- ESWriteIOM ................................................. 137
- ESWriteRegister ............................................. 134
- ESReadIOM .................................................. 135
- ESWriteIOM .................................................. 137
- ESWriteRegister ............................................. 134
- ESDeleteJob ................................................. 141
- ESRecipeRegister ............................................ 139
- ESWriteIOM .................................................. 137
- ESWriteRegister ............................................. 134
- ESReadIOM .................................................. 135

5.5 Other Functions ......................................... 146

- ESSetTimeOut ............................................... 149
- ESRecipeRegister ............................................ 139
- ESWriteIOM .................................................. 137
- ESWriteRegister ............................................. 134
- ESReadIOM .................................................. 135
- ESWriteRegister ............................................. 134

6 Appendix

6.1 Procedure to replace MOTOCOM32 with MOTOCOMES ......................................... 150

6.1.1 Differences ............................................... 150

- Procedure to connect ....................................... 150
- Communication handler ..................................... 150
- Unsupported function ...................................... 150

6.1.2 Procedure to replace ................................... 150

- Files to need to create an application ...................... 151
- Procedure to connect ....................................... 151
- Procedure to disconnect ................................... 151

6.1.3 Correspondence of interface functions .................. 152

6.2 Frequently-asked questions ................................. 155

- When the driver has been installed with USB type key connected to a personal computer .......... 155
- When the previous version key driver has been installed after installing the key driver ............ 155
INTRODUCTION

1.1 MOTOCOM ES

MOTOCOM ES is software for data transmission between a personal computer and YASKAWA industrial robot controller YRC1000, DX200, DX100, FS100. High-speed transmission can be achieved by connecting the YRC1000, DX200, DX100, FS100 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

<table>
<thead>
<tr>
<th>Hardware Item</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS</td>
<td>Microsoft Windows 10 (64bit)</td>
</tr>
<tr>
<td></td>
<td>Microsoft Windows 7 ServicePack1 (32bit / 64bit)</td>
</tr>
<tr>
<td></td>
<td>JAPANESE and ENGLISH Windows version are supported only.*1</td>
</tr>
<tr>
<td>Required Memory</td>
<td>128 Mbyte or more</td>
</tr>
<tr>
<td>Hardware Disk Capacity for Installation</td>
<td>50 Mbyte or more</td>
</tr>
<tr>
<td>Display</td>
<td>Supported by MS-Windows</td>
</tr>
<tr>
<td>Robot Controller</td>
<td>YRC1000, DX200, DX100, FS100</td>
</tr>
<tr>
<td>Transmission Cable</td>
<td>Ethernet cable</td>
</tr>
<tr>
<td>Hardware Lock Key</td>
<td>Used under single user environment. For details, refer to the following section &quot;1.3 Hardware Lock Key&quot;.</td>
</tr>
</tbody>
</table>

*1 MS-Windows 10/7 is a registered trademark of Microsoft Corporation, 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.
- MOTOCOM ES module is 32-bit application. When users create the transmission software in 64-bit, MOTOCOM ES functions can not be called directly.

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>**

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

Execute "\SentinelDriver\Sentinel System Driver Installer 7.5.9.exe" of installation CD-ROM. Refer to "\SentinelDriver\Manual\Sentinel System Driver ReadMe.pdf" for the details of installation.
```

- 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 manuaarl "\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 ES] 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]. (In the case of Windows10, all menu appears directly under [Motoman].)

**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.
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](image)

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 YRC1000 use the Ethernet connector on the ACP01 board; for the DX200 use the Ethernet connector on the YCP21 board; for the DX100 use the Ethernet connector on the YCP01 board; for the FS100 use the Ethernet connector on the CPU201R board; for the DX100 use the Ethernet connector on the YCP01 board.

“YRC1000 Options: Instructions for Binary Ethernet Server”
“DX200 Options: Instructions for Binary Ethernet Server”
“DX100 Options: Instructions for Binary Ethernet Server”
“FS100 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.

- "YRC1000 Options: Instructions for Binary Ethernet Server"
- "DX200 Options: Instructions for Binary Ethernet Server"
- "DX100 Options: Instructions for Binary Ethernet Server"
- "FS100 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.

- "YRC1000 Options: Instructions for Data Transmission Function"
- "DX200 Options: Instructions for Data Transmission Function"
- "DX100 Options: Instructions for Data Transmission Function"
- "FS100 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.

- "YRC1000 Options: Instructions for Ethernet Function"
- "DX200 Options: Instructions for Ethernet Function"
- "DX100 Options: Instructions for Ethernet Function"
- "FS100 Options: Instructions for Binary Ethernet Function"
2.3 Restriction

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

2.3.1 YRC1000, DX200, DX100, FS100 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 YRC1000, DX200, DX100, FS100 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], [MOTOCOM32], [MotoComES], and then [Host Control ES] (In the case of Windows10, [Motoman] then [Host Control ES]).

- **Exit**

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

"YRC1000 Options: Instructions for Ethernet Function"
"DX200 Options: Instructions for Ethernet Function"
"DX100 Options: Instructions for Ethernet Function"
"FS100 Options: Instructions for Data Transmission Function"

<table>
<thead>
<tr>
<th>Reading Status</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read, monitoring system</td>
<td>ESGetAlarm</td>
</tr>
<tr>
<td></td>
<td>ESGetAlarmHist</td>
</tr>
<tr>
<td></td>
<td>ESGetStatus</td>
</tr>
<tr>
<td></td>
<td>ESGetJobStatus</td>
</tr>
<tr>
<td></td>
<td>ESGetConfiguration</td>
</tr>
<tr>
<td></td>
<td>ESGetPosition</td>
</tr>
<tr>
<td></td>
<td>ESGetDeviation</td>
</tr>
<tr>
<td></td>
<td>ESGetTorque</td>
</tr>
<tr>
<td></td>
<td>ESGetMonitoringTime</td>
</tr>
<tr>
<td></td>
<td>ESGetSystemInfo</td>
</tr>
<tr>
<td></td>
<td>ESGetAlarmEx</td>
</tr>
<tr>
<td></td>
<td>ESGetAlarmHistEx</td>
</tr>
<tr>
<td>Read, data access system</td>
<td>ESGetVarData</td>
</tr>
<tr>
<td></td>
<td>ESGetStrData</td>
</tr>
<tr>
<td></td>
<td>ESGetStrData2</td>
</tr>
<tr>
<td></td>
<td>ESGetPositionData</td>
</tr>
<tr>
<td></td>
<td>ESGetBpexPositionData</td>
</tr>
<tr>
<td></td>
<td>ESGetMultiData</td>
</tr>
<tr>
<td>Control of system</td>
<td>Operation system</td>
</tr>
<tr>
<td>Operation system</td>
<td>ESSetVarData</td>
</tr>
<tr>
<td></td>
<td>ESSetStrData</td>
</tr>
<tr>
<td></td>
<td>ESSetStrData2</td>
</tr>
<tr>
<td></td>
<td>ESSetPositionData</td>
</tr>
<tr>
<td></td>
<td>ESSetBpexPositionData</td>
</tr>
<tr>
<td></td>
<td>ESSetMultiData</td>
</tr>
<tr>
<td>Editing system</td>
<td>ESSelectJob</td>
</tr>
<tr>
<td></td>
<td>ESSStartJob</td>
</tr>
<tr>
<td></td>
<td>ESPStartJob</td>
</tr>
<tr>
<td></td>
<td>ESPulseMove</td>
</tr>
<tr>
<td>Job selection system</td>
<td>ESSelectJob</td>
</tr>
<tr>
<td></td>
<td>ESSStartJob</td>
</tr>
<tr>
<td></td>
<td>ESPulseMove</td>
</tr>
<tr>
<td>Startup system</td>
<td>ESSelectJob</td>
</tr>
<tr>
<td></td>
<td>ESSStartJob</td>
</tr>
<tr>
<td></td>
<td>ESPulseMove</td>
</tr>
</tbody>
</table>
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**

### YRC1000

<table>
<thead>
<tr>
<th>Signal</th>
<th>Range</th>
<th>Name</th>
<th>Read</th>
<th>Write</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xxx</td>
<td>00010-05127(4096)</td>
<td>Robot universal input</td>
<td>☐</td>
<td>☯</td>
</tr>
<tr>
<td>1xxx</td>
<td>10010-15127(4096)</td>
<td>Robot universal output</td>
<td>☐</td>
<td>☯</td>
</tr>
<tr>
<td>2xxx</td>
<td>20010-25127(4096)</td>
<td>External input</td>
<td>☐</td>
<td>☯</td>
</tr>
<tr>
<td>27xx</td>
<td>27010-29567(2048)</td>
<td>Network input</td>
<td>☐</td>
<td>☯</td>
</tr>
<tr>
<td>3xxx</td>
<td>30010-35127(4096)</td>
<td>External output</td>
<td>☐</td>
<td>☯</td>
</tr>
<tr>
<td>37xx</td>
<td>37010-39567(2048)</td>
<td>Network input</td>
<td>☐</td>
<td>☯</td>
</tr>
<tr>
<td>4xxx</td>
<td>40010-42567(2048)</td>
<td>Robot specific input (System)</td>
<td>☐</td>
<td>☯</td>
</tr>
<tr>
<td>5xxx</td>
<td>50010-55127 (4096)</td>
<td>Robot specific output (System)</td>
<td>☐</td>
<td>☯</td>
</tr>
<tr>
<td>6xxx</td>
<td>60010-60647(512)</td>
<td>I/F panel input</td>
<td>☐</td>
<td>☯</td>
</tr>
<tr>
<td>7xxx</td>
<td>70010-79997 (7992)</td>
<td>Auxiliary relay (System)</td>
<td>☐</td>
<td>☯</td>
</tr>
<tr>
<td>8xxx</td>
<td>80010-85127 (4096)</td>
<td>Control status signal (System)</td>
<td>☐</td>
<td>☯</td>
</tr>
<tr>
<td>87xx</td>
<td>87010-87207(160)</td>
<td>Pseudo input signal (System)</td>
<td>☐</td>
<td>☯</td>
</tr>
</tbody>
</table>

### DX200

<table>
<thead>
<tr>
<th>Signal</th>
<th>Range</th>
<th>Name</th>
<th>Read</th>
<th>Write</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xxx</td>
<td>00010-05127(4096)</td>
<td>Robot universal input</td>
<td>☐</td>
<td>☯</td>
</tr>
<tr>
<td>1xxx</td>
<td>10010-15127(4096)</td>
<td>Robot universal output</td>
<td>☐</td>
<td>☯</td>
</tr>
<tr>
<td>2xxx</td>
<td>20010-25127(4096)</td>
<td>External input</td>
<td>☐</td>
<td>☯</td>
</tr>
<tr>
<td>27xx</td>
<td>27010-29567(2048)</td>
<td>Network input</td>
<td>☐</td>
<td>☯</td>
</tr>
<tr>
<td>3xxx</td>
<td>30010-35127(4096)</td>
<td>External output</td>
<td>☐</td>
<td>☯</td>
</tr>
<tr>
<td>37xx</td>
<td>37010-39567(2048)</td>
<td>Network input</td>
<td>☐</td>
<td>☯</td>
</tr>
<tr>
<td>4xxx</td>
<td>40010-41607(1280)</td>
<td>Robot specific input (System)</td>
<td>☐</td>
<td>☯</td>
</tr>
<tr>
<td>5xxx</td>
<td>50010-53007 (2400)</td>
<td>Robot specific output (System)</td>
<td>☐</td>
<td>☯</td>
</tr>
<tr>
<td>6xxx</td>
<td>60010-60647(512)</td>
<td>I/F panel input</td>
<td>☐</td>
<td>☯</td>
</tr>
<tr>
<td>7xxx</td>
<td>70010-79997 (7992)</td>
<td>Auxiliary relay (System)</td>
<td>☐</td>
<td>☯</td>
</tr>
<tr>
<td>8xxx</td>
<td>80010-80647 (512)</td>
<td>Control status signal (System)</td>
<td>☐</td>
<td>☯</td>
</tr>
<tr>
<td>82xx</td>
<td>82010-82207(160)</td>
<td>Pseudo input signal (System)</td>
<td>☐</td>
<td>☯</td>
</tr>
</tbody>
</table>
### DX100, FS100

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

#### List of Register that can be Read or Written

<table>
<thead>
<tr>
<th>Range</th>
<th>Name</th>
<th>Read</th>
<th>Write</th>
</tr>
</thead>
<tbody>
<tr>
<td>M000-M559 (560)</td>
<td>Robot universal register</td>
<td>✗</td>
<td></td>
</tr>
<tr>
<td>M560-M599 (40)</td>
<td>Analog output register</td>
<td></td>
<td>✗</td>
</tr>
<tr>
<td>M600-M639 (40)</td>
<td>Analog input register</td>
<td></td>
<td>✗</td>
</tr>
<tr>
<td>M640-M999 (36)</td>
<td>System register</td>
<td></td>
<td>✗</td>
</tr>
</tbody>
</table>
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.
3.4 Read/Write of I/O Signals

- **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.
3.5 File Data Transmission Function

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

- **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.
3.6 Macro function

- **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.
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.

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 MOTOCOMES functions can be used in the file where "MOTOCOMES.h" is included.

**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_SAVEFILE).
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.

The MOTOCOMES installation directory contains data transmission functions (Windows DLL file type, file name: MOTOCOMES.DLL, 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
2. Visual Studio 2005 or more

*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.

```csharp
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.

```
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.
```
4.3.3 Structure

Structures used by transmission function are defined in MotoComES_CS. When use the following structures, call the initialize function "Init()" after creating the instance. Some members were not reserved the memory areas by creating the instance, but calling the initialize function, the memory areas of them are reserved. And, it is not necessary to call an initialization function in any structures other than these.

<table>
<thead>
<tr>
<th>Structure</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESAlarmData</td>
<td>ESGetAlarmHist</td>
</tr>
<tr>
<td>ESAlarmDataEx</td>
<td>ESGetAlarmHistEx</td>
</tr>
<tr>
<td>ESAlarmList</td>
<td>ESGetAlarm</td>
</tr>
<tr>
<td>ESAlarmListEx</td>
<td>ESGetAlarmEx</td>
</tr>
<tr>
<td>ESJobStatusData</td>
<td>ESGetJobStatus</td>
</tr>
<tr>
<td>ESConfigurationData</td>
<td>ESGetConfiguration</td>
</tr>
<tr>
<td>ESAxisData</td>
<td>ESGetDeviation, ESGetTorque</td>
</tr>
<tr>
<td>ESPositionData</td>
<td>ESGetPosition, ESGetPositionData, ESSetPositionData</td>
</tr>
<tr>
<td>ESBpexPositionData</td>
<td>ESGetBpexPositionData, ESSetBpexPositionData</td>
</tr>
<tr>
<td>ESMonitoringTimeData</td>
<td>ESGetMonitoringTime</td>
</tr>
<tr>
<td>ESSystemInfoData</td>
<td>ESGetSystemInfo</td>
</tr>
<tr>
<td>ESMultiByteData</td>
<td>ESGetVarDataMB, ESSetVarDataMB, ESReadIOM, ESWriteIOM</td>
</tr>
<tr>
<td>ESMultiShortData</td>
<td>ESGetVarDataMI, ESSetVarDataMI</td>
</tr>
<tr>
<td>ESMultiUShortData</td>
<td>ESReadRegisterM, ESWriteRegisterM</td>
</tr>
<tr>
<td>ESMultiLongData</td>
<td>ESGetVarDataMD, ESSetVarDataMD</td>
</tr>
<tr>
<td>ESMultiRealData</td>
<td>ESGetVarDataMR, ESSetVarDataMR</td>
</tr>
<tr>
<td>ESMultiStrData</td>
<td>ESGetStrDataM, ESSetStrDataM</td>
</tr>
<tr>
<td>ESMultiStrData2</td>
<td>ESGetStrDataM2, ESSetStrDataM2</td>
</tr>
<tr>
<td>ESMultiPositionData</td>
<td>ESGetPositionDataM, ESSetPositionDataM</td>
</tr>
<tr>
<td>ESMultiBpexPositionData</td>
<td>ESGetBpexPositionDataM, ESSetBpexPositionDataM</td>
</tr>
<tr>
<td>ESPulsePosData</td>
<td>ESPulseMove</td>
</tr>
<tr>
<td>ESCartMoveData</td>
<td>ESCartMove</td>
</tr>
</tbody>
</table>

[Example]
ESAlarmData data = new ESAlarmData();
data.Init();
4.4 Each Function Program List

- **CTestDlg::OnLoadfile ( )**

```cpp
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;
}
```

- **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;
}

- 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;
    }
}
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);

    //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;
}
MOTOCOMES.DLL is a transmission library that controls the data transmission function of the YRC1000, DX200, DX100, FS100 on a personal computer. This library is composed in the form of Microsoft Windows DLL (Dynamic Link Library).

Transmission library has the following functions.

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

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.
## 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 functions are available.

<table>
<thead>
<tr>
<th>Status Read</th>
<th>System Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESGetAlarm</td>
<td>ESRset</td>
</tr>
<tr>
<td>ESGetAlarmHist</td>
<td>ESCancel</td>
</tr>
<tr>
<td>ESGetStatus</td>
<td>ESHold</td>
</tr>
<tr>
<td>ESGetJobStatus</td>
<td>ESServo</td>
</tr>
<tr>
<td>ESGetConfiguration</td>
<td>ESHlock</td>
</tr>
<tr>
<td>ESGetPosition</td>
<td>ESCycle</td>
</tr>
<tr>
<td>ESGetDeviation</td>
<td>ESBDSP</td>
</tr>
<tr>
<td>ESGetTorque</td>
<td>ESSetVarData1</td>
</tr>
<tr>
<td>ESGetMonitoringTime</td>
<td>ESSetVarData2</td>
</tr>
<tr>
<td>ESGetSystemInfo</td>
<td>ESSetStrData</td>
</tr>
<tr>
<td>ESGetVarData1</td>
<td>ESSetStrData2</td>
</tr>
<tr>
<td>ESGetVarData2</td>
<td>ESSetPositionData</td>
</tr>
<tr>
<td>ESGetStrData</td>
<td>ESSetBpexPositionData</td>
</tr>
<tr>
<td>ESGetStrData2</td>
<td>ESSetVarDataMB</td>
</tr>
<tr>
<td>ESGetPositionData</td>
<td>ESSetVarDataMI</td>
</tr>
<tr>
<td>ESGetBpexPositionData</td>
<td>ESSetVarDataMD</td>
</tr>
<tr>
<td>ESGetVarDataMB</td>
<td>ESSetVarDataMR</td>
</tr>
<tr>
<td>ESGetVarDataMI</td>
<td>ESSetStrDataM</td>
</tr>
<tr>
<td>ESGetVarDataMD</td>
<td>ESSetStrDataM2</td>
</tr>
<tr>
<td>ESGetVarDataMR</td>
<td>ESSetPositionDataM</td>
</tr>
<tr>
<td>ESGetStrDataM</td>
<td>ESSetBpexPositionDataM</td>
</tr>
<tr>
<td>ESGetStrDataM2</td>
<td>ESSelectJob</td>
</tr>
<tr>
<td>ESGetPositionDataM</td>
<td>ESSetStartJob</td>
</tr>
<tr>
<td>ESGetBpexPositionDataM</td>
<td>ESCartMove</td>
</tr>
<tr>
<td>ESGetAlarmEx</td>
<td>ESPulseMove</td>
</tr>
<tr>
<td>ESGetAlarmHistEx</td>
<td></td>
</tr>
</tbody>
</table>
## ESGetAlarm

Reads a current error data.

**FORMAT**

```c
LONG ESGetAlarm( HANDLE handle, ESAlarmList* alarmList );
```

**ARGUMENTS**

<table>
<thead>
<tr>
<th>[in] handle</th>
<th>Target handle value</th>
</tr>
</thead>
<tbody>
<tr>
<td>[out] alarmList</td>
<td>Alarm list storage pointer</td>
</tr>
</tbody>
</table>

**ESAlarmList**

Alarm list structure

**FORMAT**

```c
#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**

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

**RETURN VALUE**

- `0` : Normal completion
- Others : Error codes
5.2 Robot Control Function

REFERENCE

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

Reads an alarm history of the specified alarm number.

**FORMAT**

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

**ARGUMENTS**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1〜100</td>
<td>Major alarm</td>
</tr>
<tr>
<td>1001〜1100</td>
<td>Minor alarm</td>
</tr>
<tr>
<td>2001〜2100</td>
<td>User alarm (system)</td>
</tr>
<tr>
<td>3001〜3100</td>
<td>User alarm (user)</td>
</tr>
<tr>
<td>4001〜4100</td>
<td>off-line alarm</td>
</tr>
</tbody>
</table>

- **[out] alarmData**  
  Alarm data storage pointer

**ESAlarmData**  
Alarm data structure

**FORMAT**

```c
#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
5.2 Robot Control Function

REFERENCE

"ESGetAlarm" "ESGetStatus" "ESGetAlarmEx" "ESGetAlarmHistEx"
5.2 Robot Control Function

■ 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

<table>
<thead>
<tr>
<th>bitValue</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>D00</td>
<td>Step</td>
</tr>
<tr>
<td>D01</td>
<td>1-cycle</td>
</tr>
<tr>
<td>D02</td>
<td>Auto operation</td>
</tr>
<tr>
<td>D03</td>
<td>Operating</td>
</tr>
<tr>
<td>D04</td>
<td>Operation at safe speed</td>
</tr>
<tr>
<td>D05</td>
<td>Teach</td>
</tr>
<tr>
<td>D06</td>
<td>Play</td>
</tr>
<tr>
<td>D07</td>
<td>Command remote</td>
</tr>
</tbody>
</table>

<status2> status2

**RETURN VALUE**

0 : Normal completion

Others : Error codes

**REFERENCE**

"ESGetAlarm" "ESGetAlarmHist" "ESGetAlarmEx" "ESGetAlarmHistEx"
5.2 Robot Control Function

- **ESGetJobStatus**

  Reads a current job information of the specified task.

**FORMAT**

```c
LONG  ESGetJobStatus( HANDLE handle, LONG taskNo, ESJobStatusData jobStatusData );
```

**ARGUMENTS**

- [in] handle  
  Target handle value
- [in] taskNo  
  Task number
- [out] jobStatusData  
  Job status data storage pointer

**ESJobStatusData**

Job status data structure

**FORMAT**

```c
#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"
5.2 Robot Control Function

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

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

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

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

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

[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

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Pulse</td>
</tr>
<tr>
<td>16</td>
<td>Coordinate (base)</td>
</tr>
</tbody>
</table>

<fig> Figure

<table>
<thead>
<tr>
<th>Figure</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<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)

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

RETURN VALUE
0 : Normal completion
0xA001 : No control group
Others : Error codes
5.2 Robot Control Function

- **ESGetDeviation**

Reads a current deviation of the specified control group.

**FORMAT**

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

**ARGUMENTS**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>handle</td>
<td>Target handle value</td>
</tr>
<tr>
<td>ctrlGrp</td>
<td>Control group</td>
</tr>
</tbody>
</table>

---

**FORMAT**

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

**ARGUMENTS**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>handle</td>
<td>Target handle value</td>
</tr>
<tr>
<td>ctrlGrp</td>
<td>Control group</td>
</tr>
</tbody>
</table>
[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)

<table>
<thead>
<tr>
<th>Array</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>axis[0]</td>
<td>1st axis Pulse</td>
</tr>
<tr>
<td>axis[1]</td>
<td>2nd axis Pulse</td>
</tr>
<tr>
<td>axis[2]</td>
<td>3rd axis Pulse</td>
</tr>
<tr>
<td>axis[3]</td>
<td>4th axis Pulse</td>
</tr>
<tr>
<td>axis[4]</td>
<td>5th axis Pulse</td>
</tr>
<tr>
<td>axis[5]</td>
<td>6th axis Pulse</td>
</tr>
<tr>
<td>axis[6]</td>
<td>7th axis Pulse</td>
</tr>
<tr>
<td>axis[7]</td>
<td>8th axis Pulse</td>
</tr>
</tbody>
</table>

RETURN VALUE

0 : Normal completion
0xA001 : No control group
Others : Error codes
5.2 Robot Control Function

**ESGetTorque**

Reads a current torque of the specified control group.

**FORMAT**

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

**ARGUMENTS**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>handle</code></td>
<td>Target handle value</td>
</tr>
<tr>
<td><code>ctrlGrp</code></td>
<td>Control group</td>
</tr>
</tbody>
</table>

**FORMAT**

```c
LONG ESGetTorque( HANDLE handle, LONG ctrlGrp, ESAxisData* torqueData );
```
5.2 Robot Control Function

[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)

<table>
<thead>
<tr>
<th>Array</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>axis[0]</td>
<td>1st axis (unit: 0.01%)</td>
</tr>
<tr>
<td>axis[1]</td>
<td>2nd axis (unit: 0.01%)</td>
</tr>
<tr>
<td>axis[2]</td>
<td>3rd axis (unit: 0.01%)</td>
</tr>
<tr>
<td>axis[3]</td>
<td>4th axis (unit: 0.01%)</td>
</tr>
<tr>
<td>axis[4]</td>
<td>5th axis (unit: 0.01%)</td>
</tr>
<tr>
<td>axis[5]</td>
<td>6th axis (unit: 0.01%)</td>
</tr>
<tr>
<td>axis[6]</td>
<td>7th axis (unit: 0.01%)</td>
</tr>
<tr>
<td>axis[7]</td>
<td>8th axis (unit: 0.01%)</td>
</tr>
</tbody>
</table>

RETURN VALUE

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

Reads a current monitoring time.

**FORMAT**

```c
LONG ESGetMonitoringTime( HANDLE handle, LONG timeType, ESMonitoringTimeData* timeData );
```

**ARGUMENTS**

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in]</td>
<td>handle</td>
</tr>
<tr>
<td>[in]</td>
<td>timeType</td>
</tr>
<tr>
<td>[out]</td>
<td>timeData</td>
</tr>
</tbody>
</table>

**RETURN VALUE**

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal completion</td>
</tr>
<tr>
<td>0xA001</td>
<td>Out of type range</td>
</tr>
<tr>
<td>0xC800</td>
<td>No Monitoring time</td>
</tr>
<tr>
<td>Others</td>
<td>Error codes</td>
</tr>
</tbody>
</table>

**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)

**ESMonitoringTimeData**

- Monitoring time data structure

**FORMAT**

```c
#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)
### ESGetSystemInfo

Read a current system information.

**FORMAT**

```c
LONG ESGetSystemInfo( HANDLE handle, LONG systemType, ESSystemInfoData* infoData );
```

**ARGUMENTS**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>11 to 18</td>
<td>Model information ( R1 to R8 )</td>
</tr>
<tr>
<td>21 to 44</td>
<td>Model information ( S1 to S24 )</td>
</tr>
<tr>
<td>101 to 108</td>
<td>Application information ( R1 to R8 )</td>
</tr>
</tbody>
</table>

| [out] infoData | System information data storage pointer |

**ESSystemInfoData**

System information data structure

```c
#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];
} ESSystemInfoData;
```

**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**

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

**ARGUMENTS**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] handle</td>
<td>Target handle value</td>
</tr>
<tr>
<td>[in] type</td>
<td>Variable data type</td>
</tr>
<tr>
<td>[in] number</td>
<td>Variable number</td>
</tr>
<tr>
<td>[out] data</td>
<td>Variable data storage pointer</td>
</tr>
</tbody>
</table>

**RETURN VALUE**

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal completion</td>
</tr>
<tr>
<td>0xA001</td>
<td>Out of variable number range</td>
</tr>
<tr>
<td>Others</td>
<td>Error codes</td>
</tr>
</tbody>
</table>

**REMARKS**

Restrictions

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

<table>
<thead>
<tr>
<th>RS023</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ESGetVarData1</td>
</tr>
<tr>
<td>1</td>
<td>ESGetVarData2</td>
</tr>
</tbody>
</table>

**REFERENCE**

- "ESGetVarData2"
- "ESGetVarDataMB"
- "ESGetVarDataMI"
- "ESGetVarDataMD"
- "ESGetVarDataMR"
- "ESSetVarData1"
- "ESSetVarData2"
- "ESSetVarDataMB"
- "ESSetVarDataMI"
- "ESSetVarDataMD"
- "ESSetVarDataMR"
5.2 Robot Control Function

ESGetVarData2

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

**FORMAT**

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

**ARGUMENTS**

- **[in] handle** Target handle value
- **[in] type** Variable data type
  
<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Byte (B)</td>
</tr>
<tr>
<td>2</td>
<td>Integer (I)</td>
</tr>
<tr>
<td>3</td>
<td>Double (D)</td>
</tr>
<tr>
<td>4</td>
<td>Real (R)</td>
</tr>
</tbody>
</table>

- **[in] number** Variable number
  
<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 to 100</td>
<td>RS022=0</td>
</tr>
<tr>
<td>0 to 99</td>
<td>RS022=1</td>
</tr>
</tbody>
</table>

- **[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.

**REFERENCE**

"ESGetVarData1"  "ESGetVarDataMB"  "ESGetVarDataMI"  "ESGetVarDataMD"
"ESGetVarDataMR"  "ESSetVarData1"  "ESSetVarData2"  "ESSetVarDataMB"
"ESSetVarDataMI"  "ESSetVarDataMD"  "ESSetVarDataMR"
5.2 Robot Control Function

- **ESGetStrData**
  
  Reads a string variable. (DX100, FS100)

**FORMAT**

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

**ARGUMENTS**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 to 100</td>
<td>RS022=0</td>
</tr>
<tr>
<td>0 to 99</td>
<td>RS022=1</td>
</tr>
</tbody>
</table>

**RETURN VALUE**

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

**REMARKS**

Restrictions
This function is for DX100 and FS100. If the controller is DX200 or YRC1000, Please use "ESGetStrData2".

**REFERENCE**

"ESGetStrData2" "ESGetStrDataM" "ESGetStrDataM2" "ESSetStrData"
"ESSetStrData2" "ESSetStrDataM" "ESSetStrDataM2"
5.2 Robot Control Function

- **ESGetStrData2**
  Reads a string variable. (YRC1000, DX200)

**FORMAT**

```c
LONG  ESGetStrData2(HANDLE handle, LONG number, CHAR* cp );
```

**ARGUMENTS**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 to 100</td>
<td>RS022=0</td>
</tr>
<tr>
<td>0 to 99</td>
<td>RS022=1</td>
</tr>
</tbody>
</table>

**RETURN VALUE**

- 0 : Normal completion
- 0xA001 : Out of variable number range
- 0x9200 : Disable function (When used for DX100, FS100)
- Others : Error codes

**REMARKS**

Restrictions
This function is for DX200 and YRC1000. If the controller is DX100 or FS100, Please use "ESGetStrData".

<table>
<thead>
<tr>
<th>Controller</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DX100, FS100</td>
<td>ESGetStrData</td>
</tr>
<tr>
<td>DX200, YRC1000</td>
<td>ESGetStrData2</td>
</tr>
</tbody>
</table>

**REFERENCE**

"ESGetStrData" "ESGetStrDataM" "ESGetStrDataM2" "ESSetStrData"
"ESSetStrData2" "ESSetStrDataM" "ESSetStrDataM2"
5.2 Robot Control Function

**ESGetPositionData**

Reads a robot position variable.

**FORMAT**

```c
LONG ESGetPositionData( HANDLE handle, LONG number, ESPositionData* positionData );
```

**ARGUMENTS**

- **[in] handle** Target handle value
- **[in] number** Variable number
- **[out] positionData** Robot position data storage pointer

**ESPositionData**

Robot position data structure

**FORMAT**

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

**MEMBER**

- `<dataType>` Data type

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Pulse</td>
</tr>
<tr>
<td>16</td>
<td>Coordinate (Base-coordinate)</td>
</tr>
<tr>
<td>17</td>
<td>Coordinate (Robot-coordinate)</td>
</tr>
<tr>
<td>18</td>
<td>Coordinate (Tool-coordinate)</td>
</tr>
<tr>
<td>19</td>
<td>Coordinate (User-coordinate)</td>
</tr>
<tr>
<td>20</td>
<td>Coordinate (Master tool-coordinate)</td>
</tr>
</tbody>
</table>
### 5.2 Robot Control Function

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

**RETURN VALUE**

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

**REFERENCE**

- "ESGetStrDataM2"
- "ESSetPositionData"
- "ESSetPositionDataM"
5.2 Robot Control Function

- **ESGetBpexPositionData**
  Reads a base/external-axis position variable.

**FORMAT**

```c
LONG ESGetBpexPositionData( HANDLE handle, LONG type, LONG number,
    ESBpexPositionData* positionData );
```

**ARGUMENTS**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Base</td>
</tr>
<tr>
<td>2</td>
<td>External axis</td>
</tr>
</tbody>
</table>

- **[in] number**
  Variable number

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 to 128</td>
<td>RS022=0</td>
</tr>
<tr>
<td>0 to 127</td>
<td>RS022=1</td>
</tr>
</tbody>
</table>

**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 be added 1.

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

**ESBpexPositionData**

Base/External-axis position data structure

**FORMAT**

```c
typedef struct
{
    LONG dataType;
    ESBAxisData axesData;
} ESBpexPositionData;
```

**MEMBER**

- `<dataType>` Data type

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Pulse</td>
</tr>
<tr>
<td>16</td>
<td>Coordinate (Base only)</td>
</tr>
</tbody>
</table>
<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)

<table>
<thead>
<tr>
<th>Array</th>
<th>Pulse type</th>
<th>Coordinate type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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**

```c
LONG ESGetVarDataMB( HANDLE handle, LONG varno, LONG number, ESMultiByteData* varData );
```

**ARGUMENTS**

- `[in] handle` Target handle value
- `[in] varno` Variable number
- `[in] number` Number of variables
- `[out] varData` Multi-data storage pointer

**ESMultiByteData**

Multi-data structure (1byte)

**FORMAT**

```c
#define Length_of_Multi_1 (474)
typedef struct
{
    CHAR    data[Length_of_Multi_1];
} ESMultiByteData;
```

**MEMBER**

```c
<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"

---

**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 be added 1.

**NOTE**

It needs that Number of variables is even.

---

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

---

It needs that Number of variables is even.
"ESGetVarDataMR" "ESSetVarData1" "ESSetVarData2" "ESSetVarDataMB"
"ESSetVarDataMI" "ESSetVarDataMD" "ESSetVarDataMR"
5.2 Robot Control Function

- **ESGetVarDataMI**

Reads variables (I) continuously from the specified number.

**FORMAT**

LONG ESGetVarDataMI( HANDLE handle, LONG varno, LONG number, ESMultiShortData* varData );

**ARGUMENTS**

- [in] handle  
  Target handle value
- [in] varno  
  Variable number
- [in] number  
  Number of variables
- [out] varData  
  Multi-data storage pointer

ESMultiShortData
Multi-data structure (2byte)

```c
#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"
"ESSetVarDataML"  "ESSetVarDataMD"  "ESSetVarDataMR"

**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 be added 1.

Value | Explanation
--- | ---
1 to 100 | RS022=0
0 to 99 | RS022=1
5.2 Robot Control Function

- **ESGetVarDataMD**
  Reads variables (D) continuously from the specified number.

**FORMAT**

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

**ARGUMENTS**

- **[in] handle** Target handle value
- **[in] varno** Variable number
- **[in] number** Number of variables
- **[out] varData** Multi-data storage pointer

**ESMultiLongData**
Multi-data structure (4byte)

```
#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"

---

**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 be added 1.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 to 100</td>
<td>RS022=0</td>
</tr>
<tr>
<td>0 to 99</td>
<td>RS022=1</td>
</tr>
</tbody>
</table>
### ESGetVarDataMR

**Function**

Reads variables (R) continuously from the specified number.

**Format**

```c
LONG ESGetVarDataMR( HANDLE handle, LONG varno, LONG number, ESMultiRealData* varData );
```

**Arguments**

- `[in] handle`  Target handle value
- `[in] varno`  Variable number
- `[in] number` Number of variables
- `[out] varData` Multi-data storage pointer

**ESMultiRealData**

Multi-data structure (Real number)

**Format**

```c
#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"

**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 be added 1.
ESGetStrDataM

Reads string variables continuously from the specified number. (DX100, FS100)

FORMAT

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

ARGUMENTS

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

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

Check the “RS022” parameter before using this function. Set the value as above. If the "RS022" parameter is zero, the variable number needs to be added 1.
5.2 Robot Control Function

REMARKS

Restrictions
This function is for DX100 and FS100. If the controller is DX200 or YRC1000, Please use "ESGetStrDataM2".

<table>
<thead>
<tr>
<th>Controller</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DX100, FS100</td>
<td>ESGetStrDataM</td>
</tr>
<tr>
<td>DX200, YRC1000</td>
<td>ESGetStrDataM2</td>
</tr>
</tbody>
</table>

REFERENCE

"ESGetStrData" "ESGetStrDataM" "ESGetStrDataM2" "ESSetStrData"
"ESSetStrData2" "ESSetStrDataM" "ESSetStrDataM2"
ESGetStrDataM2

Reads string variables continuously from the specified number. (YRC1000, DX200)

**FORMAT**

LONG ESGetStrDataM2(HANDLE handle, LONG varno, LONG number, ESMultiStrData2* varData);

**ARGUMENTS**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] handle</td>
<td>Target handle value</td>
</tr>
<tr>
<td>[in] varno</td>
<td>Variable number</td>
</tr>
<tr>
<td>[in] number</td>
<td>Number of variables</td>
</tr>
<tr>
<td>[out] varData</td>
<td>Multi-data storage pointer</td>
</tr>
</tbody>
</table>

**RETURN VALUE**

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal completion</td>
</tr>
<tr>
<td>0xA001</td>
<td>Out of variable number range</td>
</tr>
<tr>
<td>0xB004</td>
<td>Including out of variable number range</td>
</tr>
<tr>
<td>0x9200</td>
<td>Disable function (When used for DX100, FS100)</td>
</tr>
<tr>
<td>Others</td>
<td>Error codes</td>
</tr>
</tbody>
</table>

**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 be added 1.

**MEMBER**

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>String data (Max number = 14/Max length = 32)</td>
</tr>
</tbody>
</table>

**FORMAT**

#define Length_of_Multi_Str2 (14)
#define Length_of_String2 (32)

typedef struct
{
  CHAR data[Length_of_Multi_Str2][Length_of_String2+1];
} ESMultiStrData2;
5.2 Robot Control Function

REMARKS

Restrictions
This function is for DX200 and YRC1000. If the controller is DX100 or FS100, Please use "ESGetStrDataM".

<table>
<thead>
<tr>
<th>Controller</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DX100, FS100</td>
<td>ESGetStrDataM</td>
</tr>
<tr>
<td>DX200, YRC1000</td>
<td>ESGetStrDataM2</td>
</tr>
</tbody>
</table>

REFERENCE

"ESGetStrData" "ESGetStrDataM" "ESGetStrDataM2" "ESSetStrData"
"ESSetStrData2" "ESSetStrDataM" "ESSetStrDataM2"
ESGetPositionDataM

Reads robot position variables continuously from the specified number.

**FORMAT**

```c
LONG ESGetPositionDataM( HANDLE handle, LONG varno, LONG number,
                      ESMultiPositionData* positionData );
```

**ARGUMENTS**

- `[in]` handle
  - Target handle value
- `[in]` varno
  - Variable number
- `[in]` number
  - Number of variables
- `[out]` positionData
  - Multi-data storage pointer

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-128</td>
<td>RS022=0</td>
</tr>
<tr>
<td>0-127</td>
<td>RS022=1</td>
</tr>
</tbody>
</table>

**Value Explanation**

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

**ESMultiPositionData**

Multi-data structure (Robot position)

```c
#define Length_of_Multi_Pos (9)

typedef struct
{
    ESPositionData    data[Length_of_Multi_Pos];
} ESMultiPositionData;
```
5.2 Robot Control Function

**MEMBER** `<data[Length_of_Multi_Pos]>` Robot position data (Max number = 9)

**ESPositionData**

Robot position structure

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

**MEMBER** : `<dataType>` Data type

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Pulse</td>
</tr>
<tr>
<td>16</td>
<td>Coordinate (Base-coordinate)</td>
</tr>
<tr>
<td>17</td>
<td>Coordinate (Robot-coordinate)</td>
</tr>
<tr>
<td>18</td>
<td>Coordinate (Tool-coordinate)</td>
</tr>
<tr>
<td>19</td>
<td>Coordinate (User-coordinate)</td>
</tr>
<tr>
<td>20</td>
<td>Coordinate (Master tool-coordinate)</td>
</tr>
</tbody>
</table>

**<fig>** figure

**<toolNo>** Tool number

**<userFrameNo>** User frame number

**<exFig>** extended figure

**<axesData>** Axes data

**ESAxisData**

Axis data structure

**FORMAT** : 
```c
#define Number_of_Axis (8)
typedef struct
{
    DOUBLE    axis[Number_of_Axis];
} ESAxisData;
```
5.2 Robot Control Function

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

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

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"
5.2  Robot Control Function

**ESGetBpexPositionDataM**

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

**FORMAT**

```c
LONG  ESGetBpexPositionDataM( HANDLE handle, LONG type, LONG varno, LONG number,
ESMultiBpexPositionData* positionData );
```

**ARGUMENTS**

<table>
<thead>
<tr>
<th>[in] handle</th>
<th>Target handle value</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] type</td>
<td>Variable data type</td>
</tr>
<tr>
<td>[in] varno</td>
<td>Variable number</td>
</tr>
<tr>
<td>[in] number</td>
<td>Number of variables</td>
</tr>
<tr>
<td>[out] positionData</td>
<td>Multi-data storage pointer</td>
</tr>
</tbody>
</table>

**Value** | **Explanation**
--- | ---
1 | Base
2 | External axis

**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 be added 1.

**Value** | **Explanation**
--- | ---
1 to 128 | RS022=0
0 to 127  | RS022=1

```
define Length_of_Multi_Bpex (13)
typedef struct
{
    ESBpexPositionData   data[Length_of_Multi_Bpex];
} ESMultiBpexPositionData;
```

HW1483705  80/156
5.2 Robot Control Function

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Pulse</td>
</tr>
<tr>
<td>16</td>
<td>Coordinate (Base only)</td>
</tr>
</tbody>
</table>

<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)

<table>
<thead>
<tr>
<th>Array</th>
<th>Pulse type</th>
<th>Coordinate type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

RETURN VALUE
0      : Normal completion
0xA01  : Out of variable number range
0xB001 : Including no data
0xB004 : Including out of variable number range
Others : Error codes

REFERENCE
"ESGetBpexPositionData" "ESSetBpexPositionData" "ESSetBpexPositionDataM"
5.2 Robot Control Function

- **ESGetAlarmEx**

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

**FORMAT**

```c
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**

```c
#define Length_of_AlarmList (4)

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

**MEMBER**

```c
<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**

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

**MEMBER**

```c
<alarmData> Alarm data
```

**ESAlarmData**

Alarm data structure

**FORMAT**

```c
#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
<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_StrData+1];
  CHAR alarmHighlightData
  [Length_of_Subcode_StrData+1];
} ESSubcodeData;

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

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

<alarmHighLightData
[Length_of_Subcode_StrData+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"
5.2 Robot Control Function

- ESGetAlarmHistEx

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

**FORMAT**

```c
LONG  ESGetAlarmHistEx( HANDLE handle, LONG alarmHistNo, ESAAlarmDataEx* alarmData );
```

**ARGUMENTS**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1〜100</td>
<td>Major alarm</td>
</tr>
<tr>
<td>1001〜1100</td>
<td>Minor alarm</td>
</tr>
<tr>
<td>2001〜2100</td>
<td>User alarm ( system )</td>
</tr>
<tr>
<td>3001〜3100</td>
<td>User alarm ( user )</td>
</tr>
<tr>
<td>4001〜4100</td>
<td>off-line alarm</td>
</tr>
</tbody>
</table>

- **[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**  
```c
typedef struct
{
    ESAAlarmData  alarmData;
    ESSubcodeData  subcodeData;
} ESAAlarmDataEx;
```

**MEMBER**  
`<alarmData>`  Alarm data

**ESAlarmData**  Alarm data structure

**FORMAT**  
```c
#define Length_of_Time (16)
#define Length_of_Name (32)
```

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

ESSSubcodeData  
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"
5.2 Robot Control Function

ESCcancel
Cancels error.

**FORMAT**
LONG ESCancel( HANDLE handle );

**ARGUMENTS**

[in] handle Target handle value

**RETURN VALUE**

0 : Normal completion
Others : Error codes

**REFERENCE**

"ESReset"
5.2 Robot Control Function

- **ESHold**
  Sets hold on/off.

**FORMAT**

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

**ARGUMENTS**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Hold ON</td>
</tr>
<tr>
<td>2</td>
<td>Hold OFF</td>
</tr>
</tbody>
</table>

**RETURN VALUE**

- `0` : Normal completion
- Others : Error codes

**REFERENCE**

"ESServo" "ESHlock"
ESServo

Sets servo on/off.

**FORMAT**

LONG ESServo( HANDLE handle, LONG onOff );

**ARGUMENTS**

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Servo ON</td>
</tr>
<tr>
<td>2</td>
<td>Servo OFF</td>
</tr>
</tbody>
</table>

**RETURN VALUE**

0 : Normal completion  
Others : Error codes

**REFERENCE**

"ESHold"  "ESHlock"
5.2 Robot Control Function

- ESHlock

Sets interlock on/off

**FORMAT**

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

**ARGUMENTS**

<table>
<thead>
<tr>
<th>[in] handle</th>
<th>Target handle value</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] onOff</td>
<td>Interlock status</td>
</tr>
</tbody>
</table>

**RETURN VALUE**

- 0 : Normal completion
- Others : Error codes

**REFERENCE**

"ESHold" "ESServo"

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Hlock ON</td>
</tr>
<tr>
<td>2</td>
<td>Hlock OFF</td>
</tr>
</tbody>
</table>
5.2 Robot Control Function

- ESCycle

Sets cycle mode.

**FORMAT**

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

**ARGUMENTS**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Step</td>
</tr>
<tr>
<td>2</td>
<td>Cycle</td>
</tr>
<tr>
<td>3</td>
<td>Auto operation</td>
</tr>
</tbody>
</table>

**RETURN VALUE**

- 0 : Normal completion
- Others : Error codes
5.2 Robot Control Function

- **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

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Byte (B)</td>
</tr>
<tr>
<td>2</td>
<td>Integer (I)</td>
</tr>
<tr>
<td>3</td>
<td>Double (D)</td>
</tr>
<tr>
<td>4</td>
<td>Real (R)</td>
</tr>
</tbody>
</table>

- **[in] number**: Variable number

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 to 100</td>
<td>RS022=0</td>
</tr>
<tr>
<td>0 to 99</td>
<td>RS022=1</td>
</tr>
</tbody>
</table>

**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 be added 1.

- **[in] data**: Variable number

**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.

<table>
<thead>
<tr>
<th>RS023</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ESSetVarData1</td>
</tr>
<tr>
<td>1</td>
<td>ESSetVarData2</td>
</tr>
</tbody>
</table>
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**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] handle</td>
<td>Target handle value</td>
</tr>
<tr>
<td>[in] type</td>
<td>Variable data type</td>
</tr>
<tr>
<td>[in] number</td>
<td>Variable number</td>
</tr>
</tbody>
</table>

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

**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.

**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 be added 1.

**RETURN VALUE**

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 to 100</td>
<td>RS022=0</td>
</tr>
<tr>
<td>0 to 99</td>
<td>RS022=1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RS023</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ESSetVarData1</td>
</tr>
<tr>
<td>1</td>
<td>ESSetVarData2</td>
</tr>
</tbody>
</table>
REFERENCE

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

Sets a string variable. (DX100, FS100)

**FORMAT**

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

**ARGUMENTS**

<table>
<thead>
<tr>
<th>[in] handle</th>
<th>Target handle value</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] number</td>
<td>Variable number</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 to 100</td>
<td>RS022=0</td>
</tr>
<tr>
<td>0 to 99</td>
<td>RS022=1</td>
</tr>
</tbody>
</table>

**RETURN VALUE**

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

**REMARKS**

Restrictions
This function is for DX100 and FS100. If the controller is DX200 or YRC1000, Please use "ESSetStrData2".

**REFERENCE**

"ESGetStrData" "ESGetStrData2" "ESGetStrDataM" "ESGetStrDataM2" "ESSetStrData2" "ESSetStrDataM" "ESSetStrDataM2"
ESSetStrData2

Sets a string variable. (YRC1000, DX200)

**FORMAT**

LONG ESSetStrData2( HANDLE handle, LONG number, CHAR* cp );

**ARGUMENTS**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 to 100</td>
<td>RS022=0</td>
</tr>
<tr>
<td>0 to 99</td>
<td>RS022=1</td>
</tr>
</tbody>
</table>

**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 be added 1.

| [in] cp | String variable data storage pointer (Max size = 32) |

**RETURN VALUE**

0 : Normal completion
0xA001 : Out of variable number range
0x9200 : Disable function (When used for DX100, FS100)
Others : Error codes

**REMARKS**

Restrictions
This function is for DX200 and YRC1000. If the controller is DX100 or FS100, Please use "ESSetStrData".

<table>
<thead>
<tr>
<th>Controller</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DX100, FS100</td>
<td>ESSetStrData</td>
</tr>
<tr>
<td>YRC1000, DX200</td>
<td>ESSetStrData2</td>
</tr>
</tbody>
</table>

**REFERENCE**

"ESGetStrData" "ESGetStrData2" "ESGetStrDataM" "ESGetStrDataM2"
"ESSetStrData" "ESSetStrDataM" "ESSetStrDataM2"
**ESSetPositionData**

Sets a robot position variable.

**FORMAT**

```c
LONG ESSetPositionData( HANDLE handle, LONG number, ESPositionData positionData );
```

**ARGUMENTS**

- **[in] handle**  
  Target handle value
- **[in] number**  
  Variable number
- **[in] positionData**  
  Robot position data structure

**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 be added 1.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 to 128</td>
<td>RS022=0</td>
</tr>
<tr>
<td>0 to 127</td>
<td>RS022=1</td>
</tr>
</tbody>
</table>

**ESPositionData**

Robot position data structure

**FORMAT**

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

**MEMBER**

- `<dataType>`  
  Data type

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Pulse</td>
</tr>
<tr>
<td>16</td>
<td>Coordinate (Base-coordinate)</td>
</tr>
<tr>
<td>17</td>
<td>Coordinate (Robot-coordinate)</td>
</tr>
<tr>
<td>18</td>
<td>Coordinate (Tool-coordinate)</td>
</tr>
<tr>
<td>19</td>
<td>Coordinate (User-coordinate)</td>
</tr>
<tr>
<td>20</td>
<td>Coordinate (Master tool-coordinate)</td>
</tr>
</tbody>
</table>
MEMBER <fig>Figure

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

<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];
} EASAxisData;

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

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

REFERENCE
"ESGetPositionData" "ESGetStrDataM2" "ESSetPositionDataM"

HW1483705 100/156
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

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Base</td>
</tr>
<tr>
<td>2</td>
<td>External axis</td>
</tr>
</tbody>
</table>

[in] number Variable number

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 to 128</td>
<td>RS022=0</td>
</tr>
<tr>
<td>0 to 127</td>
<td>RS022=1</td>
</tr>
</tbody>
</table>

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 be 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

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Pulse</td>
</tr>
<tr>
<td>16</td>
<td>Coordinate (base)</td>
</tr>
</tbody>
</table>
<axesData> Axes data
ESAxisData
Axis data structure

FORMAT : #define Number_of_Axis (8)

typedef struct
{
    DOUBLE axis[Number_of_Axis];
} EAxisData;

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

<table>
<thead>
<tr>
<th>Array</th>
<th>Pulse type</th>
<th>Coordinate type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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**

```c
LONG ESSetVarDataMB( HANDLE handle, LONG varno, LONG number, ESMultiByteData varData );
```

**ARGUMENTS**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 to 100</td>
<td>RS022=0</td>
</tr>
<tr>
<td>0 to 99</td>
<td>RS022=1</td>
</tr>
</tbody>
</table>

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

---

### ESMultiShortData

Multi-data structure (2byte)

```
# 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"

---

**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 be added 1.

---

1 to 100 RS022=0
0 to 99 RS022=1
**ESSetVarDataMD**

Sets variables (D) continuously from the specified number.

**FORMAT**

```c
LONG ESSetVarDataMD( HANDLE handle, LONG varno, LONG number, ESMultiLongData varData );
```

**ARGUMENTS**

- `[in] handle` Target handle value
- `[in] varno` Variable number
- `[in] number` Number of variables
- `[in] varData` Multi-data structure

**RETURN VALUE**

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

**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 be added 1.

**REFERENCE**

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

Sets variables (R) continuously from the specified number.

**FORMAT**

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

**ARGUMENTS**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] handle</td>
<td>Target handle value</td>
</tr>
<tr>
<td>[in] varno</td>
<td>Variable number</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 to 100</td>
<td>RS022=0</td>
</tr>
<tr>
<td>0 to 99</td>
<td>RS022=1</td>
</tr>
</tbody>
</table>

**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 be added 1.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] number</td>
<td>Number of variables</td>
</tr>
<tr>
<td>[in] varData</td>
<td>Multi-data structure</td>
</tr>
</tbody>
</table>

ESMultiRealData

Multi-data structure (Real number)

```c
#define Length_of_Multi_4 (118)

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

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;data[Length_of_Multi_4]&gt;</td>
<td>Data (Real number) (Max number = 118)</td>
</tr>
</tbody>
</table>

**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" "ESSetVarDataMD"
ESSetStrDataM

Sets string variables continuously from the specified number. (DX100, FS100)

**FORMAT**

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

**ARGUMENTS**

- **[in] handle**
  - Target handle value
- **[in] varno**
  - Variable number
- **[in] number**
  - Number of variables
- **[in] varData**
  - Multi-data structure

**VALUE**

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 to 100</td>
<td>RS022=0</td>
</tr>
<tr>
<td>0 to 99</td>
<td>RS022=1</td>
</tr>
</tbody>
</table>

**MEMBER**

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

**RETURN VALUE**

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

---

**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 be added 1.
REMARKS

Restrictions
This function is for DX100 and FS100. If the controller is DX200 or YRC1000, Please use "ESSetStrDataM2".

<table>
<thead>
<tr>
<th>Controller</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DX100, FS100</td>
<td>ESSetStrDataM</td>
</tr>
<tr>
<td>YRC1000, DX200</td>
<td>ESSetStrDataM2</td>
</tr>
</tbody>
</table>

REFERENCE

"ESGetStrData" "ESGetStrData2" "ESGetStrDataM" "ESGetStrDataM2"
"ESSetStrData" "ESSetStrData2" "ESSetStrDataM2"
**ESSetStrDataM2**

Sets string variables continuously from the specified number. (YRC1000, DX200)

**FORMAT**

`LONG ESSetStrDataM2(HANDLE handle, LONG varno, LONG number, ESMultiStrData2 varData);`

**ARGUMENTS**

- **[in]** `handle` : Target handle value
- **[in]** `varno` : Variable number
- **[in]** `number` : Number of variables
- **[in]** `varData` : Multi-data structure

**ESMultiStrData2**

Multi-data structure (String)

**FORMAT**

```c
#define Length_of_Multi_Str2 (14)
#define Length_of_String2 (32)

typedef struct
{
    CHAR    data[Length_of_Multi_Str2][Length_of_String2+1];
} ESMultiStrData2;
```

**MEMBER**

- `<data[Length_of_Multi_Str2][Length_of_String2+1]>` : String data
  
  (Max number = 14/
  Max length = 32)

**RETURN VALUE**

- **0** : Normal completion
- **0xA001** : Out of variable number range
- **0xB004** : Including out of variable number range
- **0x9200** : Disable function (When used for DX100, FS100)
- **Others** : Error codes

---

**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 be added 1.

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 to 100</td>
<td>RS022=0</td>
</tr>
<tr>
<td>0 to 99</td>
<td>RS022=1</td>
</tr>
</tbody>
</table>

Value Explanation

1 to 100 RS022=0
0 to 99 RS022=1

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

Restrictions
This function is for DX200 and YRC1000. If the controller is DX100 or FS100, Please use "ESSetStrDataM".

<table>
<thead>
<tr>
<th>Controller</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DX100, FS100</td>
<td>ESSetStrDataM</td>
</tr>
<tr>
<td>YRC1000, DX200</td>
<td>ESSetStrDataM2</td>
</tr>
</tbody>
</table>

REFERENCE

"ESGetStrData" "ESGetStrData2" "ESGetStrDataM" "ESGetStrDataM2"
"ESSetStrData" "ESSetStrData2" "ESSetStrDataM"
**ESSetPositionDataM**

Sets robot position variables continuously from the specified number.

**FORMAT**

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

**ARGUMENTS**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>handle</td>
<td>Target handle value</td>
</tr>
<tr>
<td>varno</td>
<td>Variable number</td>
</tr>
<tr>
<td>number</td>
<td>Number of variables</td>
</tr>
<tr>
<td>positionData</td>
<td>Multi-data structure</td>
</tr>
</tbody>
</table>

**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 be added 1.

**ESMultiPositionData**

Multi-data structure (Robot position)

**FORMAT**

```c
#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**

```c
typedef struct
{
    LONG  dataType;
    LONG  fig;
    LONG  toolNo;
    LONG  userFrameNo;
    LONG  exFig;
    ESAxisData axesData;
} ESPositionData;
```
MEMBER : <dataType>  Data type

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Pulse</td>
</tr>
<tr>
<td>16</td>
<td>Coordinate (Base-coordinate)</td>
</tr>
<tr>
<td>17</td>
<td>Coordinate (Robot-coordinate)</td>
</tr>
<tr>
<td>18</td>
<td>Coordinate (Tool-coordinate)</td>
</tr>
<tr>
<td>19</td>
<td>Coordinate (User-coordinate)</td>
</tr>
<tr>
<td>20</td>
<td>Coordinate (Master tool-coordinate)</td>
</tr>
</tbody>
</table>

<fig>Figure

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

<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)
RETURN VALUE

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

REFERENCE

"ESGetPositionData" "ESSetPositionData" "ESGetStrDataM2"
ESSetBpexPositionDataM

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

**FORMAT**

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

**ARGUMENTS**

- `[in] handle` Target handle value
- `[in] type` variable data type
  
<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Base</td>
</tr>
<tr>
<td>2</td>
<td>External axis</td>
</tr>
</tbody>
</table>

- `[in] varno` Variable number
  
<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 to 128</td>
<td>RS022=0</td>
</tr>
<tr>
<td>0 to 127</td>
<td>RS022=1</td>
</tr>
</tbody>
</table>

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

**ESMultiBpexPositionData**

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

**FORMAT**

```c
#define Length_of_Multi_Bpex (13)

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

**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 be added 1.
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

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Pulse</td>
</tr>
<tr>
<td>16</td>
<td>Coordinate (Base only)</td>
</tr>
</tbody>
</table>

<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)

<table>
<thead>
<tr>
<th>Array</th>
<th>Pulse type</th>
<th>Coordinate type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

RETURN VALUE

0     : Normal completion
0xA01 : Out of variable number range
0xB04 : Including out of variable number range
Others : Error codes

REFERENCE

"ESGetBpexPositionData"  "ESSetBpexPositionData"  "ESGetBpexPositionDataM"
**ESSSelectJob**

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

**FORMAT**

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

**ARGUMENTS**

- **[in]** `handle` : Target handle value
- **[in]** `jobType` : Job type

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Current job</td>
</tr>
<tr>
<td>10</td>
<td>Master job ( Master )</td>
</tr>
<tr>
<td>11</td>
<td>Master job ( Sub 1 )</td>
</tr>
<tr>
<td>12</td>
<td>Master job ( Sub 2 )</td>
</tr>
<tr>
<td>13</td>
<td>Master job ( Sub 3 )</td>
</tr>
<tr>
<td>14</td>
<td>Master job ( Sub 4 )</td>
</tr>
<tr>
<td>15</td>
<td>Master job ( Sub 5 )</td>
</tr>
<tr>
<td>16</td>
<td>Master job ( Sub 6 )</td>
</tr>
<tr>
<td>17</td>
<td>Master job ( Sub 7 )</td>
</tr>
<tr>
<td>18</td>
<td>Master job ( Sub 8 )</td>
</tr>
<tr>
<td>19</td>
<td>Master job ( Sub 9 )</td>
</tr>
<tr>
<td>20</td>
<td>Master job ( Sub 10 )</td>
</tr>
<tr>
<td>21</td>
<td>Master job ( Sub 11 )</td>
</tr>
<tr>
<td>22</td>
<td>Master job ( Sub 12 )</td>
</tr>
<tr>
<td>23</td>
<td>Master job ( Sub 13 )</td>
</tr>
<tr>
<td>24</td>
<td>Master job ( Sub 14 )</td>
</tr>
<tr>
<td>25</td>
<td>Master job ( Sub 15 )</td>
</tr>
</tbody>
</table>

- **[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
- **[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

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>øç ēİĵIJçĨĩĺĶijļĻĬçķĶĺİĻİĶĵçĶķĬĹĨĻİĶĵçïĔĖģđðç</td>
<td></td>
</tr>
<tr>
<td>úç ĚĻĹĨİĮįĻçĨĩĺĶijļĻĬçķĶĺİĻİĶĵçïĐĔĔģēð</td>
<td></td>
</tr>
<tr>
<td>úç ĚĻĹĨİĮįĻçİĵĪĹĬĴĬĵĻçĽĨijļĬçĶķĬĹĨĻİĶĵçïĐĔĔģēɺĐĔĔģðç</td>
<td></td>
</tr>
</tbody>
</table>

- `øç ēİĵIJçĨĩĺĶijļĻĬçķĶĺİĻİĶĵçĔĔģđðç`
- `úç ĚĻĹĨİĮįĻçİţĪĹĬĴĬţĻçĽĨijļĬçĶķĬĹĨĻİĶţçïĐĔĔģēɺĐĔĔģδç`
- `úç ĚĻĹĨİĮįĻçĨĩĺĶijļĻĬçķĶĺİĻİĶţçïĐĔĔģēɺĐĔĔģδç`

- `÷ç ìçïēİĵIJçĶķĬĹĨĻİĶţçĔĔģđðç`
- `øç ğçïĊĨĹĻĬĺİĩţçĶķĬĹĨĻİĶţçĔĔģēɺĐĔĔģδç`
- `ùç ğęçïĊĨĹĻĬĺİĩţçĶķĬĹĨĻİĶţçĔĔģēɺĐĔĔģδç`
<speed> Specifying the speed

<table>
<thead>
<tr>
<th>Unit</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>%</td>
<td>一部単位（デジタル式）</td>
</tr>
<tr>
<td>デジタル</td>
<td>一部単位（デジタル式）</td>
</tr>
<tr>
<td>デジタル</td>
<td>一部単位（デジタル式）</td>
</tr>
</tbody>
</table>

<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

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>Base-coordinate</td>
</tr>
<tr>
<td>17</td>
<td>Robot-coordinate</td>
</tr>
<tr>
<td>18</td>
<td>User-coordinate</td>
</tr>
<tr>
<td>19</td>
<td>Tool-coordinate</td>
</tr>
</tbody>
</table>

<fig> Figure

<toolNo> Tool number

<userFrameNo> User frame number

<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)

<table>
<thead>
<tr>
<th>Array</th>
<th>Coordinate type</th>
</tr>
</thead>
<tbody>
<tr>
<td>axis[0]</td>
<td>X-coordinate (unit: mm)</td>
</tr>
<tr>
<td>axis[1]</td>
<td>Y-coordinate (unit: mm)</td>
</tr>
<tr>
<td>axis[2]</td>
<td>Z-coordinate (unit: mm)</td>
</tr>
<tr>
<td>axis[3]</td>
<td>Rx angle (unit: deg)</td>
</tr>
<tr>
<td>axis[4]</td>
<td>Ry angle (unit: deg)</td>
</tr>
<tr>
<td>axis[5]</td>
<td>Rz angle (unit: deg)</td>
</tr>
<tr>
<td>axis[6]</td>
<td>Re angle (unit: deg)</td>
</tr>
<tr>
<td>axis[7]</td>
<td></td>
</tr>
</tbody>
</table>

<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)

<table>
<thead>
<tr>
<th>Array</th>
<th>Coordinate value</th>
</tr>
</thead>
<tbody>
<tr>
<td>axis[0]</td>
<td>1st axis position (unit: mm)</td>
</tr>
<tr>
<td>axis[1]</td>
<td>2nd axis position (unit: mm)</td>
</tr>
<tr>
<td>axis[2]</td>
<td>3rd axis position (unit: mm)</td>
</tr>
</tbody>
</table>

<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)

<table>
<thead>
<tr>
<th>Array</th>
<th>Member</th>
</tr>
</thead>
<tbody>
<tr>
<td>axis[0]</td>
<td>1st axis Pulse</td>
</tr>
<tr>
<td>axis[1]</td>
<td>2nd axis Pulse</td>
</tr>
<tr>
<td>axis[2]</td>
<td>3rd axis Pulse</td>
</tr>
<tr>
<td>axis[3]</td>
<td>4th axis Pulse</td>
</tr>
<tr>
<td>axis[4]</td>
<td>5th axis Pulse</td>
</tr>
<tr>
<td>axis[5]</td>
<td>6th axis Pulse</td>
</tr>
</tbody>
</table>

• 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 controller.

• 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 |

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>øç ēĮĴçĨĩĺĶįļĻĬçķĶĺİĻİĶţçĶķĬĹĨĻİĶţçïĔĖģđđç</td>
<td></td>
</tr>
<tr>
<td>ùç ĚĻĹĨİĮįĻçĨĩĺĶįļĻĬçķĶĺİĻİĶţçĶķĬĹĨĻİĶţçïĔĖģēđç</td>
<td></td>
</tr>
<tr>
<td>÷ç įçïēĮĴçĶķĬĹĨĻİķĮĹĨĻİĶţçĔĖģēđç</td>
<td></td>
</tr>
</tbody>
</table>

[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

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
</table>
| øç ēĮĮçĨĩĺĶįļĻĬçķĶĺİĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻĮķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻĮķĮĹĨĻİķĮĹĨĻİķĮĹĨĻĮķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻİķĮĹĨĻĮķĮĹĨĻİķĮĹĨĻĮķĮĹĨ>Login into the application

HW1483705 123/156
<speed> Specifying the speed

<table>
<thead>
<tr>
<th>Unit</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>%</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<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)

<table>
<thead>
<tr>
<th>Array</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>axis[0]</td>
<td>1st axis Pulse</td>
</tr>
<tr>
<td>axis[1]</td>
<td>2nd axis Pulse</td>
</tr>
<tr>
<td>axis[2]</td>
<td>3rd axis Pulse</td>
</tr>
<tr>
<td>axis[3]</td>
<td>4th axis Pulse</td>
</tr>
<tr>
<td>axis[4]</td>
<td>5th axis Pulse</td>
</tr>
<tr>
<td>axis[5]</td>
<td>6th axis Pulse</td>
</tr>
<tr>
<td>axis[6]</td>
<td>7th axis Pulse</td>
</tr>
<tr>
<td>axis[7]</td>
<td>8th axis Pulse</td>
</tr>
</tbody>
</table>

<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)

<table>
<thead>
<tr>
<th>Array</th>
<th>Pulse value</th>
</tr>
</thead>
<tbody>
<tr>
<td>axis[0]</td>
<td>1st axis Pulse</td>
</tr>
<tr>
<td>axis[1]</td>
<td>2nd axis Pulse</td>
</tr>
<tr>
<td>axis[2]</td>
<td>3rd axis Pulse</td>
</tr>
</tbody>
</table>

<stationData> Station position pulse data structure
ESSStationData
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)

<table>
<thead>
<tr>
<th>Array</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>axis[0]</td>
<td>1st axis Pulse</td>
</tr>
<tr>
<td>axis[1]</td>
<td>2nd axis Pulse</td>
</tr>
<tr>
<td>axis[2]</td>
<td>3rd axis Pulse</td>
</tr>
<tr>
<td>axis[3]</td>
<td>4th axis Pulse</td>
</tr>
<tr>
<td>axis[4]</td>
<td>5th axis Pulse</td>
</tr>
<tr>
<td>axis[5]</td>
<td>6th axis Pulse</td>
</tr>
</tbody>
</table>

<toolNo> Tool number

**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 controller.
- 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
5.3 I/O Signal Read/Write Function

- **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.)

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>YRC1000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>DX200</td>
<td></td>
</tr>
</tbody>
</table>
5.3 I/O Signal Read/Write Function

**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.

<table>
<thead>
<tr>
<th>RS023</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ESReadIO1</td>
</tr>
<tr>
<td>1</td>
<td>ESReadIO2</td>
</tr>
</tbody>
</table>

**REFERENCE**

"ESReadIO2" "ESWriteIO1" "ESWriteIO2" "ESReadIOM" "ESWriteIOM"
## 5.3 I/O Signal Read/Write Function

**ESReadIO2**

Reads I/O signals.

**FORMAT**

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

**ARGUMENTS**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] handle</td>
<td>Target handle value</td>
</tr>
<tr>
<td>[in] ioNumber</td>
<td>I/O address (Sets a signal divided by 10.)</td>
</tr>
</tbody>
</table>

### YRC1000

```plaintext
<table>
<thead>
<tr>
<th>handle</th>
<th>ioNumber</th>
<th>ioData</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

### DX200

```plaintext
<table>
<thead>
<tr>
<th>handle</th>
<th>ioNumber</th>
<th>ioData</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
### 5.3 I/O Signal Read/Write Function

#### DX100, FS100

<table>
<thead>
<tr>
<th>RS023</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ESReadIO1</td>
</tr>
<tr>
<td>1</td>
<td>ESReadIO2</td>
</tr>
</tbody>
</table>

#### 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.

#### REFERENCE

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

[out] ioData I/O signals Data storage pointer
5.3 I/O Signal Read/Write Function

■ 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.) |

YRC1000, DX200

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>2701 to 2956</td>
<td>Network input #27010 to #29567(2048)</td>
</tr>
</tbody>
</table>

DX100, FS100

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>2501 to 2756</td>
<td>Network input #25010 to #27567(2048)</td>
</tr>
</tbody>
</table>

| 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.

<table>
<thead>
<tr>
<th>RS023</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ESWriteIO1</td>
</tr>
<tr>
<td>1</td>
<td>ESWriteIO2</td>
</tr>
</tbody>
</table>

**REFERENCE**

"ESReadIO1" "ESReadIO2" "ESWriteIO2" "ESReadIOM" "ESWriteIOM"
5.3 I/O Signal Read/Write Function

ESWriteIO2

Sets I/O signals.

FORMAT

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

ARGUMENTS

<table>
<thead>
<tr>
<th>[in] handle</th>
<th>Target handle value</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] ioNumber</td>
<td>I/O address (Sets a signal divided by 10.)</td>
</tr>
</tbody>
</table>

YRC1000, DX200

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>2701 to 2956</td>
<td>Network input #27010 to #29567 (2048)</td>
</tr>
</tbody>
</table>

DX100, FS100

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>2501 to 2756</td>
<td>Network input #25010 to #27567 (2048)</td>
</tr>
</tbody>
</table>

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

RETURN VALUE

<table>
<thead>
<tr>
<th>0</th>
<th>Normal completion</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA001</td>
<td>Out of variable number range</td>
</tr>
<tr>
<td>Others</td>
<td>Error codes</td>
</tr>
</tbody>
</table>

REMARKS

Restrictions

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

<table>
<thead>
<tr>
<th>RS023</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ESWriteIO1</td>
</tr>
<tr>
<td>1</td>
<td>ESWriteIO2</td>
</tr>
</tbody>
</table>

REFERENCE

"ESReadIO1" "ESReadIO2" "ESWriteIO1" "ESReadIOM" "ESWriteIOM"
5.3 I/O Signal Read/Write Function

- **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

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 to 1000</td>
<td>RS022=0</td>
</tr>
<tr>
<td>0 to 999</td>
<td>RS022=1</td>
</tr>
</tbody>
</table>

- [out] regData: Register data storage pointer

**RETURN VALUE**

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

**REFERENCE**

"ESReadRegister" "ESReadRegisterM" "ESWriteRegisterM"

**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 be added 1.
5.3 I/O Signal Read/Write Function

- **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
- `[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"

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 to 560</td>
<td>RS022=0</td>
</tr>
<tr>
<td>0 to 559</td>
<td>RS022=1</td>
</tr>
</tbody>
</table>

**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 be added 1.
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.)

YRC1000

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Description</td>
</tr>
<tr>
<td>1</td>
<td>Description</td>
</tr>
<tr>
<td>2</td>
<td>Description</td>
</tr>
<tr>
<td>3</td>
<td>Description</td>
</tr>
<tr>
<td>4</td>
<td>Description</td>
</tr>
<tr>
<td>5</td>
<td>Description</td>
</tr>
<tr>
<td>6</td>
<td>Description</td>
</tr>
<tr>
<td>7</td>
<td>Description</td>
</tr>
<tr>
<td>8</td>
<td>Description</td>
</tr>
<tr>
<td>9</td>
<td>Description</td>
</tr>
</tbody>
</table>

DX200

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Description</td>
</tr>
<tr>
<td>1</td>
<td>Description</td>
</tr>
<tr>
<td>2</td>
<td>Description</td>
</tr>
<tr>
<td>3</td>
<td>Description</td>
</tr>
<tr>
<td>4</td>
<td>Description</td>
</tr>
<tr>
<td>5</td>
<td>Description</td>
</tr>
<tr>
<td>6</td>
<td>Description</td>
</tr>
<tr>
<td>7</td>
<td>Description</td>
</tr>
<tr>
<td>8</td>
<td>Description</td>
</tr>
<tr>
<td>9</td>
<td>Description</td>
</tr>
</tbody>
</table>
5.3 I/O Signal Read/Write Function

DX100, FS100

<table>
<thead>
<tr>
<th>[in] number</th>
<th>Number of I/O signals group</th>
</tr>
</thead>
<tbody>
<tr>
<td>[out] ioData</td>
<td>Multi-data storage pointer</td>
</tr>
</tbody>
</table>

ESMultiByteData
Multi-data structure (1byte)

```c
#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"

NOTE
It needs that Number of I/O signals group is even.
5.3 I/O Signal Read/Write Function

- **ESWriteIOM**

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

**FORMAT**

```c
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.)
  - YRC1000, DX200
    - | Value  | Explanation          |
    - |-------|----------------------|
    - | 2701 to 2956 | Network input #27010 to #29567 (2048) |
  - DX100, FS100
    - | Value  | Explanation          |
    - |-------|----------------------|
    - | 2501 to 2756 | Network input #25010 to #27567 (2048) |

- **[in]** number: Number of I/O signals group

- **[in]** ioData: Multi-data structure
  - `ESMultiByteData`
  - Multi-data structure (1byte)
    ```c
    #define Length_of_Multi_1 (474)
    typedef struct
    {
        CHAR    data[Length_of_Multi_1];
    } ESMultiByteData;
    ```
    - `<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"
5.3 I/O Signal Read/Write Function

- **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
- **[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"

---

**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 be added 1.
ESWriteRegisterM

Sets register data continuously from the specified number.

**FORMAT**

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

**ARGUMENTS**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] handle</td>
<td>Target handle value</td>
</tr>
<tr>
<td>[in] regNumber</td>
<td>Register number</td>
</tr>
<tr>
<td>[in] number</td>
<td>Number of register</td>
</tr>
<tr>
<td>[in] regData</td>
<td>Multi-data structure</td>
</tr>
</tbody>
</table>

**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 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 available.

- 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.
5.4 File Data Transmission Function

- **ESLoadFile**
  Sends the specified file to the robot controller.

**FORMAT**

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

**ARGUMENTS**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] handle</td>
<td>Target handle value</td>
</tr>
<tr>
<td>[in] filePath</td>
<td>Full path storage pointer of sent file</td>
</tr>
</tbody>
</table>

**RETURN VALUE**

- 0 : Normal completion
- Others : Error codes

**REFERENCE**

"ESSaveFile"
5.4 File Data Transmission Function

- **ESSaveFile**

  Receives the specified file from the robot controller.

**FORMAT**

```c
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"
5.4 File Data Transmission Function

- **ESFileListFirst**
  Refreshes the file list and reads the first file name of the list.

**FORMAT**

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

**ARGUMENTS**

- `[in] handle` Target handle value
- `[in] fileType` File type
- `[out] fileName` File name string data storage pointer

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>*.JBI</td>
</tr>
<tr>
<td>2</td>
<td>*.DAT</td>
</tr>
<tr>
<td>3</td>
<td>*.CND</td>
</tr>
<tr>
<td>4</td>
<td>*.PRM</td>
</tr>
<tr>
<td>5</td>
<td>*.SYS</td>
</tr>
<tr>
<td>6</td>
<td>*.LST</td>
</tr>
</tbody>
</table>

**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
5.5 Other Functions

- **ESOpen**

  Opens the connection and Gets a communication handler.

**FORMAT**

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

**ARGUMENTS**

- [in] controllerType  Controller Type
  
<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DX100</td>
</tr>
<tr>
<td>2</td>
<td>FS100</td>
</tr>
<tr>
<td>3</td>
<td>DX200</td>
</tr>
<tr>
<td>4</td>
<td>YRC1000</td>
</tr>
</tbody>
</table>

- [in] ipAddress  IP address

- [out] handle  Communication handler

**RETURN VALUE**

- 0    : Normal completion
- 0x0069 : Hardware Lock Key is not found.
- 0x006A : Hardware Lock Key is found, but is not for MOTOCOM32.
- 0x9000 : Connection Error
- 0x9100 : Parameter has error.
- Others : Error codes

**REFERENCE**

"ESClose"
5.5 Other Functions

- **ESClose**

Close the connection of the specified communication handler.

**FORMAT**

```c
LONG ESClose( HANDLE handle );
```

**ARGUMENTS**

- `[in] handle` Communication handler

**RETURN VALUE**

- `0` : Normal completion
- Others : Error codes

**REFERENCE**

"ESOpen"
5.5 Other Functions

- ESSetTimeOut

Sets a communication control 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

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

<table>
<thead>
<tr>
<th>Motocom32</th>
<th>Motocomes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Definition of functions (include file)</td>
<td>MOTOCOM.h</td>
</tr>
<tr>
<td>Library to link</td>
<td>MOTOCOM32.lib</td>
</tr>
<tr>
<td>DLL file</td>
<td>MOTOCOM32.dll</td>
</tr>
</tbody>
</table>

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 );

Functions to connect

<table>
<thead>
<tr>
<th>Motocom32</th>
<th>Motocomes</th>
</tr>
</thead>
<tbody>
<tr>
<td>BscOpen</td>
<td>ESOOpen</td>
</tr>
<tr>
<td>BscSetCom ( serial )</td>
<td></td>
</tr>
<tr>
<td>BscSetEther ( Ethernet )</td>
<td></td>
</tr>
<tr>
<td>BscSetEServer ( ethernet server )</td>
<td></td>
</tr>
<tr>
<td>BscConnect</td>
<td></td>
</tr>
</tbody>
</table>

Procedure to disconnect

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

<table>
<thead>
<tr>
<th>Motocom32</th>
<th>Motocomes</th>
</tr>
</thead>
<tbody>
<tr>
<td>BscClose</td>
<td>ESClose</td>
</tr>
</tbody>
</table>
### 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.

<table>
<thead>
<tr>
<th>MOTOCOM32</th>
<th>MOTOCOMES</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>File Data Transmission Function</strong></td>
<td></td>
</tr>
<tr>
<td>BscDownload</td>
<td>ESLoadFile</td>
</tr>
<tr>
<td>BscDownloadEx</td>
<td></td>
</tr>
<tr>
<td>BscUpload</td>
<td>ESSaveFile</td>
</tr>
<tr>
<td>BscUploadEx</td>
<td></td>
</tr>
<tr>
<td><strong>Robot Control Function</strong></td>
<td></td>
</tr>
<tr>
<td>BscFindFirst</td>
<td>ESFileListFirst</td>
</tr>
<tr>
<td>BscFindNext</td>
<td>ESFileListNext</td>
</tr>
<tr>
<td>BscFindFirstIMaster</td>
<td>No corresponding function</td>
</tr>
<tr>
<td>BscFindNextIMaster</td>
<td>No corresponding function</td>
</tr>
<tr>
<td>BscGetCtrlGroup</td>
<td></td>
</tr>
<tr>
<td>BscGetCtrlGroupXrc</td>
<td></td>
</tr>
<tr>
<td>BscGetCtrlGroupDX</td>
<td></td>
</tr>
<tr>
<td>BscIsCtrlGroup</td>
<td></td>
</tr>
<tr>
<td>BscIsCtrlGroupXrc</td>
<td></td>
</tr>
<tr>
<td>BscIsCtrlGroupDX</td>
<td></td>
</tr>
<tr>
<td>BscIsTaskInf</td>
<td></td>
</tr>
<tr>
<td>BscIsTaskInfXrc</td>
<td></td>
</tr>
<tr>
<td>BscGetError</td>
<td>ESGetAlarm ( alarm only )</td>
</tr>
<tr>
<td>BscGetError2</td>
<td>ESGetAlarmEx (for applying the sub code character strings)</td>
</tr>
<tr>
<td>BscReadAlarmS</td>
<td></td>
</tr>
<tr>
<td>BscGetFirstAlarm</td>
<td></td>
</tr>
<tr>
<td>BscGetNextAlarm</td>
<td></td>
</tr>
<tr>
<td>BscGetFirstAlarmS</td>
<td></td>
</tr>
<tr>
<td>BscGetNextAlarmS</td>
<td></td>
</tr>
<tr>
<td>BscGetStatus</td>
<td>ESGetStatus</td>
</tr>
<tr>
<td>BscGetUFrame</td>
<td>No corresponding function</td>
</tr>
<tr>
<td>BscGetVarData</td>
<td>ESGetVarData1 (B,I,D,R)</td>
</tr>
<tr>
<td>BscGetVarData2</td>
<td>ESGetVarData2 (B,I,D,R)</td>
</tr>
<tr>
<td>BscHostGetVarData</td>
<td>ESGetStrData (S)</td>
</tr>
<tr>
<td>BscGetVarDataEx</td>
<td>ESGetStrData2 (S)</td>
</tr>
<tr>
<td></td>
<td>ESGGetPositionData (P)</td>
</tr>
<tr>
<td>BscHostGetVarDataM</td>
<td>ESGetVarDataMB (B)</td>
</tr>
<tr>
<td></td>
<td>ESGetVarDataMI (I)</td>
</tr>
<tr>
<td></td>
<td>ESGetVarDataMD (D)</td>
</tr>
<tr>
<td></td>
<td>ESGetVarDataMR (R)</td>
</tr>
<tr>
<td>BscIsAlarm</td>
<td>ESGetStatus</td>
</tr>
<tr>
<td>BscIsCycle</td>
<td>ESGetStatus</td>
</tr>
<tr>
<td>BscIsError</td>
<td>ESGetStatus</td>
</tr>
<tr>
<td>BscIsHold</td>
<td>ESGetStatus</td>
</tr>
<tr>
<td>BscIsJobLine</td>
<td>ESGetJobStatus</td>
</tr>
<tr>
<td>BscIsJobName</td>
<td>ESGetJobStatus</td>
</tr>
<tr>
<td>BscIsJobStep</td>
<td>ESGetJobStatus</td>
</tr>
<tr>
<td>BscIsLoc</td>
<td>ESGetPosition</td>
</tr>
<tr>
<td>BscGetPulsePos</td>
<td></td>
</tr>
<tr>
<td>BscIsPlayMode</td>
<td>ESGetStatus</td>
</tr>
<tr>
<td>BscIsTeachMode</td>
<td>ESGetStatus</td>
</tr>
</tbody>
</table>
### 6.1 Procedure to replace MOTOCOM32 with MOTOCOMES

<table>
<thead>
<tr>
<th>Function</th>
<th>Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>BscIsRemoteMode</td>
<td>ESGetStatus</td>
</tr>
<tr>
<td>BscGetCartPos</td>
<td>ESGetPosition( Base coordinate only )</td>
</tr>
<tr>
<td>BscIsServo</td>
<td>ESGetStatus</td>
</tr>
<tr>
<td>BscJobWait</td>
<td>No corresponding function</td>
</tr>
</tbody>
</table>

**System Control**

<table>
<thead>
<tr>
<th>Function</th>
<th>Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>BscCancel</td>
<td>ESCancel</td>
</tr>
<tr>
<td>BscChangeTask</td>
<td>Target task can be set by each function.</td>
</tr>
<tr>
<td>BscContinueJob</td>
<td>ESStartJob</td>
</tr>
<tr>
<td>BscConvertJobP2R</td>
<td>No corresponding function</td>
</tr>
<tr>
<td>BscConvertJobR2P</td>
<td>No corresponding function</td>
</tr>
<tr>
<td>BscDeleteJob</td>
<td>ESDeleteJob</td>
</tr>
<tr>
<td>BscHoldOff</td>
<td>ESHold</td>
</tr>
<tr>
<td>BscHoldOn</td>
<td>ESHold</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function</th>
<th>Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>BscPutVarData</td>
<td>ESSetVarData1 (B,I,D,R)</td>
</tr>
<tr>
<td>BscPutVarData2</td>
<td>ESSetVarData2 (B,I,D,R)</td>
</tr>
<tr>
<td>BscHostPutVarData</td>
<td>ESStrData (S)</td>
</tr>
<tr>
<td>BscPutVarDataEx</td>
<td>ESStrData2 (S)</td>
</tr>
<tr>
<td>BscPutUFrame</td>
<td>ESStrPositionData (P)</td>
</tr>
<tr>
<td>BscPutUFrameEx2</td>
<td>ESStrPositionData (BP,EX)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function</th>
<th>Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>BscHostPutVarDataM</td>
<td>ESSetVarDataMB (B)</td>
</tr>
<tr>
<td>BscOPLock</td>
<td>ESSetVarDataM1 (I)</td>
</tr>
<tr>
<td>BscOPUnLock</td>
<td>ESSetVarDataMD (D)</td>
</tr>
<tr>
<td>BscReset</td>
<td>ESSetVarDataMR (R)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function</th>
<th>Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>BscMov</td>
<td>ESSetVarData1 (B,I,D,R)</td>
</tr>
<tr>
<td>BscMovEx</td>
<td>ESSetVarData2 (B,I,D,R)</td>
</tr>
<tr>
<td>BscMovEx2</td>
<td>ESSetStrData (S)</td>
</tr>
<tr>
<td>BscMovl</td>
<td>ESSetStrData2 (S)</td>
</tr>
<tr>
<td>BscMovlEx</td>
<td>ESSetPositionData (P)</td>
</tr>
<tr>
<td>BscMDSP</td>
<td>ESSetBpexPositionData (BP,EX)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function</th>
<th>Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>BscStartJob</td>
<td>ESSetVarDataMB (B)</td>
</tr>
<tr>
<td>BscSelectJob</td>
<td>ESSetVarDataM1 (I)</td>
</tr>
<tr>
<td>BscSelectMode</td>
<td>ESSetVarDataMD (D)</td>
</tr>
<tr>
<td>BscReset</td>
<td>ESSetVarDataMR (R)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function</th>
<th>Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>BscSetCtrlGroup</td>
<td>Target control group and task can be set by each function.</td>
</tr>
<tr>
<td>BscSetCtrlGroupXrc</td>
<td>Target control group and task can be set by each function.</td>
</tr>
<tr>
<td>BscSetCtrlGroupDX</td>
<td>Target control group and task can be set by each function.</td>
</tr>
</tbody>
</table>
### 6.1 Procedure to replace MOTOCOM32 with MOTOCOMES

| BscServoOff | ESServo |
| BscServoOn  |         |

**I/O Signal Read/Write Function**

| BscReadIO  | ESReadIO1 |
| BscReadIO2 | ESReadIO2  |
|           | ESReadIOM  |

| BscWriteIO | ESWriteIO1 |
| BscWriteIO2| 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 installing 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.)