YASKAWA

MOTOCOM32
OPERATOR'S MANUAL

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

MANUAL NO.
HW9482689 ⟨11⟩ 1/313

DANGER

- **This manual explains the MOTOCOM32. 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.**

CAUTION

- **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.**

NOTICE

- **The drawings and photos in this manual are representative examples and differences may exist between them and the delivered product.**
- **YASKAWA may modify this model without notice when necessary due to product improvements, modifications, or changes in specifications. If such modification is made, the manual number will also be revised.**
- **If your copy of the manual is damaged or lost, contact a YASKAWA representative to order a new copy. The representatives are listed on the back cover. Be sure to tell the representative the manual number listed on the front cover.**
-

NOTE

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”.

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.

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.

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 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.

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:

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

Description of the Operation Procedure

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

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

Registered Trademark

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

1 INTRODUCTION

2 SETUP

3 OPERATION OF HIGH SPEED JOBEXCHANGER FUNCTIONS

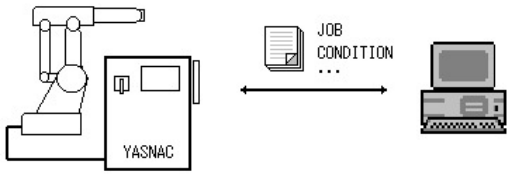
4 OPERATION OF HOST CONTROL FUNCTION

1 INTRODUCTION

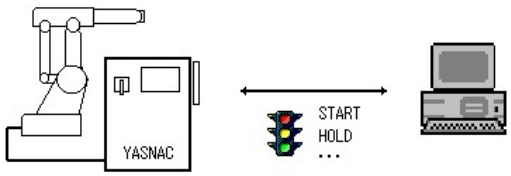
1.1 MOTOCOM32

MOTOCOM32 is software for data transmission between a personal computer and YASKAWA industrial robot controller YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, YASNAC XRC, MRC, MRCII, ERC and ERCII.
High-speed transmission can be achieved by connecting the YRC1000, YRC1000micro, DX200, DX100, FS100, NX100 or YASNAC type controller and the personal computer with a serial cable (RC232C cable), or by connecting the YRC1000 or YRC1000micro or DX200 or DX100 or FS100 or NX100 or YASNAC XRC or MRC to a LAN by using a Ethernet cable (the YASNAC XRC or MRC requires mounting an Ethernet I/F board).
This software operates on a personal computer and has the following function

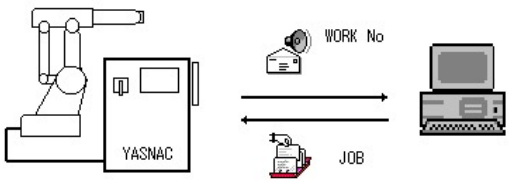
High Speed JobExchanger Function
Loads or saves files such as jobs or condition data according to the command from a personal computer.



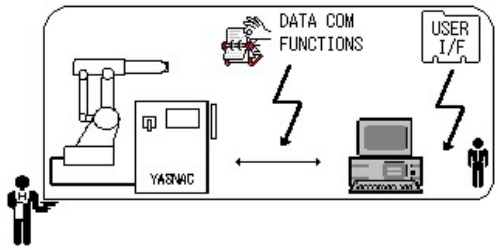
Host Control Function
Can perform the following tasks easily according to the command from a personal computer.
- Reads robot status (current position, alarm, error, servo status, etc.) or controls the system (start, hold, job call, etc.)
- Reads or writes I/O signals.



Automatic Work Job Replacing Function
Can automatically replace a work job using DCI (data communication by instruction) transmitting function.
In order to compensate for the memory shortage of robots, a personal computer is used as external memory to receive work number information from the robot and send back corresponding jobs.



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. (Motocom32.dll)
-Describes application preparation procedures using sample programs including the above functions.



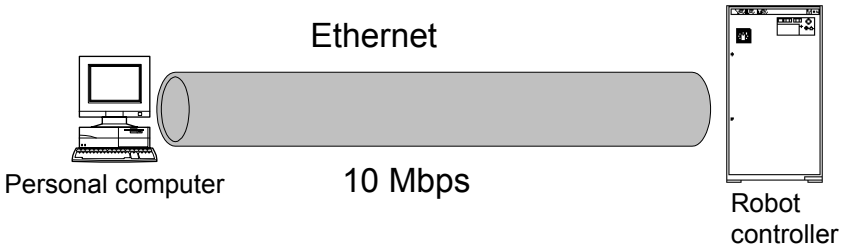
1.2 Features of Ethernet Communications

The Ethernet I/F board and the "Ethernet" function of the MOTOCOM32 transmit data at higher than normal speeds.

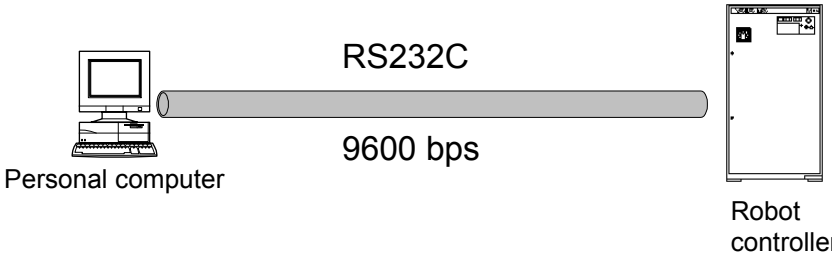
■ High speed transmission

In comparison with transmissions using RS232C, higher speed transmissions are possible with the Ethernet.

When Ethernet is used



When RS232C is used



NOTE The above transmission speed is the communication speed between network devices, not including the time used for format check of transmitted data, etc.

■ Transmissions between a multiple number of HOSTS

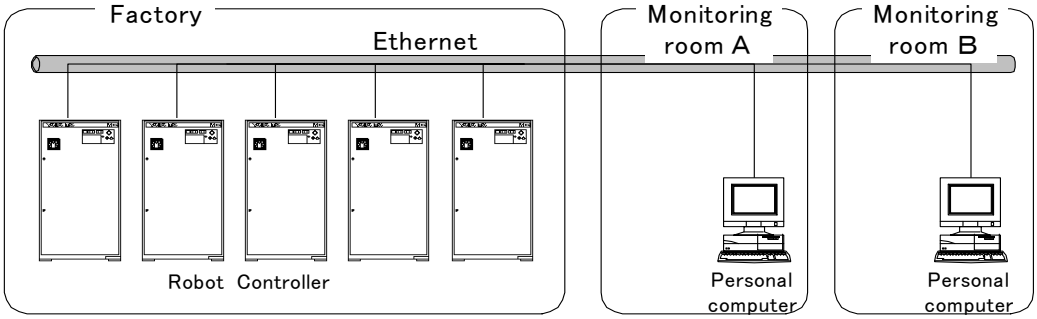
As N:N transmission is possible with an Ethernet cable, the following system configurations

can be prepared.

NOTE Refer to paragraph " 2.4 Restrictions " with the following Configuration Examples.

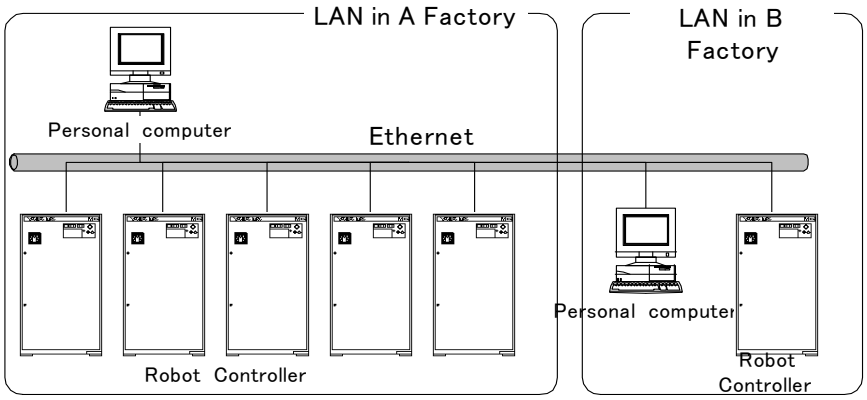
<Configuration Example>

Since an Ethernet cable can be connected to a multiple number of network devices, the factory operation state and alarm occurrences can be monitored from several places.



<Configuration Example 2>

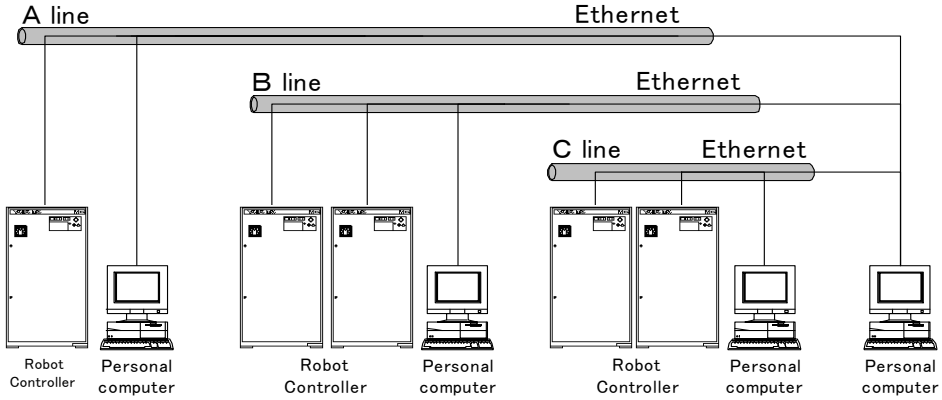
By connecting the LANs of different factories with one Ethernet cable, transmission in each factory can be executed simultaneously. The transmission between the robot controller and the personal computer in factory A does not interfere with the transmission between the robot controller and the personal computer in factory B. Transmission can also be done between factories. (In both cases, the settings should be correct.)



<Configuration Example 3>

With the Ethernet cables, the job on a personal computer can be executed on the robot controller by installing a personal computer for each production line and transferring the job from the personal computers to the robot controller. Then, by connecting one personal computer to the Ethernet cables in all the production lines, monitoring of the state of all the production lines

and data backup can be executed.



1.3 Hardware Requirements for MOTOCOM32

OS	Microsoft Windows 10 (64bit) Microsoft Windows 7 ServicePack1 (32bit / 64bit) JAPANESE and ENGLISH Windows version are supported only. *1
Reguired Memory	128 Mbyte or more
Hardware Disk Capacity for Installation	50 Mbyte or more
Display	Supported by MS-Windows
Robot Controller	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, YASNAC XRC, MRC, MRC2, ERC, and ERC2.
Transmission Cable	Ethernet cable or RS232C cable
Hardware Lock Key	Used under single user environment. For details, refer to the following section " 1.4 Hardware Lock Key ".

*1 MS-Windows10 and MS-Windows 7 is a registerd trademark of Microsoft Coporation,USA.

NOTE

- The controller Data Transmission function, Ethernet function, Ethernet board, transmission cable, or the personal computer OS are not included in this package.
- Use either an RS-232C cable or an Ethernet cable for transmission, depending on the data transmission function specifications set in the robot controller manuals. Before starting this software, check the hardware and software specifications of the robot controllers.
- Ethernet transmission is not available for the YASNAC MRC2/ ERC / ERC2 since they do not support the Ethernet function. For MRC, Ethernet transmission is available for version 4.111 or later.
- Transmission with RS-232C cable is not available for YRC1000micro since it does not support the RS-232C cable.
- To create a transmission application, a development tool such as Microsoft Visual Basic Ver6.0 or Microsoft Visual C++ Ver6.0 is required.

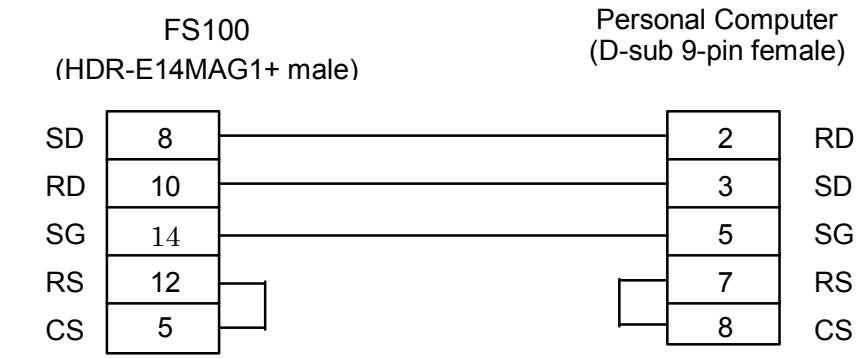
For softwares and devices, refer to the robot controller Operator's Manuals, Data Transmission Operator's Manual, Ethernet I/F Board Instructions (or "Ethernet Function Instructions" for the YRC1000, YRC1000micro, DX200, DX100, FS100, NX100), Manuals for MS-Windows, etc.

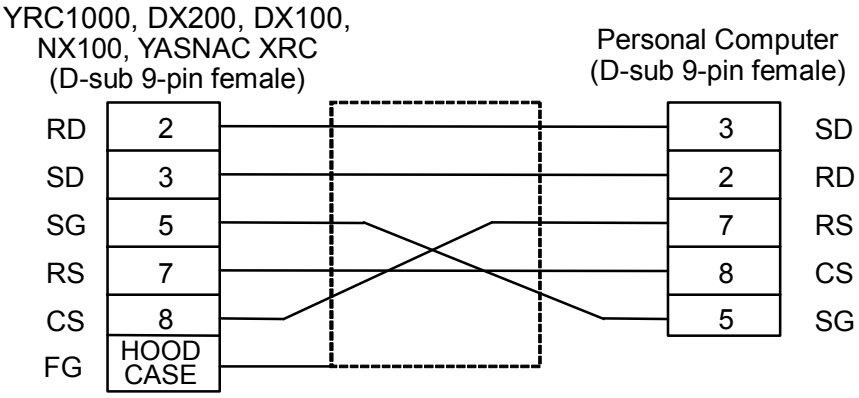
1.3.1 RS-232C Transmission cable specifications

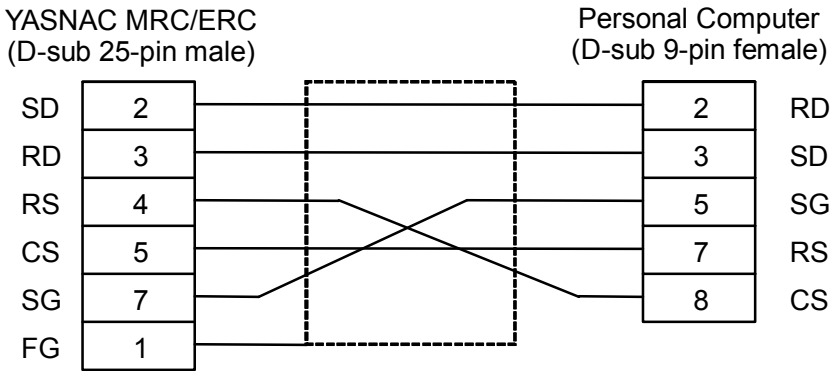
RS-232C Transmission cable specifications are as follows.

NOTE When using an Ethernet cable, the RS232C cable is not required.

[PC/AT]







1.4 Hardware Lock Key

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

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

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

<Installation of driver>

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

Execute "\SentinelDriver\Sentinel System Driver Installer 7.6.0.exe" of installation CD-ROM. Refer to"\SentinelDriver\Manual\Sentinel System Driver ReadMe.pdf" for the details of installation.

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

Refer to " 8.1 Frequently-asked questions " for other countermeasures concerning hardware lock key.

2 SETUP

2.1 Installing MOTOCOM32

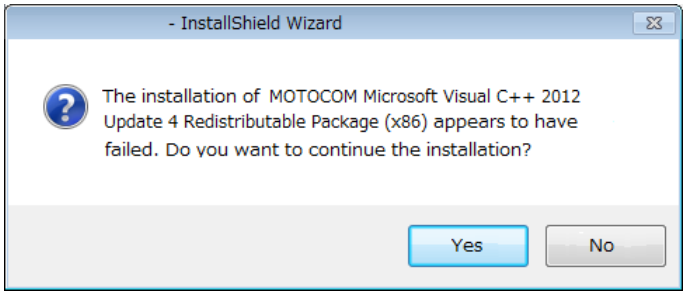
1. It is strongly recommended that you exit all applications before running the setup program.

NOTE Be sure to login in administrator mode when installing the MOTOCOM32 in Windows 7/10, or else the system related DLL files in Windows might not be updated.

2. When the install CD is inserted into the CD-ROM drive, the [MOTOCOM32-InstallShield Wizard] window appears automatically.
If you are using Windows 7/10, the [User Account Control] dialog appears, so click [OK] inthe dialog.

3. Follow the on-screen instructions.

NOTE
- In the case of Windows 10, [VB 6.0 comm control] will be installed after installation. Follow the displayed instructions.
- If the following message appears during the installation, press [Yes] to continue installation.
  This message is displayed when the installation of redistribution package of Visual C ++ 2012 has been executed but installation is required.
  If this message is displayed, run manually
  \ISSetupPrerequisites\{BF2F04CD-3D1F-444e-8960-D08EBD285C3F}\
  vcredist_x86.exe
  after installing MOTOCOM32, and install or repair the redistribution package of Visual C ++ 2012.



4. When the setup is completed, [High Speed JobExchanger], [Host Control], [Auto Job Changer], and [MOTOCOM 32 Help] are registered under [MOTOCOM 32] folder that appears by clicking the [Start] button in the task bar to select [Program] and then [Motoman] (In the case of Windows10, all menu appears directly under [Motoman]).

5. Connect the hardware key to the printer port.
For details, refer to " 1.4 Hardware Lock Key " in this chapter.

2.2 Robot Controller Configuration

When transmission to the controller is done using MOTOCOM32, whether by RS232C or Ethernet, the following controller configuration are necessary .

■ Parameter settings

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

■ Transmission protocol designation

RS000 : Protocol designation for Std. port #1

RS001 : Protocol designation for Std. port #2 (MRC only)

Settings for parameter RS000 / RS001 :
0 : Not used
1 : System reserved
2 : BSC LIKE protocol (used for data transmission)
3 : FC1 protocol

These parameters are used to designate the transmission protocol for Std port #1, port #2 or the Ethernet board for the robot controller. If the Ethernet communication function is not to be used, RS000 and RS001 correspond to the Std port #1 and port #2 respectively as the above. When the Ethernet communication function plus either Std port #1 or port #2 are used, parameters according to this port number must be set. Any other parameters can be used for Ethernet communication.
To use the MOTOCOM 32, set RS000 or RS001 to the value "2."
For example, if port #1 is already used for FC1 or FC2 and its parameter RS000 is set to the value "3," RS001 is required to be set to the value "2" to use the MOTOCOM 32.

NOTE RS000/001 parameters cannot have the same setting.
Ethernet communication function only supports the BSC LIKE protocol.

■ Command Remote Setting

In order to transmit with the host control function, the remote command must be set to valid. However, when DCI function or stand-alone function are to be used, the remote command must set to invalid.

For MRC, ERC:

Using the programming pendant, the customer options are set in the maintenance mode. For detail on the maintenance mode operations, please refer to the controller Operator's Manual. To be able to the command remote to valid, the customer options need to be set as follows.

Customer options
I/O = Not used

Command mode = Used (This must be always set to "Used.")
PP/PBOX (programming pendant / playback box) = Not used

For YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC:

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"
"YRC1000micro 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"
"NX100 Options: Instructions for Data Transmission Function"
"YASNAC XRC Options: Instructions for Data Transmission Function"
"YASNAC MRC Options: Instructions for Data Transmission Function"
"YASNAC ERC Data Transmission Function, User Manual"

2.3 Environmental Settings for Use of Ethernet

The following configurations are required for Ethernet transmissions. These settings are not necessary for the RS232C communication. Refer to the "RS232C Condition".

2.3.1 MOTOCOM32 Application Settings

■ parameter Setting

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

2.3.2 Personal Computer Settings

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

■ Hardware settings

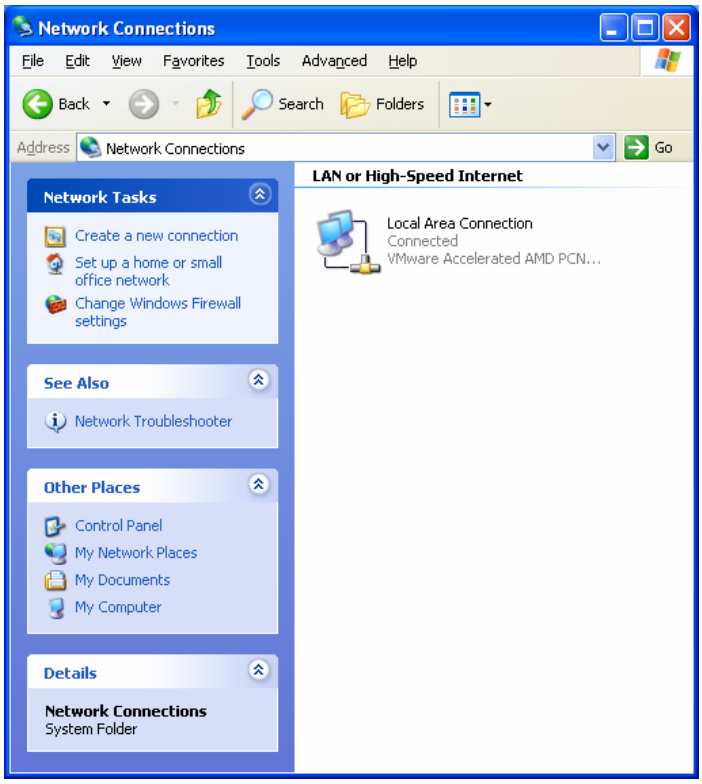
Before using the MOTOCOM32, connect the Ethernet board to the personal computer and check if the Ethernet board operates correctly.
For connection methods, refer to the manual for the Ethernet board used.

■ Windows Network settings

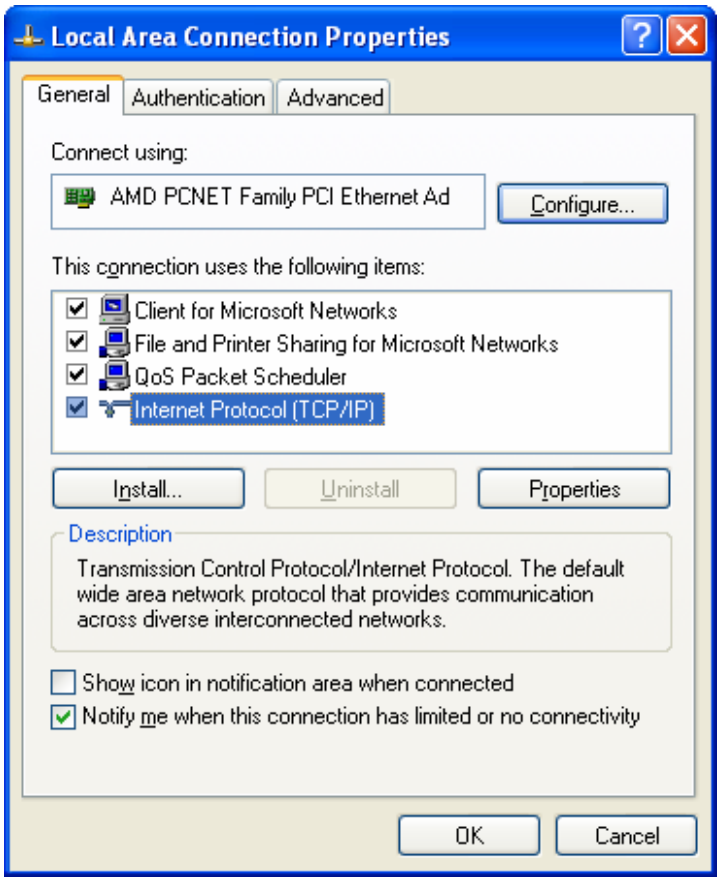
To communicate via the Ethernet, set the settings related to the Windows network. (The example below is based on Windows XP.)

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].

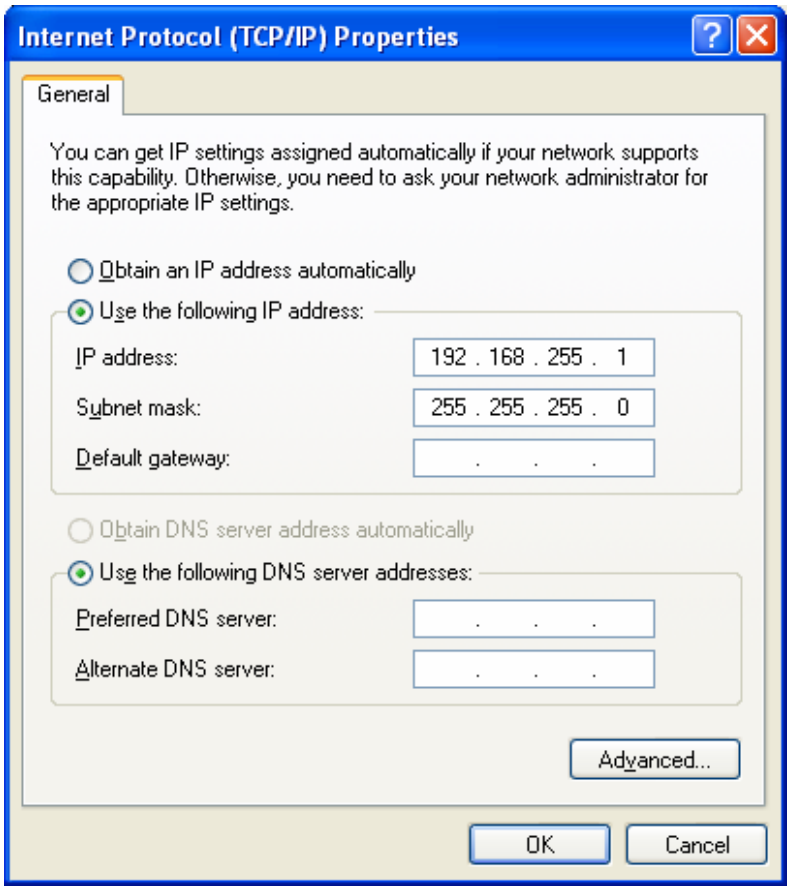
2. Select the network connection to use for data transmission, and select the "Properties" from the right click menu.



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.



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

2.3.3 Robot Controller Setting

■ Hardware settings

To communicate using Ethernet, YRC1000 use the Ethernet connector on the ACP01 board; for the YRC1000micro use the Ethernet connector on the ACP30 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 NX100 use the Ethernet connector on the NCP01 board; for the YASNAC XRC/MRC, the installation of an Ethernet I/F board is required. To setup the IP address and submask, refer to the following manuals.

"YRC1000 Options: Instructions for Ethernet Function"
"YRC1000micro Options: Instructions for Ethernet Function"
"DX200 Options: Instructions for Ethernet Function"
"DX100 Options: Instructions for Ethernet Function"

"FS100 Options: Instructions for Ethernet Function"
"NX100 Options: Instructions for Ethernet Function"
"YASNAC XRC Options: Instructions for Ethernet I/F Board"
"YASNAC MRC Options: Instructions for Ethernet I/F Board"

■ 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.
When the host control function is used, there is no need to set the SERVER ADDRESS. It is used for DCI or stand-alone function. For details, refer to the following manuals.
"YRC1000 Options: Instructions for Ethernet Function"
"YRC1000micro Options: Instructions for Ethernet Function"
"DX200 Options: Instructions for Ethernet Function"
"DX100 Options: Instructions for Ethernet Function"
"FS100 Options: Instructions for Ethernet Function"
"NX100 Options: Instructions for Ethernet Function"
"YASNAC XRC Options: Instructions for Ethernet I/F Board"
"YASNAC MRC Options: Instructions for Ethernet I/F Board"

2.3.4 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"
"YRC100micro Options: Instructions for Ethernet Function"
"DX200 Options: Instructions for Ethernet Function"
"DX100 Options: Instructions for Ethernet Function"
"FS100 Options: Instructions for Ethernet Function"
"NX100 Options: Instructions for Ethernet Function"
"YASNAC XRC Options: Instructions for Ethernet I/F Board"
"YASNAC MRC Options: Instructions for Ethernet I/F Board"

2.3.5 High Speed Link Server

The High Speed Link Server uses MOTOCOM32 for communication via the Ethernet. It can-

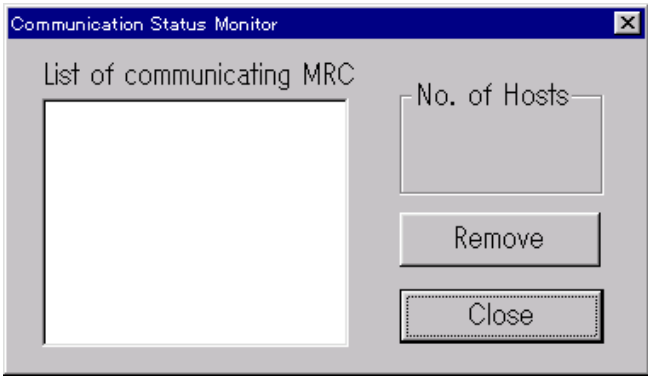
not be used for communication via RS232C.
The following explains the High Speed Link Server.

■ Server

To communicate with a robot controller via the Ethernet, the server program (HSLSR32.EXE) is required.
The server receives a request from the application using MOTOCOM32 and sends/receives the data. When the application begins transmission to the robot controller via Ethernet, it automatically starts the server program.
To close the server program, right-click the [High Speed Link Server] in the task bar to display the menu. Click the [Exit] item from the menu to exit the server program.
To communicate between a personal computer and a robot controller, the server program and the related DLL files have to be located in the same folder as the application.
If the priorities of the applications other than the server program (include the applications using MOTOCOM32) are higher than that of the server program, there is a possibility not to send/receive data correctly.
It is necessary that the priority of the server program is higher than, or equal to, those of the other applications.

■ Communication Status Monitor

The [Communication Status Monitor] display appears when the High Speed JobExchanger is starrted up.



To hide the [Communication Status Monitor] window for some reason (for example, to work on by another operation), right-click the [High Speed Link Server] in the task bar to display the menu. Select [Monitoring] from the menu to switch the [Communication Status Monitor] display ON and OFF.

■ Deletion of transmission information

If a Ethernet transmission error occurs for any of the following reasons during communication, the transmitted information may sometimes remain in the robot controller and/or the personal computer.

(a) The Ethernet cable is removed during transmission.
(b) The [REMOTE] button on the playback box of the robot controller is turned OFF during transmission.

If the transmission cannot be continued, reset the information to be transmitted by the following procedures.

(Robot Controller side)

(1) Press the [Reset] key if any error message is shown in the display of the programming pendant.
(2) Turn OFF the [Remove] button on the playback box once and then turn it ON again.

(Personal Computer side)

(1) Press the [Remove] button in the "Communication Status Monitoring" window.
(2) To reset information, input the IP address of the robot controller where the error occurred, and press the [OK] button

2.4 Restrictions

When using the MOTOCOM 32, pay attention to the following restrictions.

2.4.1 YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, YASNAC XRC/MRC and Personal Computer Restrictions

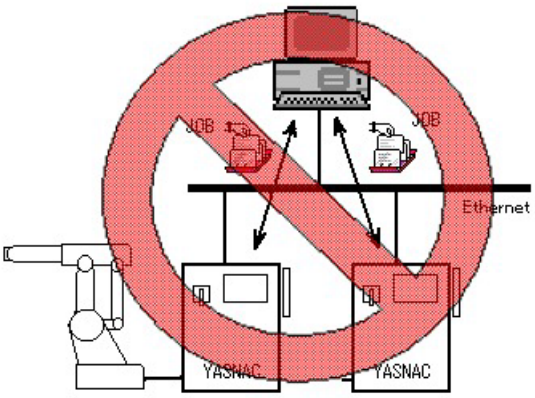
■ The port used for TCP/IP

The MOTOCOM 32 uses TCP/IP for the communication protocol. To communicate in TCP/IP, the service identification numbers called "Port No" are used internally, while MOTOCOM 32 uses the port numbers from 10000 to 10008 for the data transmission.
When these numbers overlap with the numbers used for other network devices, correct communication cannot be performed.
To use the MOTCOM 32, be sure in advance that any network device in the same network does not use the above explained port numbers.

2.4.2 Personal Computer Restrictions

■ Same file access

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



■ Same Window handle cannot be used

When using Ethernet Communication, a Windows handle is designated in the BscSetEther() function argument. This Windows handle needs to be a real Windows handle. Also, when there are multiple robots, the same Windows handle cannot be designated for all of them.

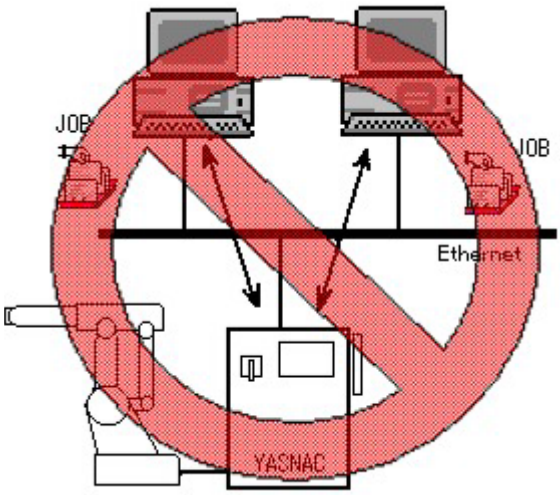
■ Prohibit the use in multi-thread

When using Ethernet Communication, Be sure to execeute the functions of MOTOCOM32 in the main thread.

2.4.3 YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, YASNAC Robot Controller Restrictions

■ Multiple personal computer access

With the MOTOCOM32, one personal computer can communicate with one robot controller. Simultaneous communication with a multiple number of personal computers is not possible.
(However, the simultaneous communication between one personal computer and a multiple number of robot controllers is possible.) With YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, it is possible to manage connection between multiple personal computers and one controller by using the Ethernet Server function to establish the connections.



■ CMOS batch storage

The BSC LIKE protocol and the FC1 protocol are available for the MOTOCOM 32 to communicate with external devices. The MOTOCOM 32 uses the BSC LIKE protocol for transmission. As the CMOS batch storage uses the FC1 protocol, CMOS batch storage is not available in the MOTOCOM 32. For CMOS batch storage, use the YASNAC FC1/FC2 or PC Card/CF.

2.4.4 String Variable Transmission Support

Communication of string variables (character type variables) is supported by the YRC1000 or YRC1000micro, DX200, DX100, FS100, NX100 version 3.00-00 and after. In order to perform transmission of string variables, MOTOCOM32 version 3.10 and after must be used.

2.5 Execution of MOTOCOM 32 Programs

The MOTOCOM 32 program consists of three programs;

[High Speed JobExchanger],
[Host Control],
[Auto Job Changer].

To execute each program, select the application to be executed from the start menu.

3 OPERATION OF HIGH SPEED JOB-EXCHANGER FUNCTIONS

3.1 What is the High Speed JobExchanger?

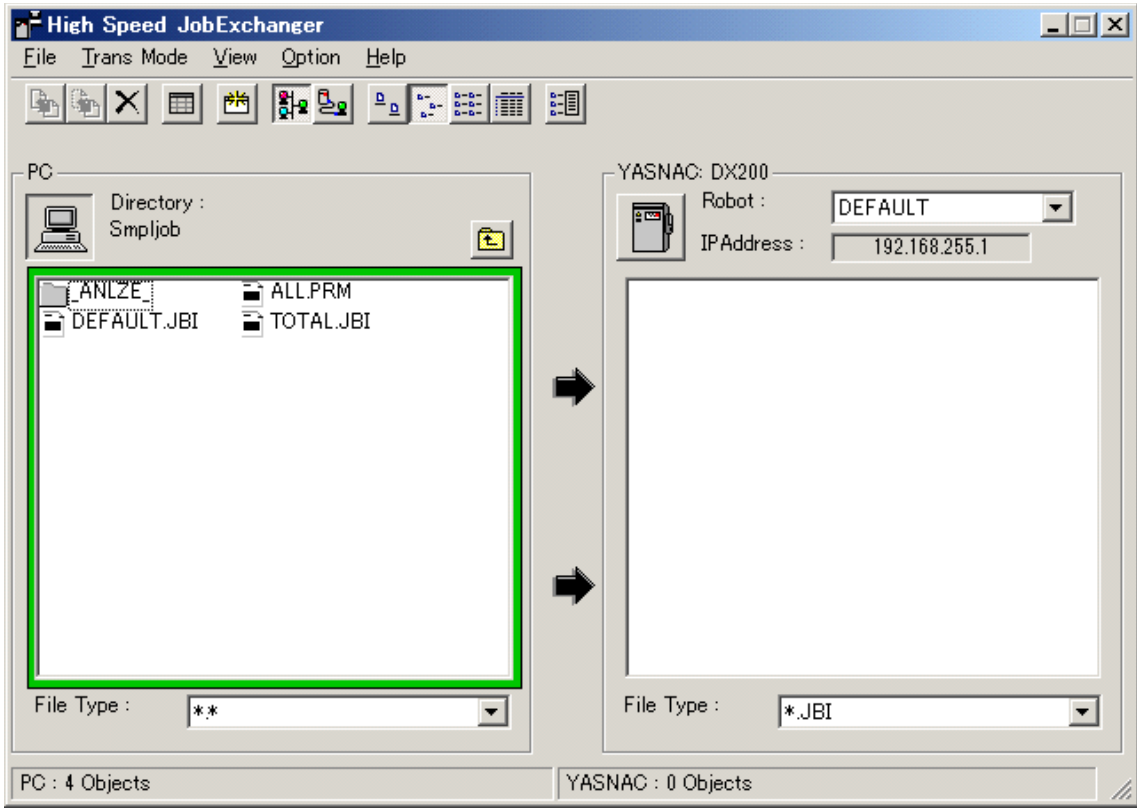
The High Speed JobExchanger is an application software for transfer of the files relating to the robot, by connecting YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, YASNAC XRC/MRC to a personal computer with Ethernet cables or RS232C cables.
The following lists shows the transmission commands supported by the file data transmission function.

Job Transmission	UPLOAD/ DOWNLOAD	Single job
		Related job
Various Condition Data Transmis-sion, System Informa-tion File Transmis-sion	UPLOAD/ DOWNLOAD	Transmission of varios condion data supported by YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC, ERC

NOTE

- To communicate with the robot controller using the High Speed JobExchanger, set the robot controller to "Command Remote".
- The only robot controllers for Ethernet communication are YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, YASNAC XRC/MRC. YASNAC MRC2, ERC, and ERC2 are not available.

3.2 Main Display



3.2.1 Menu Structure

■ [File] menu

Copy	Copies marked file from the controller to the personal computer or vice versa.
Move	Moves marked file from the controller to the personal computer or vice versa.
Delete	Deletes marked file in the personal computer or in the controller.
Select All	Selects all files in the active window.
Display File Contents	Opens the marked file in the window on the controller side.
Print File	To print out the contents in a job file. You must first mark and open the file in the window on the controller side.
Exit High Speed JobExchanger	Terminates the program. In the [Setup] command of the [Option] menu, if [Not Save Current Settings] is not marked, the currently set values are stored in the initial file to operate with the same values at the next startup.

■ [Trans Mode] menu

Ethernet	Transmits via the Ethernet.
RS232C	Transmits via a RS232C.

■ [View] menu

Large Icons	The file lists of the currently selected object (the personal computer or the robot controller) are displayed in large icons.
Small Icons	The file lists of the currently selected object (the personal computer or the robot controller) are displayed in small icons.
List	The file lists of the currently selected object (the personal computer or the robot controller) are displayed in list format.
Details	The file lists on the personal computer side are displayed in details. In the details display, "Name," "Size" and "Modified" are displayed. NOTE The details display is available only for the file list of the personal computer, but not that of the robot controller.
Arrange Icons	Icons can be arranged according to the "Name," "Size" or "Modified." When the file lists of the personal computer are displayed in "Details," clicking the title can also arrange the icons.

Refresh	The file lists of the currently selected object (the personal computer or the robot controller) are refreshed. After the following operations, always execute [Refresh] operation. • When copying, moving or deleting the files of the personal computer using applications other than the High Speed JobExchanger. • When moving or deleting the files of the robot controller using an external memory unit such as FC1, FC2.

■ [Option] menu

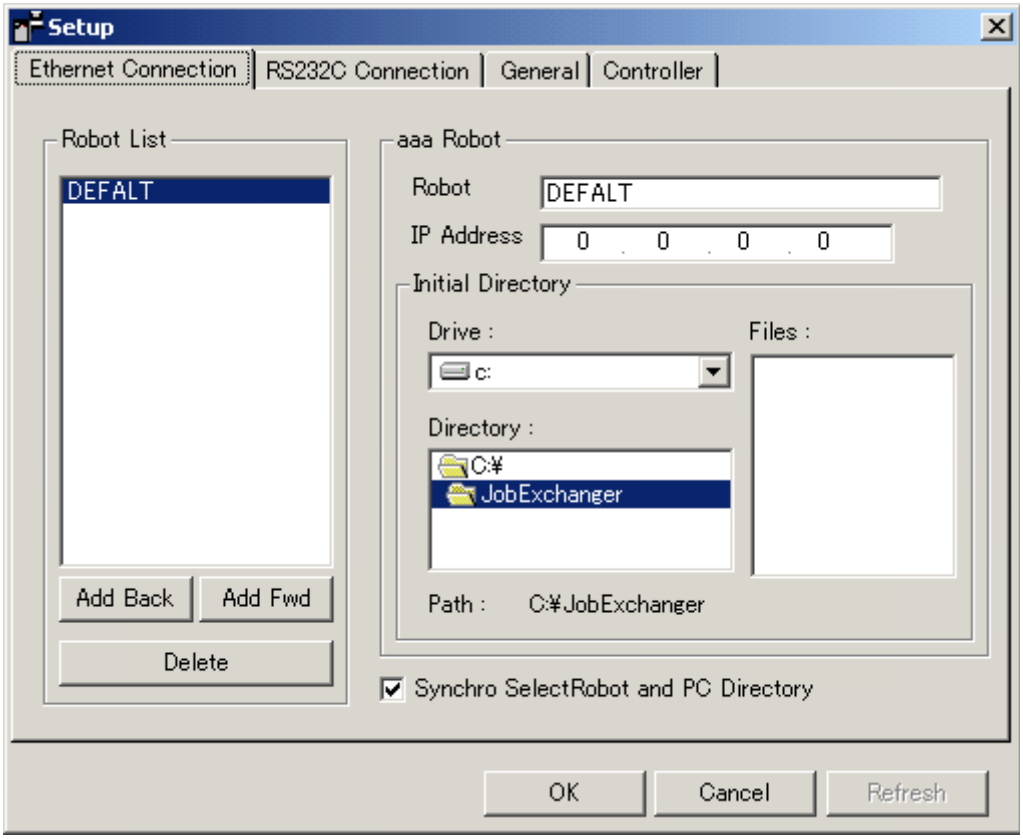
Setup	Set the configurations necessary to operate the High Speed JobExchanger. Select the [Option] menu and the following [Setup] dialog box appears. Enter the required information to set each item.
Upload Whole Jobs	By selecting this from the [Option] menu and inputting the name of the file containing the names of files to be uploaded, a multiple number of files can be uploaded. For details, refer to " 3.3.9 Batch Processing ".
Download Whole Jobs	By selecting this from the [Option] menu and inputting the name of the file containing the names of files to be downloaded, a multiple number of files can be downloaded. For details, refer to " 3.3.9 Batch Processing ".
Select Language	Every time you start the High Speed JobExchanger for Windows, you will be given the facility to choose the language for menus and commands. For details, refer to " Select Language ".

Setup

Set the configurations necessary to operate the High Speed JobExchanger.

Ethernet Condition

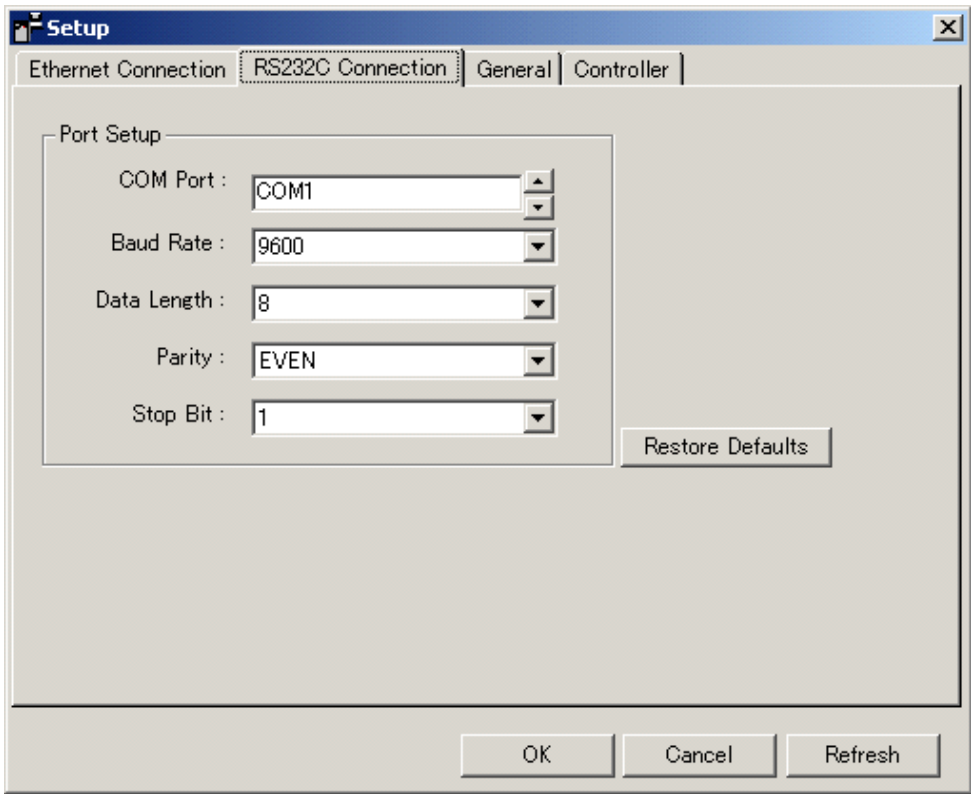
The robot information set according to Ethernet transmission parameters is stored in the file (robot.ini), and used for communicating with the robot controller.
Robot name, IP address and initial directory are registered for each robot.



Robot List	A list of robots registered is displayed. Clicking a robot name displays the corresponding parameters (Robot name, IP address, Initial directory in the personal computer) of the selected robot.
Add Back	A new robot is added to the list and inserted before the currently highlighted robot, and its details entered on the right side of the dialog.
Add Fwd	A new robot is added to the list and inserted after the currently highlighted robot, and its details entered on the right side of the dialog.
Delete	Removes the robot from the list. NOTE The robot currently being used cannot be removed.
Robot	Robot name to identify the robots.
IP Address	IP address used for Ethernet communication. Input the same values as the values set for the destination, robot controller.
Initial Directory	Sets the initial directory of each robot. The initial directory is used when changing the robot required for communication.
Synchro Select Robot and PC Directory	When this item is marked, changing the [Robot] (robot name) on the robot controller side changes the file list display of the personal computer side, to the [Initial Directory] of the set robot name.

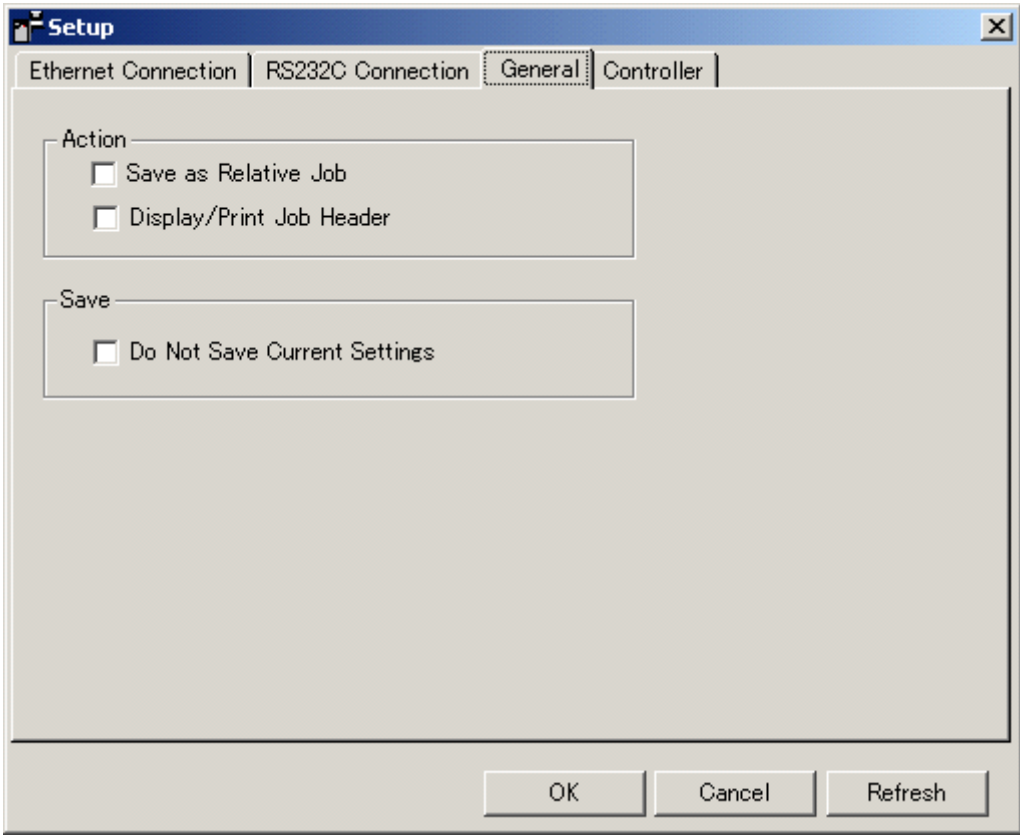
RS232C Condition

Set the communication port for RS232C communication with the robot controller. Select "COM Port," "Baud Rate," "Data Length," "Parity," and "Stop Bit" from the list according to the setting of the robot controller.
Clicking the [Restore Defaults] button displays the default data.



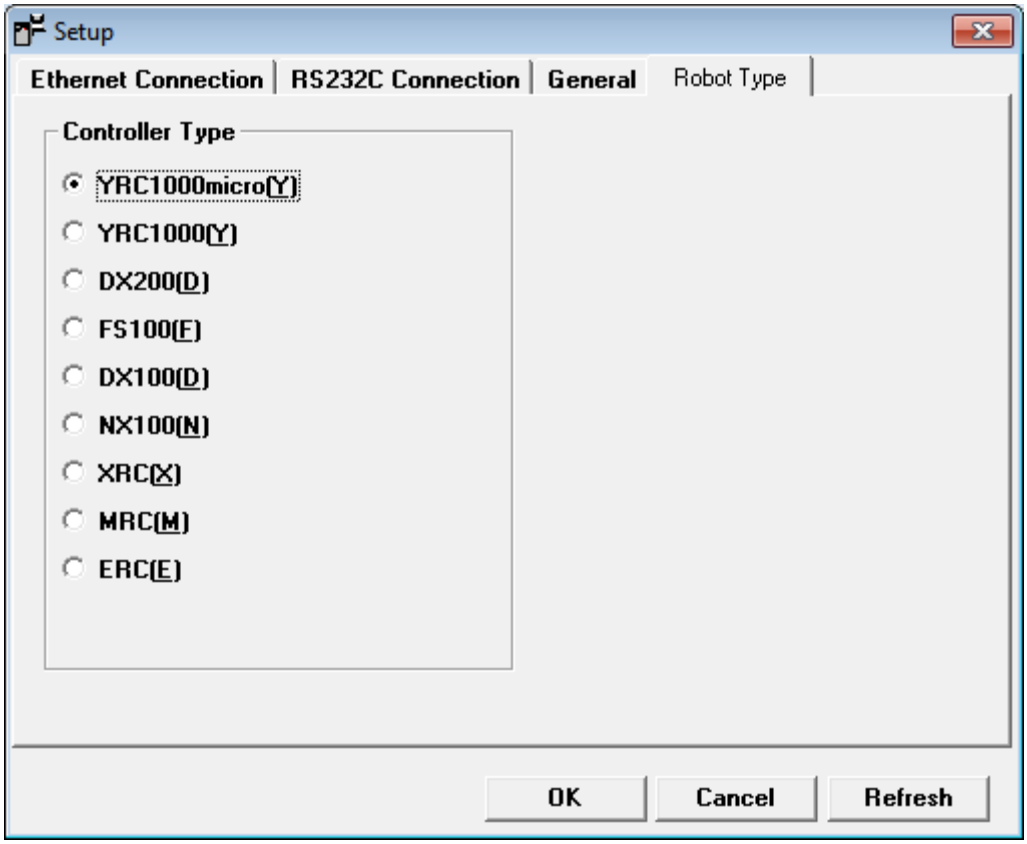
Port Setup	Set the communication parameters for serial communication. Communication with the robot controller is executed according to the parameters set in this dialog box.
Restore Defaults	Selecting this button changes the set value for the communication port to the default value.

General



Save as a Related Job	If this check box is marked, the job from the robot controller is received as related jobs.
Display/Print Job Header	If this check box is marked, a job is opened or printed out including header information. NOTE The related jobs and the condition files are not the subject of this parameter.
Not Save Current Settings	If this checkbox is marked, the current settings are reset to the last settings when High Speed JobExchanger is terminated. When the current settings are not to be reflected on the next startup, for example, after the environment has been changed temporarily, mark this checkbox.

Controllerl



Controller Type	Select either “YRC1000”, “YRC1000micro”, "DX200", "DX100", "FS100", "NX100", "XRC", "MRC" or "ERC" from the list, for the robot controller connected to the High Speed JobExchanger.

Select Language

Every time you start the High Speed JobExchanger for Windows, you will be given the facility to choose the language for menus and commands.
Choose the language and then click the [OK] button.
It is even possible to correct and change each word or to create a totally new language file.

For details, refer to " 3.4 Editing of Language Files ".

Ignore language selection at program start	Click the box [Ignore language selection at program start], and the [Select Language] display does not appear when starting the High Speed Job Exchanger software. Clicking the box again removes the check mark and displays the [Select Language] window when starting.

NOTE Some languages may not be correctly displayed unless fonts are set correctly.

■ [Help] menu

Help Topics	From this, you'll get short information. A brief explanation about commands, functions and other related topics are given to provide assistance when running the software.
Version	The product version is shown. Closes by clicking the [OK] button.

3.2.2 Tool Bar

Some of the operations are also available from the button on the tool bar.

	Copies marked files from the controller to the personal computer or vice versa. Menu : {File} - {Copy}

	Moves marked files from the controller to the personal computer or vice versa. Menu : {File} - {Move}
	Deletes marked files in the personal computer or in the controller. Menu : {File} - {Delete}
	Selects all files in the active window. Menu : {File} - {Select All}
	Updates the file list. Menu : {View} - {Refresh}
	Executes the Ethernet transmission. Menu : {Trans Mode} - {Ethernet}
	Executes the RS232C transmission. Menu : {Trans Mode} - {RS232C}
	Displays the files using large icons. Menu : {View} - {Large Icons}
	Displays the files using small icons. Menu : {View} - {Small Icons}
	Displays the files in list. Menu : {View} - {List}
	Displays the files of the personal computer in details. Menu : {View} - {Details} NOTE The detailed display is available only for the files of the personal computer.
	Displays the contents of the selected file in the window of the controller side. Menu : {File} {Display File Contents}

3.3 Operation Procedure

3.3.1 Starting Up High Speed JobExchanger

1. To start up the High Speed JobExchanger, click the [Start] button in the task bar and select the icon [Program], [Motoman], [MOTOCOM 32], and then [High Speed JobExchanger] (In the case of Windows10, [Motoman] then [High Speed JobExchanger]).
2. The message "Read job list from YASNAC. Ready?" will appear asking whether or not the computer is to be connected to the robot controller. Click the button [Yes] or [No]. If the [No] button is selected, the computer will not be connected to the controller but will run separately.
3. The main display of the High Speed JobExchanger is displayed. The personal computer information is shown on the left side of the display. The robot controller information is shown on the right side of the display.

3.3.2 Copying Files

1. Select the file to be copied from the list. (For selecting a multiple number of files, see "3.3.5 Selecting a Multiple Number of Files")
2. Select the {Copy} command from the {File} menu, or select  from the tool bar.
3. Follow the displayed message.

NOTE
- To receive the job on the robot controller side as related job, select [Save as a Related Job] in the [Setup] dialog box of the {Option} menu in advance.
- To transmit the related jobs in the personal computer side, select the file with the extension "JBR." Jobs with the same name on the robot contoller side should be deleted in advance.

■ File operations other than Menu and Tool Bar

Popup menu

By right-clicking in the file list window, a popup menu appears and the file operations are available.

3.3.3 Moving Files

1. Select the file to be moved from the list. (For selecting a multiple number of files, see "3.3.5 Selecting a Multiple Number of Files".)
2. Select the {Move} command from the {File} menu, or select  from the tool bar.
3. Follow the displayed message.

NOTE
- To receive the job on the robot controller side as related job, select [Save as a Related Job] in the {Setup} dialog box of the {Option} menu in advance.
- The condition files on the robot controller side cannot be moved.

3.3.4 Deleting Files

1. Select the file to be deleted from the list. (For selecting a multiple number of files, see "3.3.5 Selecting a Multiple Number of Files ".)
2. Select the {Delete} command from the {File} menu, or select  from the tool bar.
3. Follow the displayed message.

NOTE The condition files on the controller side cannot be deleted.

3.3.5 Selecting a Multiple Number of Files

A multiple number of files can be selected in the following five manners.

(a) Drag and drop the mouse in the file list, and a selecting frame appears. Enclose the files to be selected with the frame. Thus, the files in the frame are selected.
(b) First, select one file, then pressing the [SHIFT] key, click the last file of the range to be selected. Thus, the files in the range up to the last file are selected.
(c) Select a file, then pressing the [CTRL] key, click the file to be selected. Thus, the selected file is added one by one.
(d) To select a range of files as in (b) without using a mouse, move the cursor key up, down, left, and right, pressing the [SHIFT] key.
(e) To select an individual file as in (c) without using a mouse, move the cursor key to the file to be selected and press the space key, pressing the [CTRL] key.

3.3.6 Displaying File Contents

■ Displaying File Contents

The contents of a file can be displayed in the following two manners. Only the files corresponding to the robot can be displayed.

 Icon for robot-related file

 Icon for other files

(a) Double-click on the selected file to view contents. The contents of the file will be displayed in the Windows Notepad.
(b) Select the file to be displayed from the personal computer file list. Selecting the {Display File Contents} command from the {File} menu, or selecting the  button from the tool bar displays the contents of the file on the controller side. While the contents of one file are displayed, another file may be selected and its contents displayed in a

new dislay. Pressing the  button again returns to the previous display.

NOTE Only the contents of the files on the personal computer side can be displayed, but not the files on the controller side.

■ Display of the job header information

To display job header information, mark the check box of [Display/Print Job Header] in the [General] command displayed by selecting the [Setup] dialog box from the {Option} menu.

3.3.7 File Type Setting

Select a file type from the [File Type] combo box displayed on the personal computer side.

3.3.8 Print Out

1. Select the file to be printed from the file list on the personal computer side. Then select the {Display File Contents} command from the {File} menu, or select the  button to display the file contents on the controller side.
2. Selecting the {Print File} from the {File} menu, or selecting the  button on the controller side prints out the file.

NOTE
- To print out the file content with the job header, select the [Display/Print Job Header] in the [Setup] dialog box of the {Option} menu in advance.
- The file contents of the personal computer side can be printed out, but not the file contents of the robot controller side.

3.3.9 Batch Processing

Using a 'job-list file' created by an editor such as Notepad, a multiple number of files can be uploaded or downloaded.
The following is the required format of the list file.

<Job List Format>
- One file name must be described per line.
- Both capital or lowercase characters can be used.
- Files with no extensions are shown as "JBI."
- When there are blank lines in the middle of the list, these blank lines are skipped and ignored.

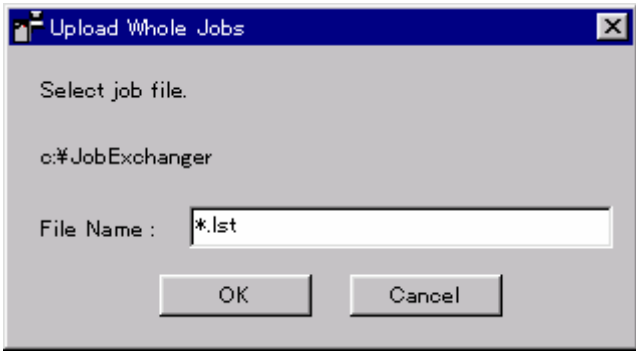
<Example of a job list File>

```
1.JBI
```

2.JBI
: (When an extension is omitted, it is regarded as JBI.)
:
5.JBI

■ Upload Batch Jobs

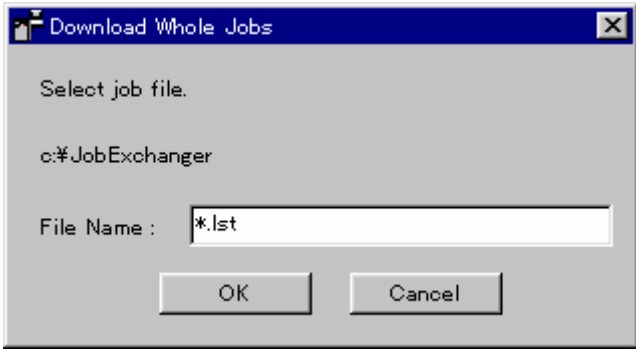
1. Select the {Upload Batch Jobs} command from the {Option} menu to display the [Upload Whole Jobs] dialog box. Input the file name of the list of jobs in the currently selected folder.



2. Click the [OK] button to start uploading the files in the list one by one in order.

■ Download Batch Jobs

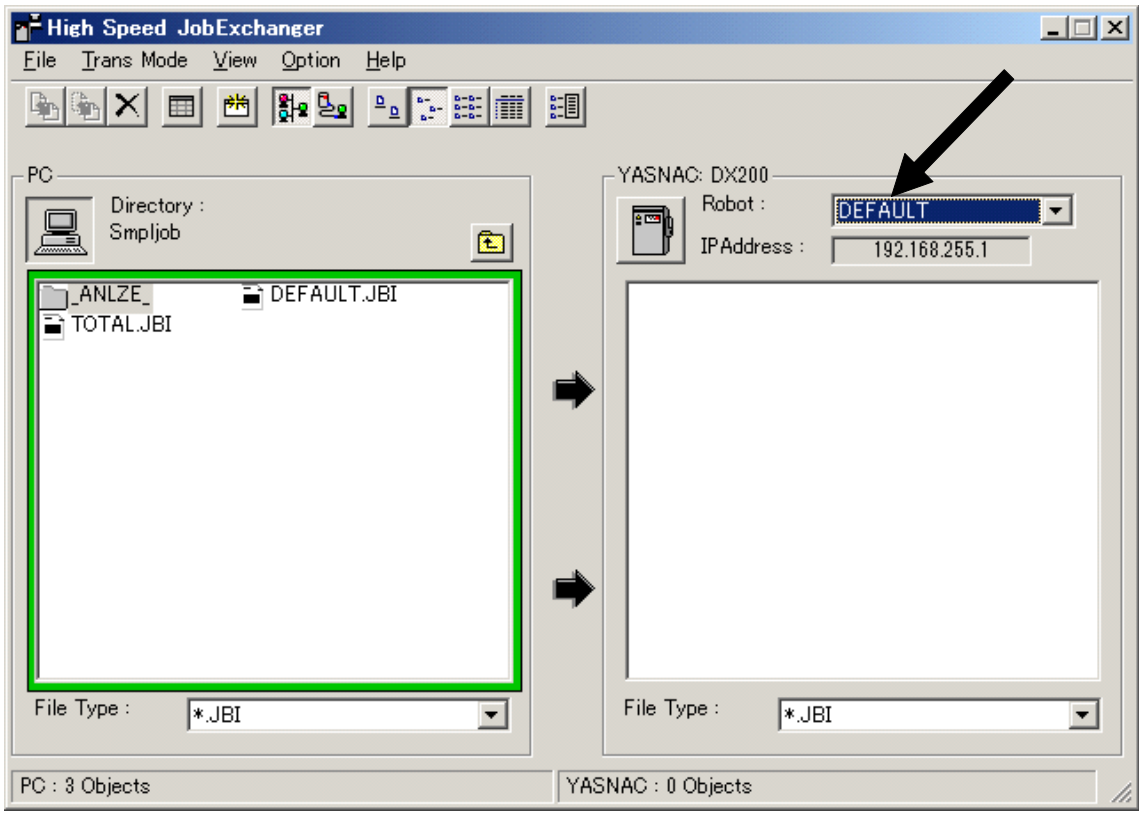
1. Select the {Download Batch Jobs} command from the {Option} menu to display the [Download Whole Jobs] dialog box. Input the file name of the list of jobs in the currently selected folder.



2. Click the [OK] button to start downloading the files in the list one by one in order.

3.3.10 Switching the Target Robot

For Ethernet communication, select the robot name from the [Robot] combo box in the main display. The target robot is switched.



NOTE
- Switching the target robot is available only for Ethernet communication.
- When the robot is switched, the file list of robots in the window is not updated automatically. Click the {Refresh} command from the {View} menu on the controller side before performing any file operations.

3.3.11 Transmission Files

The following files can be transmitted.
However, "I/O Data File(.dat .lst)," "Customer Data File (dat .sys)" and "Parameter File (.prm)" cannot be downloaded to the robot controller nor deleted.

Job File (.jbi .jbr)
Condition Data File (.cnd)
General-purpose Data File (.dat)
I/O Data File (.dat .lst) (*)
Customer Data File (.dat .sys) (*)
Parameter File (.prm) (*)

NOTE
- Loading of the system parameter to the robot controller is disabled for safety reasons. (Use FC1, FC2 or FC1 Emulator for Windows.)
- When the robot controller does not support the Ethernet function, transmission of the files marked with (*) is not possible.

3.3.12 INI Files

The High Speed JobExchanger stores the necessary environmental information in 2 INI files:

JobExchanger.ini	Information about Font, Start-Language, Window-size, etc.
robot.ini	Information about the registration of robots.

The INI files are stored in a difference storage location by OS which installed MOTOCOM32.
The INI files are stored in the following places, respectively.

Windows XP
Both INI files are stored in a High Speed JobExchanger installation folder.

Windows 7 / 10

JobExchanger.ini	The INI file is stored in a "C:\Users\<username>\Documents\MOTOMAN\MOTOCOM32\HighSpeedJobExchanger".
robot.ini	The INI file is stored in a "C:\ProgramData\Motoman\MOTOCOM32\HighSpeedJobExchanger\".

3.3.13 Language Files

The High Speed JobExchanger supports two languages: English and Japanese.
A language file can be edited, and a new language file can also be created. For details, refer to " 3.4 Editing of Language Files ".

3.4 Editing of Language Files

3.4.1 Editing Language Files

In the "Select Language" dialog box, press the [Edit] button. The language file of the currently selected language is opened in the Windows Notepad. Using Notepad, the language file can be edited.

NOTE Since the language files are important parts of the software, extreme care must be taken during modification.

3.4.2 Creating New Language Files

Modification can be carried out from the Use Windows Notepad to make modifications.
If you want to add other language which is not yet available, follow the instructions below:
The language files are stored in a difference storage location by OS which installed MOTOCOM32. The language files are stored in the following places, respectively.

Windows XP
The language files are stored in a High Speed JobExchanger installation folder.
Windows 7 / 10
The language files are stored in a "C:\ProgramData\Motoman\MOTOCOM32\HighSpeed-JobExchanger\".

<Example : Chinese>

1. Copy the "English.ing" file, then rename it "User13.ing."

NOTE Do not rename it "Chinese.ing.

2. In the "User13.ing" file, change "English" to "Chinese" and place the Chinese language translation of each word in the file.
3. If you want to add more languages, continue with "User14.ing" and so on.

NOTE The new languages are not displayed under the "System" language. By selecting "User13," the Chinese language appears.

4 OPERATION OF HOST CONTROL FUNCTION

4.1 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

For details, refer to " 4.3 Robot Control Function " " 4.4 Read/Write of I/O Signals " in the following section.

4.2 Startup and Exit

■ Startup

To start up the [Host Control 32], click the [Start] button in the task bar and point to [Program] and select [Motoman], [MOTOCOM 32], and then [Host Control] (In the case of Windows10, [Motoman] then [Host Control]). The [Select Language] display appears. Select a desired language and click the [OK] button. The language can be changed at any time from the main menu of this application.

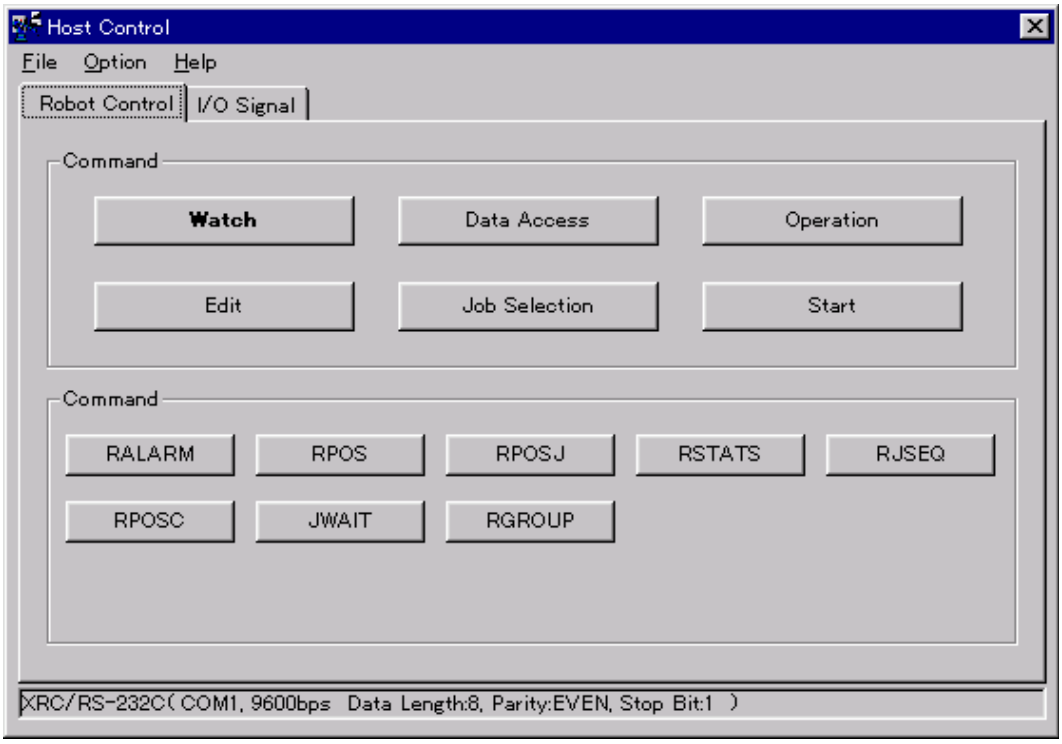
■ Exit

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

4.3 Robot Control Function

This function reads the robot 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. (The robot must be set to remote mode.)

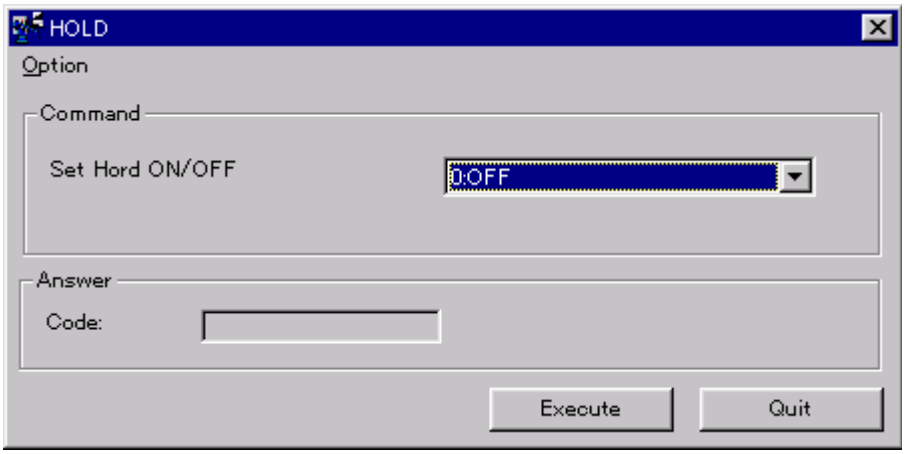
1. Select [Host Control] and the [Host Control] Display appears.



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

NOTE
- With the ERC, some commands are grayed and are not available for selection.
- With the YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, if the Ethernet Server function is used some of the commands may not be used.

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.
   Transmission commands are as follows. For details, refer to the robot controller Data Transmission Operator's Manual.
   - “YRC1000 OPTIONS INSTRUCTIONS FOR DATA TRANSMISSION FUNCTION”
   - “YRC1000micro 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”
   - “NX100 OPTIONS INSTRUCTIONS FOR DATA TRANSMISSION FUNCTION”
   - “YASNAC XRC OPTIONS INSTRUCTIONS FOR DATA TRANSMISSION FUNCTION”
   - “YASNAC MRC OPTIONS INSTRUCTIONS FOR DATA TRANSMISSION FUNCTION”
   - "YASNAC ERC Data Transmission Function, User Manual"

.

			ERC	MRC	XRC	NX100/DX100/FS100/DX200 YRC1000/YRC1000micro
Reading status	Read, monitoring system	RALARM	○	○	○	○
		RPOS *2	○	○	-	-
		RPOSJ	○	○	○	○
		RSTATS	○	○	○	○
		RJSEQ	○	○	○	○
		RPOSC	○	○	○	○
		JWAIT	○	○	○	○
		RGROUP *1	-	○	○	○
		RALARMS *3	-	-	-	○
	Read, data access system	RJDIR	○	○	○	○
		RUFRAME	○	○	○	○
		UPLOAD	○	○	○	○
		SAVEV *1	-	○	○	○
		SAVEVP *3	-	-	-	○
Control of system	Operation system	HOLD	○	○	○	○
		RESET	○	○	○	○
		CANCEL	○	○	○	○
		MODE *1	-	○	○	○
		CYCLE	○	○	○	○
		HLOCK	○	○	○	○
		MDSP	○	○	○	○
		SVON	○	○	○	○
		CGROUP *1	-	○	○	○
		CTASK *1	-	○	○	○
	Editing system	DELETE	○	○	○	○
		WUFRAME	○	○	○	○
		CVTRJ	○	○	○	○
		DOWNLOAD	○	○	○	○
		CVTSJ *1	-	○	○	○
		LOADV *1	-	○	○	○
		LOADVP *3	-	-	-	○
	Job selection system	SETMJ	○	○	○	○
		JSEQ	○	○	○	○
	Startup system	START	○	○	○	○
		MOVJ	○	○	○	○
		MOVL	○	○	○	○
		IMOV	○	○	○	○
		PMOVJ	○	○	○	○
		PMOVL	○	○	○	○

1 Commands validated only with the YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC and MRC2

2 Commands validated only with the MRC ,MRC2, ERC and ERC2

3 Commands validated only with the YRC1000, YRC1000micro, DX200, DX100, FS100and NX100

4.4 Read/Write of I/O Signals

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

NOTE The robot must be set to remote mode.

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

MRC

Signal	Range	Name	Read	Write
0xxx	0010-0167 (128)	Robot universal input	○	×
1xxx	1010-1167 (128)	Robot universal output	○	×
2xxx	2010-2187 (144)	External input	○	×
3xxx	3010-3187 (144)	External output	○	×
4xxx	4010-4167 (128)	Robot specific input	○	×
5xxx	5010-5247 (192)	Robot specific output	○	×
6xxx	6010-6047 (32)	Timer/counter	×	×
7xxx	7010-7327 (256)	Auxiliary relay	○	×
8xxx	8010-8087 (64)	Control status signal	○	×
82xx	8210-8247 (32)	Pseudo input signal	○	×
9xxx	9010-9167 (128)	Network input	○	○

XRC

Signal	Range	Name	Read	Write
0xxx	0010-0247 (192)	Robot universal input	○	×
1xxx	1010-1247 (192)	Robot universal output	○	×
2xxx	2010-2327 (256)	External input	○	×
3xxx	3010-3327 (256)	External output	○	×
4xxx	4010-4287 (224)	Robot specific input	○	×
5xxx	5010-5387 (304)	Robot specific output	○	×
6xxx	-	-	×	×
7xxx	7010-7887 (704)	Auxiliary relay	○	×
8xxx	8010-8127 (96)	Control status signal	○	×
82xx	8210-8247 (32)	Pseudo input signal	○	×
9xxx	9010-9167 (128)	Network input	○	○

NX100

Signal	Range	Name	Read	Write
0xxx	00010-01287 (1024)	Robot universal input	○	×
1xxx	10010-11287 (1024)	Robot universal output	○	×
2xxx	20010-21287 (1024)	External input	○	×
22xx	22010-23287 (1024)	Network input	○	○
3xxx	30010-31287 (1024)	External output	○	×
32xx	32010-33287 (1024)	Network input	○	×
4xxx	40010-40807 (640)	Robot specific input (System)	○	×
5xxx	50010-51007 (800)	Robot specific output (System)	○	×
6xxx	-	-	×	×
7xxx	70010-79997 (7992)	Auxiliary relay (System)	○	×
8xxx	80010-80647 (512)	Control status signal (System)	○	×
82xx	82010-82127 (96)	Pseudo input signal (System)	○	×

DX100

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

FS100

Signal	Range	Name	Read	Write
0xxx	00010-01287(1024)	Robot universal input	○	×
1xxx	10010-11287(1024)	Robot universal output	○	×
2xxx	20010-21287(1024)	External input	○	×
25xx	25010-26287(1024)	Network input	○	○
3xxx	30010-31287(1024)	External output	○	×
35xx	35010-36287(1024)	Network input	○	×
4xxx	40010-41607(1280)	Robot specific input (System)	○	×
5xxx	50010-52007 (1600)	Robot specific output (System)	○	×
6xxx	-	-	×	×
7xxx	70010-79997 (7992)	Auxiliary relay (System)	○	×
8xxx	80010-80647 (512)	Control status signal (System)	○	×
82xx	82010-82207(160)	Pseudo input signal (System)	○	×

DX200

Signal	Range	Name	Read	Write
0xxx	00010-05127(4096)	Robot universal input	○	×
1xxx	10010-15127(4096)	Robot universal output	○	×
2xxx	20010-25127(4096)	External input	○	×
27xx	27010-29567(2048)	Network input	○	○
3xxx	30010-35127(4096)	External output	○	×
37xx	37010-39567(2048)	Network input	○	×
4xxx	40010-41607(1280)	Robot specific input (System)	○	×
5xxx	50010-53007 (2400)	Robot specific output (System)	○	×
6xxx	-	-	×	×
7xxx	70010-79997 (7992)	Auxiliary relay (System)	○	×
8xxx	80010-80647 (512)	Control status signal (System)	○	×
82xx	82010-82207(160)	Pseudo input signal (System)	○	×

YRC1000

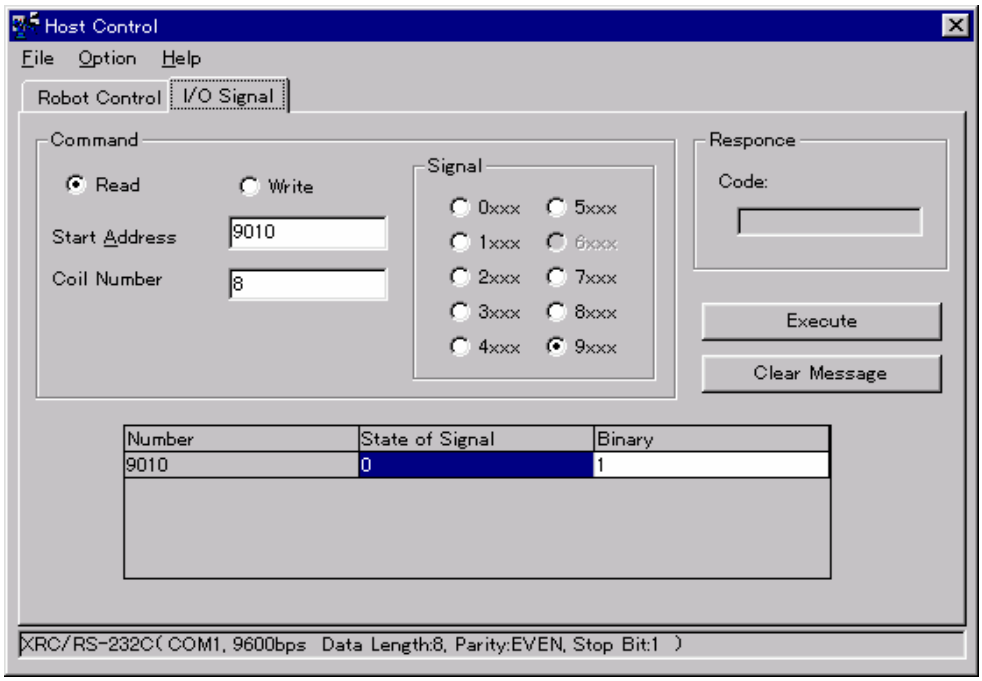
Signal	Range	Name	Read	Write
0xxx	00010-05127(4096)	Robot universal input	○	×
1xxx	10010-15127(4096)	Robot universal output	○	×
2xxx	20010-25127(4096)	External input	○	×
27xx	27010-29567(2048)	Network input	○	○
3xxx	30010-35127(4096)	External output	○	×
37xx	37010-39567(2048)	Network input	○	×
4xxx	40010-42567(2048)	Robot specific input (System)	○	×
5xxx	50010-55127 (4096)	Robot specific output (System)	○	×
6xxx	-	-	×	×
7xxx	70010-79997 (7992)	Auxiliary relay (System)	○	×
8xxx	80010-85127 (4096)	Control status signal (System)	○	×
87xx	87010-87207(160)	Pseudo input signal (System)	○	×

YRC1000micro

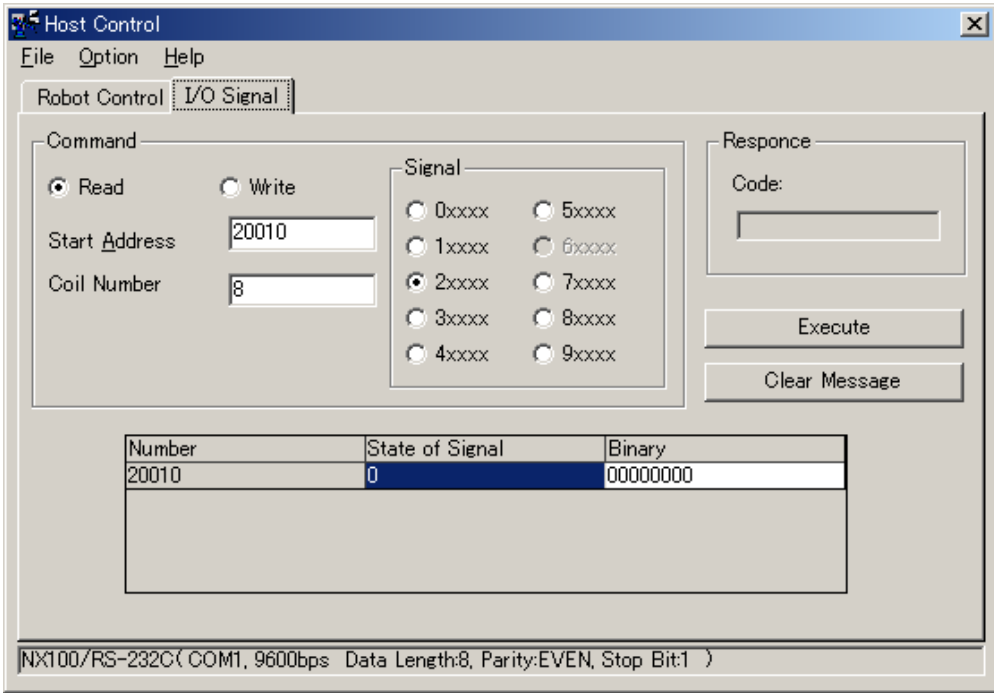
Signal	Range	Name	Read	Write
0xxx	00010-05127(4096)	Robot universal input	○	×
1xxx	10010-15127(4096)	Robot universal output	○	×
2xxx	20010-21287(1024)	External input	○	×
27xx	27010-29567(2048)	Network input	○	○
3xxx	30010-31287(1024)	External output	○	×
37xx	37010-39567(2048)	Network input	○	×
4xxx	40010-42567(2048)	Robot specific input (System)	○	×
5xxx	50010-55127 (4096)	Robot specific output (System)	○	×
6xxx	-	-	×	×
7xxx	70010-79997 (7992)	Auxiliary relay (System)	○	×
8xxx	80010-85127 (4096)	Control status signal (System)	○	×
87xx	87010-87207(160)	Pseudo input signal (System)	○	×

■ I/O signal read/write display

XRC, MRC



YRC1000, YRC1000micro, DX200, DX100, FS100, NX100



To write the I/O signals, select [Write] in the command, and click the column to be edited from the signal list. Then the I/O editing display appears. Input the signal status to be changed.

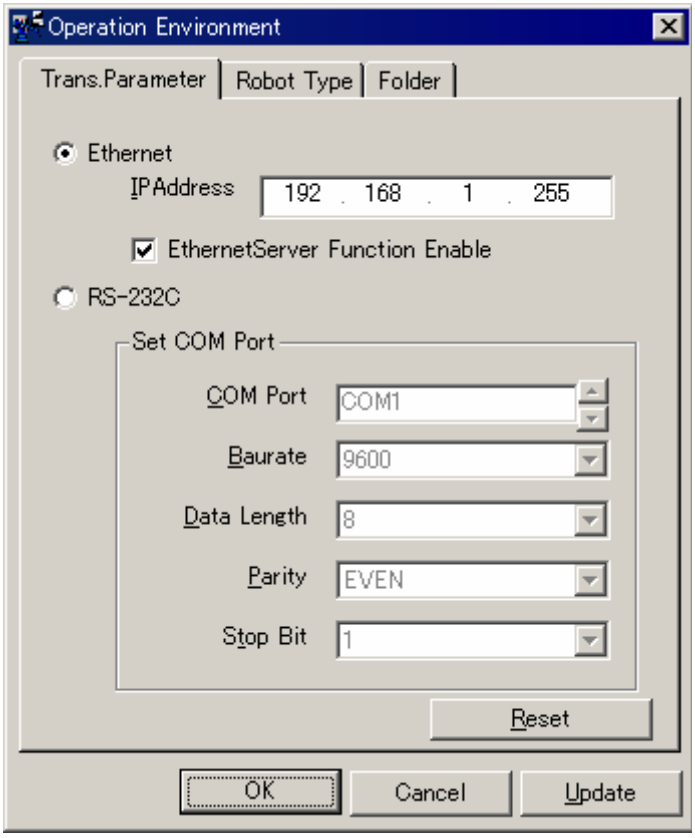
NOTE This function is valid only for the MRC V4.111 or after, or for the XRC, or for the NX100, or for the DX100, or for the FS100, or for the DX200, or for the YRC1000, or for the YRC1000micro.

4.5 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

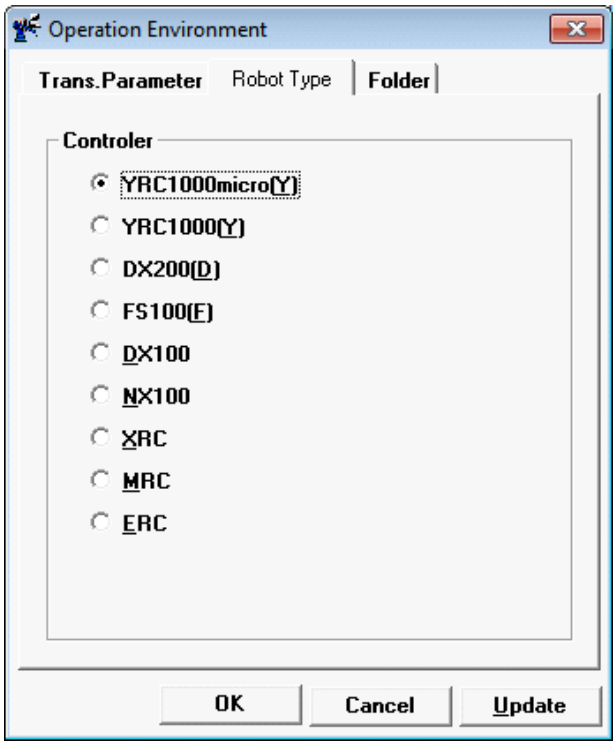
Select the controller and communications protocol and choose their settings.
There are two types of communications; the Ethernet and RS-232C transmission. Click the radio button to select one.



Ethernet		Only the IP address can be set for the Ethernet transmission. Enter the IP address assigned to the controller connected to the network.
	EthernetServer Function Enable	Displayed when the controller is set to YRC1000, YRC1000micro, DX200, DX100, FS100, NX100. When the controller has the Ethernet Server function is enabled, check this box to enable the communication using the Ethernet Server function.
RS-232C		Select the port number, baud rate, data length, parity, and stop bit according to the controller settings. Click the [Reset] button to set to the default data.

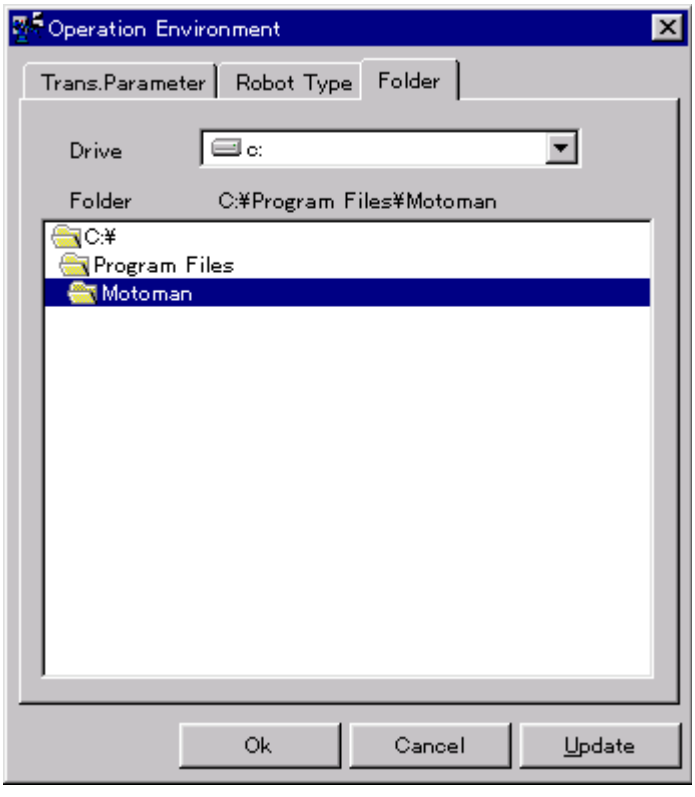
Setting the robot model

Select the controller model for transmission in the display.



Selecting a folder

Select the folder for communications in the display.
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 such as DOWNLOAD, UPLOAD, etc.



4.6 Language Selection

Change the language displayed in the Host Control according to the environment.
Select the [Language] command from the [Option] menu to display the [Select Language] dialog box. The language is selected.

Ignore language selection at program start	Click the box [Ignore language selection at program start], and the [Select Language] display does not appear when starting the Host Control software. Clicking the box again removes the check mark and displays the [Select Language] window when starting.

NOTE Some languages may not be correctly displayed unless fonts are set correctly.

4.7 Version Information

Displays the Host Control version information.

5 AUTO JOB CHANGER OPERATION

5.1 AUTO JOB CHANGER OPERATION

Using DCI (Data Communication by Instruction) transmitting function, the job corresponding to the work number, which is required to be sent from the actual robot, is transmitted.
If the robot memory runs short because there are too many types of workpieces or a job becomes excessively long due to a complicated job, a personal computer can be used as an external memory. In such a case, the work number is transmitted from the robot side to the personal computer side, and the required job is uploaded to execute the loaded job.
Assuming that the robot memory runs short, all work jobs to be uploaded from the personal computer are loaded with their name changed to the same name (WORKJOB).
In this application, operation is performed continuously from work numbers 1 to 5.
(Work jobs can be added when necessary.)

5.2 Before Execution of "Automatic Operation"

In order to execute automatic operation of Auto Job Changer, the following job is needed for the robot side. (This job must be positioned as the master job for Auto Job Changer at the robot side.) This job and the jobs from No. 1 to No. 5 have been already registered at the sample robot in the personal computer when the MOTOCOM 32 is installed. Before execution of Auto Job Changer, use the host control to transmit this job to the robot side.

Job name: ROBJOB

Job contents:

```
NOP
'DATA INITIALIZE
SET I000 0
'
*START
TIMER T=5.00
'DATA SET
INC I000
'
'DCI INSTRUCTION
SAVEV I000
TIMER T=1.00
LOADJ JOB:WORKJOB JBI
CALL JOB:WORKJOB
JUMP *START IF I000<5
'
END
```

5.3 Startup and Exit

■ Startup

To start up the [Auto Job Changer], click the [Start] button and point to [Program], and select [Motoman], [MOTOCOM32], and then [Auto Job Changer] (In the case of Windows10, [Motoman] then [Auto Job Changer]). The [Select Language] display appears. Select a desired language and click the [OK] button. The language can be changed at any time from the main menu of this application.

■ Exit

Select the {Exit} command from the {File} menu, and the Auto Job Changer is ended.

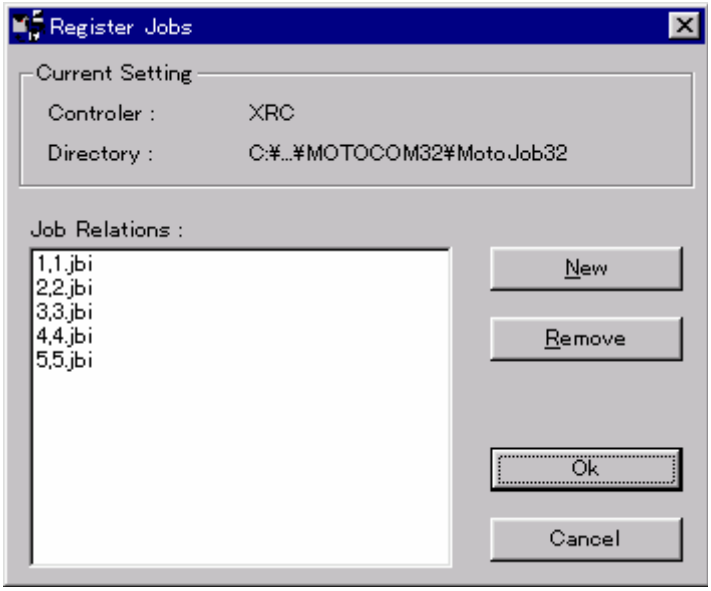
5.4 Operation Procedure

5.4.1 Register Job

Work numbers and their corresponding job names must be registered for each robot. Register them as described below.

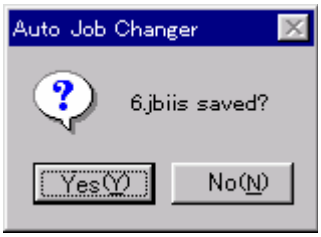
■ New registration

1. Press the [New] button in the [Register Jobs] display.



2. When the [New Job Registration] display appears, input the work number and registered job name (including the extension), and press the [Execute] button.

3. When the registered job already exists in the current robot directory, it can be coordinated with the job. (An inquiry is given by the program.) If not, the registered job is received from the controller. Press the [Yes] or [No] button.



4. Pressing the [OK] button in the [Register Jobs] display registers all information that has been newly registered. Pressing the [Cancel] button cancels all the registering procedures.

■ Deletion

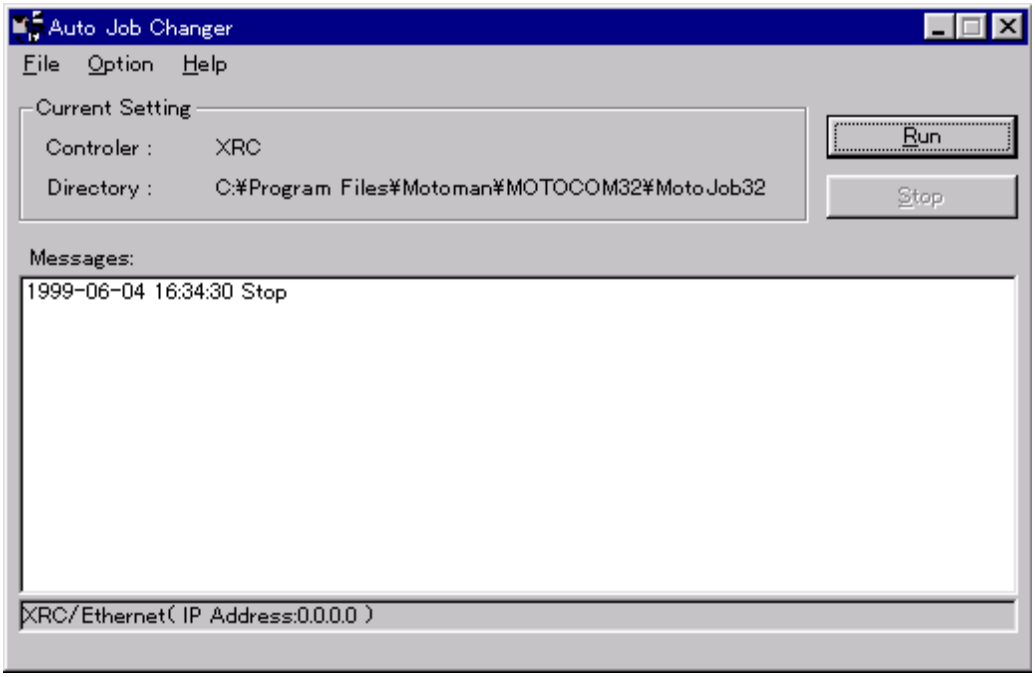
1. From the [Job Relations] list in the Register job display, select the job to be deleted. (More than one job can be selected.)
2. Press the [Delete] button.
3. Pressing the [OK] button in the [Register jobs] display registers of all the deleted information. Pressing the [Cancel] button cancels all the registering procedures.

5.4.2 Automatic operation

Pressing the automatic operation button makes the personal computer waiting for the work number to be received from the controller which is currently selected. The following sequence is repeated until the cancel button is pressed.

1. When the work number is received from the controller, the job registered for that number in advance is copied to a file with a name of WORKJOB.
2. The computer waits for a request to receive WORKJOB from the controller.
3. The computer sends the job when it receives the request and enters the work number waiting status again.

NOTE During automatic operation, the grayed functions in the display such as robot change cannot be used.



5.4.3 Cancel

When an automatic operation is terminated, press the [STOP] button from the [Auto Job Changer] dialog box.
Normally, stop the DCI operation first from the controller side. If the personal computer is terminated first, an alarm may occur on the controller side.

NOTE It may take some time until the stopped status is actually entered after the cancel button is pressed.

5.4.4 Display Log File

Select the {Display Logging File} command from the {Option} menu. Displays the contents of a log file.

5.4.5 Deletion Message

Select the {Clear Messages} command from the {Option} menu. Deletes an automatic operation message.
Messages are deleted periodically by the program.

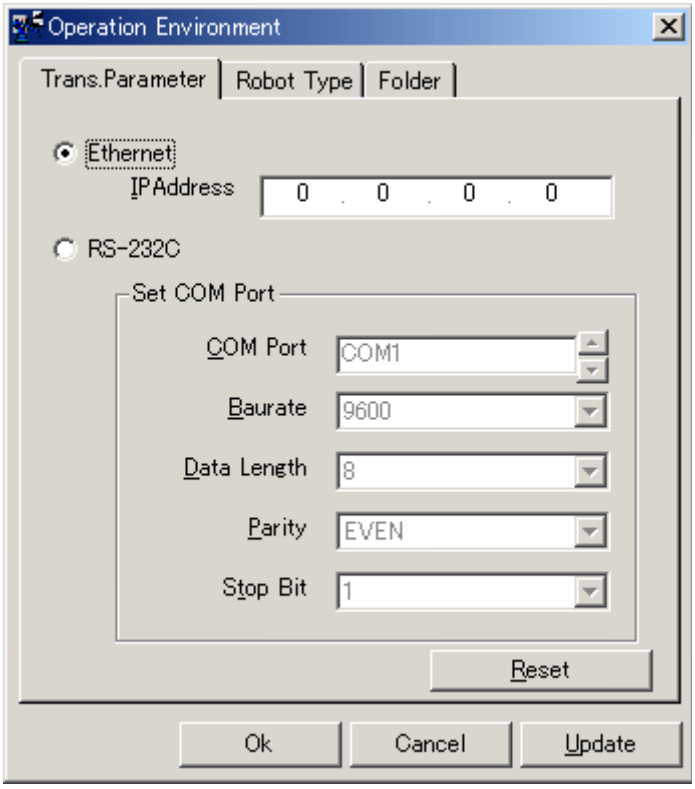
5.5 Environmental Settings

5.5.1 Environmental Settings

The environmental settings define the operations of the Auto Job Changer.
Select {Option} - {Operation Environment}. Each item of the [Operation Environment] dialog box is set.

Setting transmission parameters

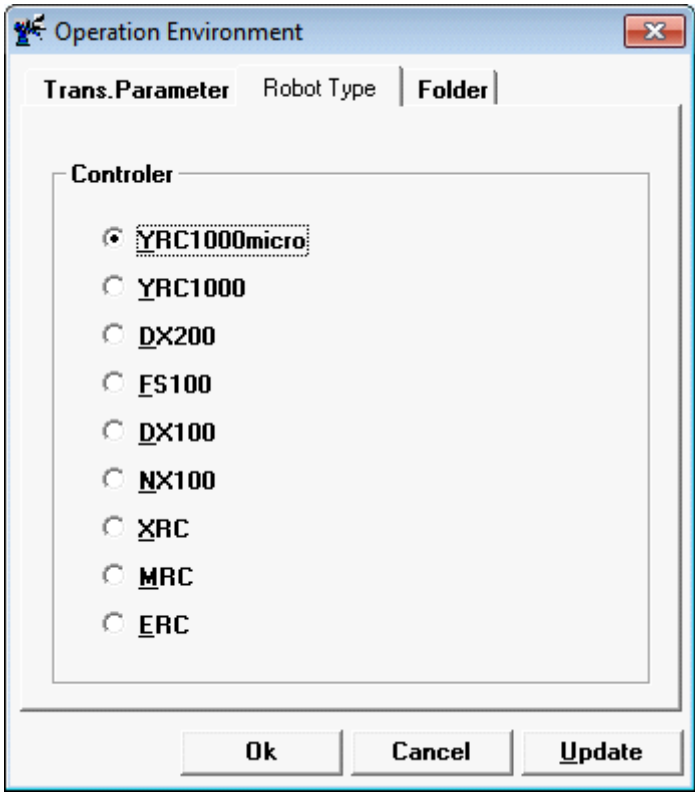
Select the controller and communications protocol and choose their settings.
There are two types of communications; the Ethernet and RS-232C transmission. Click the ratio button to select one.



Ethernet	Only the IP address can be set for the Ethernet transmission. Enter the IP address assigned to the controller connected to the network.
RS-232C	Select the port number, baud rate, data length, parity, and stop bit according to the controller settings. Click the [Reset] button to set to the default data.

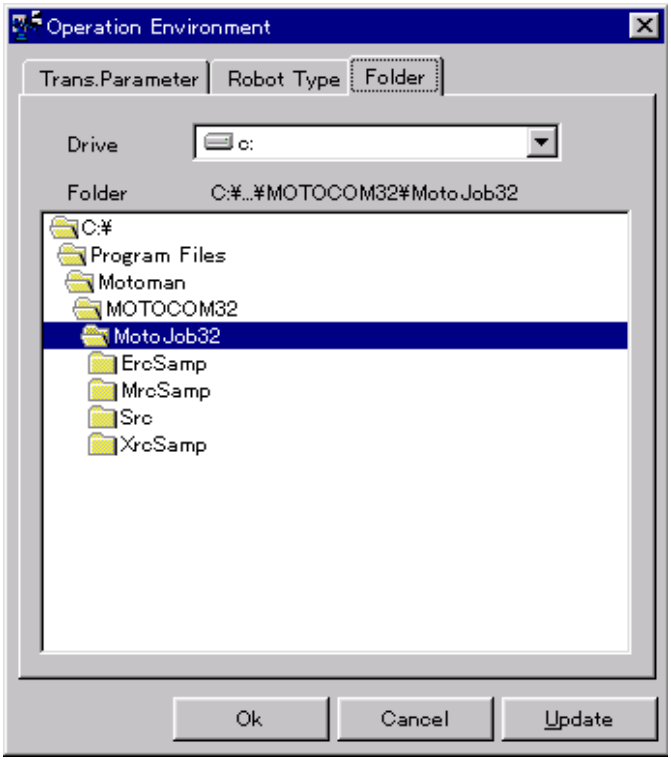
Selecting the robot model

Select the controller model for transmission in the display.



Selecting a folder

Select the folder for communications in the display.
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 such as DOWNLOAD, UPLOAD, etc.



5.5.2 Automatic Operation Startup

Select the {Auto Run} command from the {Option} menu. Determines whether automatic operation is to be started at the same time as startup of Auto Job Changer. A check mark at the left side of the menu means that automatic operation is going to start.

5.5.3 Take Log File

Select the {Write Logging Message} command from the {Option} menu. Determines whether a communication log is to be taken at automatic operation. A check mark at the left side of the menu means that the log is going to be taken.
A log file is created in the directory where the MOTOCOM32 is installed with the name of "ONLINE.LOG." The program is overwritten when this file exceeds a certain size, however, the log file from one generation before is kept in the file named "ONLINE.LO$."

5.6 Language Selection

Change the language displayed in the Auto Job Changer according to the environment. Select the [Select Language] command from the [Option] menu to display the [Select Language] dialog box. The language is selected.

Ignore language selection at program start	Click the box [Ignore language selection at program start], and the [Select Language] display does not appear when starting the Auto Job Changer. Clicking the box again removes the check mark and displays the [Select Language] window when starting.

NOTE Some languages may not be correctly displayed unless fonts are set correctly.

5.7 Version Information

Displays the Auto Job Changer version information.

6 CREATING A TRANSMISSION APPLICATION

6.1 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 Basic" and "Visual C++" "Visual C#") which employs a data transmission function (MS-Windows DLL file type: file name: MOTOCOM32.DLL). (Other languages can also be used.)

The program list of the sample program is in the “MOTOMAN\MOTOCOM32\MOTOCOM32DLL\Sample” folder below "Public Documents" folder. And, the program list of the autojob change software is in the “MOTOMAN\MOTOCOM32\MotoJob32\Src” folder.

NOTE
- Execute the sample program in the MOTOCOM32 installation directory.
- YASKAWA is not responsible for anything that may result from using the sample program

6.2 Using Visual Basic

6.2.1 Preparation

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

(1) Microsoft Windows 7 / 10 *1

(2) Visual Basic Ver6.0 or more *2

*1 MS Windows 7 / 10 is a registered trademark of Microsoft Corporation, U.S.A.

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

6.2.2 How to Create a transmission application

This paragraph explains a simple program, as an example, which sends/receives a job that was input to the text box to/from the controller.

■ Creation of Code Module

In order to call up "Motocom32. DLL" from Visual Basic, declaration of the Motocom 32. DLL I/F functions to be used is needed. The following two methods are available.

Write the declaration of the DLL functions yourself.

Use the definition file attached to the Motocom 32 package.

Write the declaration of the DLL functions yourself.

Add the code module and write the following declaration in the declaration area.

```
Declare Function BscOpen Lib "MotoCom32" Alias "_BscOpen@8" _
        (ByVal Path As String, ByVal mode As Integer) As Integer
Declare Function BscClose Lib "MotoCom32" Alias "_BscClose@4" _
        (ByVal nCid As Integer) As Integer
Declare Function BscSetCom Lib "MotoCom32" Alias "_BscSetCom@24" _
        (ByVal nCid As Integer, ByVal Port As Integer, ByVal Baud As Long, ByVal
        Parity As Integer, ByVal clen As Integer, ByVal stp As Integer) As Integer
Declare Function BscSetEther Lib "MotoCom32" Alias "_BscSetEther@16" _
        (ByVal nCid%, ByVal IPaddr$, ByVal mode%, ByVal hWnd&) As Integer
Declare Function BscConnect Lib "MotoCom32" Alias "_BscConnect@4" _
        (ByVal nCid As Integer) As Integer
Declare Function BscDisConnect Lib "MotoCom32" Alias "_BscDisConnect@4" _
        (ByVal nCid As Integer) As Integer
Declare Function BscDownLoad Lib "motocom32.dll" Alias "_BscDownLoad@8" _
        (ByVal nCid As Integer, ByVal fName As String) As Integer
Declare Function BscUpLoad Lib "motocom32.dll" Alias "_BscUpLoad@8" _
        (ByVal nCid As Integer, ByVal fName As String) As Integer
```

Define the followings as the parameters for BscOpen() to select the type of connection line.

```
Public Const PACKETCOM = (1)
Public Const PACKETETHERNET = (16)
```

Use the definition file attached to the Motocom32 package.

The "Motocom32.DLL" package includes a "Motocom32.BAS" file that defines the DLL I/F functions. Add this file to the Visual Basic project.

(1) Copy the "Motocom32.BAS" file to the source directory of the application to be created.
(2) Click "Project" and specify the "Motocom 32.BAS" from the "File" menu to add it to the project.

Create a sub-module to open/close the following ports.

```
Function Ms_BscOpenComm(TransMode%, FuncMode%)
Function Ms_BscCloseComm()
```

" Function Ms_BscOpenComm() " to select the data part (program list) of the above function. Use "Copy" to copy this section to Ms_BscOpenComm () function. Repeat for Ms_BscCloseComm(" Function Ms_BscCloseComm() ").

■ Creation of Form Module

Create the following module.

(1) Form to be program display
On this form, create the following controls.
(2) Text Box to input the job name (control name: "TxtJobName", text name: 0.JBI")
(3) Send button (control name:"CmdDownLoad", caption name: "Send")
(4) Receive button (control name: "CmUpLoad", caption name: "Receive")
(5) Exit button (control name: "CmdExit", caption name: "Exit")

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

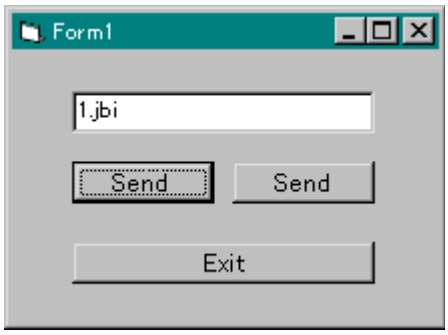
```
Sub CmdDownload_Click ()
Sub CmdUpLoad_Click ()
Sub CmdExit_Click ()
```

" Sub CmdDownload_Click () " to select the data part (program list) of the above function. Use "Copy" to copy this section to CmdDownload_Click () function. Repeat for CmdUpLoad(" Sub CmdUpLoad_Click () ") and CmExit(" Sub CmdExit_Click () ").

■ Creation and Execution of EXE File

Select "EXE file creation" from the Visual Basic file menu to create an execution enabled module. By putting this module in the same directory as the job to be sent or received and execut-

ing it, the job can be sent or received.



NOTE

The MOTOCOM installation directory contains data transmission functions (Windows DLL file type, **file name: Motocom32.DLL and Motolk.DLL, Motolkr.DLL, Vrp32.DLL**). When executing an application, copy the functions to the directory where the module to be executed is created. For transmission via Ethernet, copy **HslSrv32.exe** to the same directory as Motocom32.DLL.

6.3 Using Visual C++

6.3.1 Preparation

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

(1) Microsoft Windows 7 / 10 *1
(2) Visual C++ Ver6.0 or more *2

*1 MS Windows 7 / 10 is a registered trademark of Microsoft Corporation, U.S.A.
*2 Visual C++ is a registered trademark of, Microsoft Corporation U.S.A.

6.3.2 How to Create a transmission application

This paragraph explains a simple program, as an example, which sends/receives a job that was input to the text box to/from the controller.

■ 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 "motocom.h" attached to the MOTOCOM32 application using the dialog class source file (TestDig.cpp). Also include the header file, "direct.h," as the sample source as shown below.

```
#include "stdafx.h"
#include "Test.h"
#include "TestDlg.h"
#include <direct.h>        <------Add this line.
#include "motocom.h"       <------Add this line.
```

(2) Copy the "motocom32.lib" file, the "motocom.h" file and the data transmission func-

tion (Windows DLL file type, **file name: Motocom32.DLL and Motolk.DLL, Motolkr.DLL, Vrp32.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 "motocom32.lib" file in the "Object/Library Module" setting column, and click the [OK] button.

The MOTOCOM32 functions can be used in the file where "motocom.h" is included.

NOTE The library file (**file name: Motocom32.Lib**) and the included file (**file name: Motocom.h**) are in the MOTOCOM32 installation directory.

■ Editing with a Dialog Box

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

(1) Delete the [CANCEL] pushbutton which was created by default.
(2) Change the caption of the [OK] pushbutton to "Exit."
(3) Create an edit control to input the job name, and name the ID "IDC_JOBNAME."
(4) Create a pushbutton for sending, and name the caption "Send" and the ID "IDC_DOWNLOAD."
(5) Create a pushbutton for receiving, and name the caption "Receive" and the ID "IDC_UPLOAD."

■ Addition of Functions and Variables

(1) Open the TestDLg.h file to add the following function declaration.

```
short TestOpenComm( int TransMode, int FuncMode );
short TestCloseComm( short ncid );
```

(2) Create a function "CTestDlg::TestOpenComm" to open the communications port.
(3) Create a function "CTestDlg::TestCloseComm" to close the communicatinos port.
(4) Create a function "CTestDlg::OnDownload" for BN_CLICKED message in Class Wizard using the [Send] pushbutton (IDC_DOWNLOAD).
(5) Create a function "CTestDl::OnUpload" for BN_CLICKED message in Class Wizard using the [Receive] pushbutton (IDC_UPLOAD).
(6) Add variable "m_jobname" in Class Wizard by Cedit type for inputting characters of the job name edit control (IDC_JOBNAME).
After adding the functions, write the code in each function.

```
CTestDlg::TestOpenComm function
CTestDlg::TestCloseComm function
CTestDlg::OnDownload function
CTestDlg::OnUpload function
```

In TestOpenComm(), the storage passing of IP address (IPAddress), the communication mode (mode), and the application (cur_dir) is specified. Please change according to cus-

tomer's environment.

" CTestDlg::TestOpenComm function " to select the data part (program list) of the above function. Use "Copy" to copy this section to CTestDlg::TestOpenComm() function. Repeat for CTestDlg::TestCloseComm(" CTestDlg::TestCloseComm function "), CTestDlg::OnDownload(" CTestDlg::OnDownload function "), and CTestDlg::OnUpload(" CTestDlg::OnUpload function ").

■ Creation and Execution of EXE File

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

NOTE

The MOTOCOM installation directory contains data transmission functions (Windows DLL file type, **file name: Motocom32.DLL and Motolk.DLL, Motolkr.DLL, Vrp32.DLL**). When executing an application, copy the functions to the directory where the module to be executed is created. For transmission via Ethernet, copy **HslSrv32.exe** to the same directory as Motocom32.DLL.

6.4 Using Visual C#

6.4.1 Preparation

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

(1) Microsoft Windows 7 / 10 *1

(2) Visual Studio 2012, .NETFramework 4.0 or more *2

*1 MS Windows 7 / 10 is a registered trademark of Microsoft Corporation, U.S.A.

*2 Visual Studio 2012, .NETFramework is a registered trademark of, Microsoft Corporation U.S.A.

6.4.2 How to Create a transmission application

This paragraph explains a simple program, as an example, which sends/receives a job that was input to the text box to/from the controller.

■ 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 "Motocom32.DLL" in Visual Studio C#, "Motocom32CS.DLL" must be referenced. Select "Add Reference" from "Project" menu to display the "Add Reference" display. Then select "Reference" tag, and select "Motocom32CS.DLL" in the "MOTOCOM32DLL" folder located in the installed folder of "Motocom32", and "Motocom32CS.DLL" is added to the project.

To import the Data types defined in the namespace of "Motocom32CS.DLL", descript the following using directive.

```
using MotoCom32CS;
```

And, create the following methods to open/close the communications port.

```
private short Ms_BscOpenComm( int mode )
private short Ms_BscCloseComm( short nCid )
```

" private short Ms_BscOpenComm() " to select the data part (program list) of the above function. Use "Copy" to copy this section to Ms_BscOpenComm function. Repeat for Ms_BscCloseComm(" private short Ms_BscCloseComm() ").

■ Creation of Form Module

Create the following module.

(1) Form to be program display
On this form, create the following controls.
(2) Text Box to input the job name (control name: "TxtJobName", text name: "0.JBI")
(3) Send button (control name: "CmdDownLoad", caption name: "Send")
(4) Receive button (control name: "CmdUpLoad", caption name: "Receive")
(5) Exit button (control name: "CmdExit", caption name: "Exit")

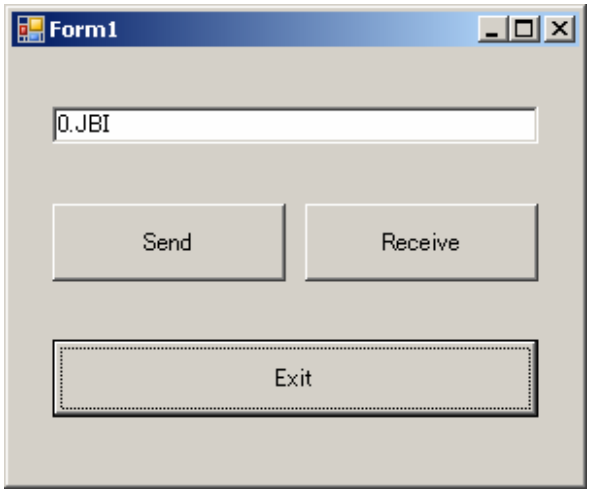
When the control is created, describe the event procedure for each button.

```
private void CmdDownLoad_Click( object sender, EventArgs e )
private void CmdUpLoad_Click( object sender, EventArgs e )
private void CmdExit_Click( object sender, EventArgs e )
```

" private void CmdDownLoad_Click() " to select the data part (program list) of the above function. Repeat for CmdUpLoad_Click(" private void CmdUpLoad_Click() "), and CmdExit_Click(" private void CmdExit_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.



NOTE The MOTOCOM installation directory contains data transmission functions (Windows DLL file type, **file name: Motocom32.DLL and Motolk.DLL, Motolkr.DLL, Vrp32.DLL**). When executing an application, copy the functions to the directory where the module to be executed is created. For transmission via Ethernet, copy **HslSrv32.exe** to the same directory as Motocom32.DLL.

6.5 Explanation of Auto Job Changer Software Creation Procedure

The creation procedure of DCI application will be descrived as follows.
Here, take for an example the procedure (procedure name: Sub DciOnline) to be called when "automatic operation" button is pressed.
Since the Auto Job Changer software is created in Visual Basic, the following description is given in the Visual Basic. However, Visual C++ or any other language can also be used.

Processing is divided into the following 5 major parts.

(1) Opening of transmission port (Function name: Ms_BscOpenComm())
(2) Receiving of job number (Function name:DciGetJobNo())
(3) Preparation for sending job (Function name:GetJobNameByNo(), JobCopy())
(4) Sending of job (Function name:DciLoadSave())
(5) Closing of transmission port (Function name:Ms_BscCloseComm())

The following describes the list of each processing.

■ Sub DciOnline

```
Sub DciOnline (CvtName As String, lst As Control, LogFile As String)
'input      CvtName      Job name to be copied
'            Lst            List name for message output
'            LogFile       Log file name
'output     None
    Dim nCid As Integer
    Dim JobNo As Integer
    Dim JobName As String
    Dim rc As Integer
    Dim Cycle As Long

    Cycle = 0
    'Get of communication handler
    nCid = Ms_BscOpenComm( mode, 1 )  ' mode=0 or 1
    If nCid >= 0 Then
     'Work No. receiving and job sending are repeated until Cancel button is pressed
     (F_QUIT flag becomes true).
        Do While Not F_QUIT
        DispLogMsg lst, "***** Waiting for work No.*****", ""
            'Receiving work No.
            If Not DciGetJobNo(nCid, JobNo, lst, LogFile) Then Exit Do
            DispLogMsg lst, "Work No.(" + Format$(JobNo) + ")reveived", LogFile
            'Fetching job name corresponding to work No.
            JobName = GetJobNameByNo(JobNo)
            If JobName = "" Then
```

```
            MsgBox"No corresponding job is registered."
            Exit Do
        End If
        'Copying corresponding job to name for sending
        If Not JobCopy(JobName, CvtName) Then
            MsgBox "Job copy disabled.(" + JobName + ")"
            Exit Do
        End If
        DispLogMsg lst, JobName + "copied to " + CvtName + "ÅD" , LogFile.
        DispLogMsg lst, "***** Waiting for request for job transmission. *****", ""
        'Sending job due to instruction from YASNAC.
        If Not DciLoadSave(nCid, lst, LogFile) Then Exit Do
        Cycle = Cycle + 1
         DispLogMsg lst, "Job has been sent.(" + Format$(Cycle) + "Circulating).",
         LogFile
    Loop

    'No. of communication handlers.
    rc = Ms_BscCloseComm(nCid)
     If rc <> 0 Then
        MsgBox "BscCloseComm terminates in fail."  + "(" + Format$(rc) + ")."
    End If
  Else
    MsgBox "Cannot open."
  End If
End Sub
```

SUPPLE-MENT

- single underline indicates functions of which program lists are described below

  Ms_BscOpenComm : " Function Ms_BscOpenComm() "
  Ms_BscCloseComm : " Function Ms_BscCloseComm() "
  DciGetJobNo : " Function DciGetJobNo "
  DciLoadSave : " Function DciLoadSave "

- DispLogMsg : Other function defined in the project.
  Outputs a message to the list box and log file.
- GetJobNameByNo : Other function defined in the project.
  Returns the number corresponding to the work number.
- JobCopy : Other function defined in the project.
  Copies a job to a specified file.

- MsgBox : Visual Basic function
  Displays a message in the dialog box and waits for the button to be pressed.
- Format$: Visual Basic function

■ Function DciGetJobNo

```
Function DciGetJobNo (nCid As Integer, JobNo As Integer, lst As Control, LogFile As
String) As Integer
'input      nCid          Communication handler
'            Lst           List name for message output
'            LogFile       Log file name
'output     JobNo         Received job No.
'return value TRUE     Completion of sending
'                FALSE    Cancel or error occurrence
    Dim rc As Integer
    Dim rc0 As Integer
    'Declaring return value of BscDciGetPos.
    ReDim axis6(5) As Double
    Dim datatype As Integer
    Dim RConf As Integer
    rc = False
    rc0 = -1
     'Request for receiving is repeated until Cancel button is pressed (F_QUIT flag
     becomes true) or work No. is received.
    Do While Not F_QUIT
        rc0 = BscDCIGetPos(nCid, datatype, RConf, axis6(0))
        If rc0 >= 0 Then Exit Do  'Work No. received.
    Loop
    If Not F_QUIT Then
         If datatype <= 2 Then 'Only byte or integer type accepted.
            'Received work No. set.
            JobNo = axis6(0)
            rc = True
        Else
             DispLogMsg lst, "Unexpected data type received. (" + Str$(datatype) + ")",
             LogFile
        End If
    Else
        DispLogMsg lst, "Canceled.", ""
    End If
    DciGetJobNo = rc
End Function
```

SUPPLE-MENT

- Double underline indicates transmission functions belonging to the MOTOCOM32.
- DispLogMsg : MOTOCOM function
  Outputs a message to the list box and log file.

■ Function DciLoadSave

```
Function DciLoadSave (nCid As Integer, lst As Control, LogFile As String) As Integer
'input      nCid            Communication handler
'           Lst             List name for message output
'           LogFile         Log file name
'output     None
'return value TRUE          Completion of sending
'              FALSE        Cancel or error occurrenceê¹
    Dim rc As Integer
    Dim rc0 As Integer
    rc = False
    'Repeated until Cancel button is pressed (F_QUIT flag becomes true) or sending is
    completed.
    Do While Not F_QUIT
        rc0 = BscDCILoadSave(nCid, 1)
        If rc0 > 0 Then 'Sending completed.
            rc = True
            Exit Do
        ElseIf rc0 = 0 Then
         'No request for receiving from YASNAC. Waiting for request for receiving again.
        Else  'Job transmission error occurs.
            MsgBox "Job transmission error occurs. (" + Format$(rc0) + ")"
            Exit Do
        End If
    Loop
    If F_QUIT = True Then
        DispLogMsg lst, "Canceled.", ""
    End If
End Function
```

SUPPLE-MENT

- Double underline indicates transmission functions belonging to the MOTOCOM32.
- DispLogMsg : Other function defined in the project.
  Outputs a message to the list box and log file.
- MsgBox : Visual Basic function
  Displays a message in the dialog box and waits for the button to be pressed.

6.6 Each Function Program List

■ Function Ms_BscOpenComm()

```
'TransMode:  0...RS-232C   1...Ethernet
'FuncMode:  0...Host Control (Client)   1...DCI/Stand Alone (Server)
Function Ms_BscOpenComm( TransMode%, FuncMode% ) as Integer
    Dim ncid As Integer
    Dim rc As Integer
    Dim IPAddrress As string
    Ms_BscOpenComm = -1
    if TransMode=0 then
        'Open the port.
        ncid = BscOpen(CurDir$, 1)
        If ncid < 0 Then GoTo Ms_BscOpenComm_Exit

        'Set serial communications parameters.
        rc = BscSetCom(ncid, 1, 9600, 2, 8, 0)
    else
        'Open the Ethernet line.
        ncid = BscOpen(CurDir$, PACKETETHERNET)
        If ncid < 0 Then GoTo Ms_BscOpenComm_Exit

        'Set Ethernet communications parameters.
        IPAddrress = "999.999.99.99"  '<---Specify any IP address.
        rc = BscSetEther( ncid , IPAddrress , FuncMode, frmMain.hWnd )
    end if
        If rc <> 1 Then
        rc = BscClose(ncid)
        ncid = -1
        GoTo Ms_BscOpenComm_Exit
    End If

    'Connect communications line.
    rc = BscConnect(ncid)
    If rc <> 1 Then
        rc = BscClose(ncid)
        ncid = -1
        GoTo Ms_BscOpenComm_Exit
    End If

Ms_BscOpenComm_Exit:
    Ms_BscOpenComm = ncid

End Function
```

SUPPLE-MENT
- Double underline indicates transmission functions belonging to the MOTOCOM32.
- CurDir$: Visual Basic functionî

This function opens the COM port or the Ethernet line. After the connection is finished, the handle values are sent back as return values. The following operation for the Motocom32.DLL is performed using these handle values.

■ Function Ms_BscCloseComm()

```
Function Ms_BscCloseComm(ncid as integer) as Integer
    Dim rc As Integer

    'Cut the communications line.
    rc = BscDisConnect(ncid)

    'Close the port.
    rc = BscClose(ncid)

    Ms_BscCloseComm = rc

End Sub
```

SUPPLE-MENT
Double underline indicates transmission functions belonging to the MOTOCOM32.

■ Sub CmdDownload_Click ()

```
Sub CmdDownload_Click ()
    Dim nCid As Integer
    Dim rc As Integer
    Dim JobName As String

    'The job name is acquired from the text box.
    JobName = UCase$(TxtJobName.Text)

    'Check the specified job name.
    If JobName <> "" Then
        'Get of communication handler.
        nCid = Ms_BscOpenComm( mode, 0 )  ' mode=0 or 1
         'It is checked whether the communication handler was able to be acquired.
        If nCid >= 0 Then
            'Transmit job.
            rc = BscDownLoad(nCid, JobName)
            'Confirms the return value.
            If rc = 0 Then
                MsgBox "Job transmission completion."
            Else
                MsgBox "Job transmission failure :" + Format$(rc)
            End If

            ‘No. of communication handlers.
            rc = Ms_BscCloseComm(nCid)
       Else
            MsgBox "Cannot open the port."
       End If
    Else
       MsgBox "Please specify the job name. "
    End If

End Sub
```

SUPPLE-MENT

- Double underline indicates transmission functions belonging to the MOTOCOM32.
- single underline indicates functions of which program lists are described below

  Ms_BscOpenComm : " Function Ms_BscOpenComm() "
  Ms_BscCloseComm : " Function Ms_BscCloseComm() "
- UCase$: Visual Basic function
- MsgBox : Visual Basic function

  Displays a message in the dialog box and waits for the button to be pressed.
- Format$: Visual Basic function

■ Sub CmdUpLoad_Click ()

```
Sub CmdUpLoad_Click ()
    Dim nCid As Integer
    Dim rc As Integer
    Dim JobName As String

    'The job name is acquired from the text box.
    JobName = UCase$(TxtJobName.Text)

    'Check the specified job name.
    If JobName <> "" Then
        'Get of communication handler.
        nCid = Ms_BscOpenComm( mode, 0 )  ' mode=0 or 1
         'It is checked whether the communication handler was able to be acquired.

        If nCid >= 0 Then
            'Received job.
            rc = BscUpLoad(nCid, JobName)
            'Confirms the return value.
            If rc = 0 Then
                MsgBox "Job reception completion."
            Else
                MsgBox"Job reception failure :" + Format$(rc)
            End If

            'No. of communication handlers.
            rc = Ms_BscCloseComm(nCid)

        Else
            MsgBox "Cannot open the port."
        End If
    Else
        MsgBox "Please specify the job name. "
    End If

End Sub
```

SUPPLE-MENT

- Double underline indicates transmission functions belonging to the MOTOCOM32.
- single underline indicates functions of which program lists are described below
  - Ms_BscOpenComm : " Function Ms_BscOpenComm() "
  - Ms_BscCloseComm : " Function Ms_BscCloseComm() "
- UCase$: Visual Basic function
- MsgBox : Visual Basic function
  - Displays a message in the dialog box and waits for the button to be pressed.
- Format$: Visual Basic function

■ Sub CmdExit_Click ()

```
Sub CmdExit_Click ()
    'The form is unloaded and the program is ended.
    Unload Me
End Sub
```

UCase$: Visual Basic function

■ CTestDlg::TestOpenComm function

```
// TransMode : 0...RS-232C   1...Ethernet
// FuncMode : 0...Host Control (Client)   1...DCI/Stand Alone (Server)
short CTestDlg::TestOpenComm( int TransMode, int FuncMode )
{
    short ncid;
    short rc;
    char cur_dir[_MAX_DIR ];
    char *IPAddress="999.999.99.99"; //Specify any IP address.

    _getcwd( cur_dir, _MAX_DIR );
    if( TransMode==0 )
    {
        //Open the communications port
        ncid = BscOpen( cur_dir, PACKETCOM );
        if( ncid < 0 )
         {
            return( ncid );
         }
        //Set serial communications parameters.
        rc = BscSetCom( ncid, 1, 9600, 2, 8, 0 );
        if( rc != 1 )
        {
            rc = BscClose( ncid );
            return( -1 );
        }
    }else
    {
        //Open the Ethernet line.
        ncid = BscOpen( cur_dir, PACKETETHERNET );
        if( ncid < 0 )
        {
            return( ncid );
        }
        //Set Ethernet communications parameters.
        rc = BscSetEther( ncid, IPAddress, FuncMode, GetSafeHwnd() );
        if( rc != 1 )
        {
             rc = BscClose( ncid );
            return( -1 );
        }
    }
    //Connect communications line.
    rc = BscConnect( ncid );
    if( rc != 1 )
    {
```

```
        rc = BscClose( ncid );
        return( -1 );
    }

    return( ncid );
}
```

SUPPLE-MENT

- Double underline indicates transmission functions belonging to the MOTOCOM32.
- _getcwd : Visual C++ function

■ CTestDlg::TestCloseComm function

```
short CTestDlg::TestCloseComm( short ncid )
{
    short rc;

    //Cut the communications line.
    rc = BscDisConnect( ncid );

    //Close the port.
    rc = BscClose( ncid );

    return( rc );
}
```

SUPPLE-MENT

Double underline indicates transmission functions belonging to the MOTOCOM32.

■ CTestDlg::OnDownload function

```
void  CTestDlg::OnDownload()
{
    short  nCid;
    short  rc;
    CString  JobName;

    //The job name is acquired from the edit control.
    m_jobname.GetWindowText(JobName);
    JobName.MakeUpper();

    //Check the specified job name.
    if(!JobName.IsEmpty() )
    {
        //Get of communication handler.
       nCid = TestOpenComm ( mode, 0 ); // mode=0  or  1
       //It is checked whether the communication handler was able to be acquired.
       if( nCid >= 0 )
       {
            //Transmit job.
            rc = BscDownLoad( nCid, (char*)(LPCSTR)JobName);
            //Confirms the return value.
            if( rc == 0 )
                AfxMessageBox ( "Job transmission completion." );
            else
                AfxMessageBox ( "Job transmission failure." );

            //No. of communication handlers.
            rc = TestCloseComm( nCid );
        }
        else
            AfxMessageBox ("Cannot open the port." );
    }
    else
    {
        AfxMessageBox ( "Please specify the job name." );
    }
}
```

SUPPLE-MENT

- Double underline indicates transmission functions belonging to the MOTOCOM32.
- single underline indicates functions of which program lists are described below
  - TestOpenComm : " CTestDlg::TestOpenComm function "
  - TestCloseComm : " CTestDlg::TestCloseComm function "
- MakeUpper : Visual C++ function
- AfxMessageBos : Visual C++ function

■ CTestDlg::OnUpload function

```
void  CTestDlg::OnUpload()
{
    short   nCid;
    short   rc;
    CString JobName;

    //The job name is acquired from the edit control.
    m_jobname.GetWindowText(JobName);
    JobName.MakeUpper();

    //Check the specified job name.
    if(!JobName.IsEmpty() )
    {
        //Get of communication handler.
        nCid = TestOpenComm ( mode, 0 ); // mode=0  or  1
        //It is checked whether the communication handler was able to be acquired.
        if( nCid >= 0 )
        {
            //Received job.
            rc = BscUpLoad(nCid, (char*)(LPCSTR)JobName);
            //Confirms the return value.
            if( rc == 0 )
                AfxMessageBox ( "Job reception completion." );
            else
                AfxMessageBox("Job reception failure." );

            //No. of communication handlers.
            rc = TestCloseComm( nCid );
        }
        else
            AfxMessageBox ( "Cannot open the port." );
    }
    else
    {
        AfxMessageBox ( "Please specify the job name." );
    }
}
```

SUPPLE-MENT

- Double underline indicates transmission functions belonging to the MOTOCOM32.
- single underline indicates functions of which program lists are described below
  - TestOpenComm : " CTestDlg::TestOpenComm function "
  - TestCloseComm : " CTestDlg::TestCloseComm function "
- MakeUpper : Visual C++ function
- AfxMessageBos : Visual C++ function

■ private short Ms_BscOpenComm()

```
// mode: 0...RS-232C 1...Ethernet
private short Ms_BscOpenComm( int mode )
{
    short nCid;
    short rc;
    byte[] IPAddress;

    // Convert into byte array.
    byte[] path = Encoding.Default.GetBytes( Application.StartupPath );

    if( mode == 0 )
    {
        // Open the COM port.
        nCid = MotoCom32.BscOpen( ref path[ 0 ], (short)MotoCom32.PACKET.COM );

        if( nCid < 0 )
        {
            return nCid;
        }

        // Set serial communications parameters.
        rc = MotoCom32.BscSetCom( nCid, 1, 9600, 2, 8, 0 );
    }
    else
    {
        // Open the Ethernet line
        nCid = MotoCom32.BscOpen( ref path[ 0 ], (short)MotoCom32.PACKET.ETHER-
NET );

        if( nCid < 0 )
        {
            return nCid;
        }

        // Set Ethernet communications parameters.
        // Specify any IP address
        IPAddress = Encoding.Default.GetBytes( "192.168.0.10" );

        rc = MotoCom32.BscSetEther( nCid, ref IPAddress[ 0 ], 0, this.Handle );
    }

    if( rc != 1 )
    {
        rc = MotoCom32.BscClose( nCid );
        return -1;
```

```
    }

    // Connect communications line.
    rc = MotoCom32.BscConnect( nCid );

    if( rc != 1 )
    {
        rc = MotoCom32.BscClose( nCid );
        return -1;
    }

    return nCid;
}
```

SUPPLEMENT

- Double underline indicates transmission functions belonging to the MOTOCOM32.
- Encoding.Default.GetBytes() : Function of Visual Studio C#
  Convert String in brackets into byte array.
- Application.StartupPath : Function of Visual Studio C#
  Get the path of the folder where the application executes.

This function opens the COM port or the Ethernet line. After the connection is finished, the handle values are sent back as return values. The following operation for the Motocom32.DLL is performed using these handle values.

■ private short Ms_BscCloseComm()

```
private short Ms_BscCloseComm( short nCid )
{
    short rc;

    // Cut the communications line.
    rc = MotoCom32.BscDisConnect( nCid );

    // Close the port.
    rc = MotoCom32.BscClose( nCid );

    return rc;
}
```

SUPPLEMENT

Double underline indicates transmission functions belonging to the MOTOCOM32.

■ private void CmdDownLoad_Click()

```
private void CmdDownLoad_Click( object sender, EventArgs e )
{
    short nCid;
```

```
    short rc;
    byte[] jobname;

    // the job name is acquired from the text box.
    jobname = Encoding.Default.GetBytes( TxtJobName.Text.ToUpper() );

    // Check the specified the job name.
    if( jobname.Length > 0 )
    {
        // Set the communication mode. (mode: 0:RS-232C, 1:Ethernet)
        nCid = Ms_BscOpenComm( 1 );

        // It is checked whether the communication handler was able to be acquired.
        if( nCid != -1 )
        {
            // Transmit job.
            rc = MotoCom32.BscDownLoad( nCid, ref jobname[ 0 ] );

            // Confirms the return value.
            if( rc == 0 )
            {
                MessageBox.Show( "Job transmission completion." );
            }
            else
            {
                MessageBox.Show( "Job transmission failure :" + rc.ToString() );
            }

            // Release the communication handle.
            rc = Ms_BscCloseComm( nCid );
        }
        else if( nCid == -8 )
        {
            MessageBox.Show( "License error." );
        }
        else
        {
            MessageBox.Show( "Cannot open the port." );
        }
    }
    else
    {
        MessageBox.Show( "Please specify the job name." );
    }
}
```

SUPPLEMENT

- Double underline indicates transmission functions belonging to the MOTOCOM32.
- Single underline indicates functions of which program lists are described below.
  Ms_BscOpenComm : " private short Ms_BscOpenComm() "
  Ms_BscCloseComm : " private short Ms_BscCloseComm() "
- ToUpper() : Function of Visual Studio C#
  Uppercase the string.
- MessageBox.Show() : Function of Visual Studio C#
  Display the message in the dialog box, and wait for clicking the button.
- ToString() : Function of Visual Studio C#
  Convert into string using the specified form or the specified culture-specific formatting information.

■ private void CmdUpLoad_Click()

```
private void CmdUpLoad_Click( object sender, EventArgs e )
{
    short nCid;
    short rc;
    byte[] jobname;

    // The job name is acquired from the text box.
    jobname = Encoding.Default.GetBytes( TxtJobName.Text.ToUpper() );

    // Check the specified job name.
    if( jobname.Length > 0 )
    {
        // Set the communication mode. (mode: 0:RS-232C, 1:Ethernet)
        nCid = Ms_BscOpenComm( 1 );

        // It is checked whether the communication handler was able to be acquired.
        if( nCid >= 0 )
        {
            // Receive job.
            rc = MotoCom32.BscUpLoad( nCid, ref jobname[ 0 ] );

            // Confirms the return value.
            if( rc == 0 )
            {
                MessageBox.Show( "Job reception completion." );
            }
            else
            {
                MessageBox.Show( "Job reception failure :" + rc.ToString() );
            }
```

```
            // Release the communication handle.
            rc = Ms_BscCloseComm( nCid );
        }
        else if( nCid == -8 )
        {
            MessageBox.Show( "License error." );
        }
        else
        {
            MessageBox.Show( "Cannot open the port." );
        }
    }
    else
    {
        MessageBox.Show( "Please specify the job name." );
    }
}
```

SUPPLEMENT

- Double underline indicates transmission functions belonging to the MOTOCOM32.
- Single underline indicates functions of which program lists are described below.
  Ms_BscOpenComm : " private short Ms_BscOpenComm() "
  Ms_BscCloseComm : " private short Ms_BscCloseComm() "
- ToUpper() : Function of Visual Studio C#
  Uppercase the string.
- MessageBox.Show() : Function of Visual Studio C#
  Display the message in the dialog box, and wait for clicking the button.
- ToString() : Function of Visual Studio C#
  Convert into string using the specified form or the specified culture-specific formatting information.

■ private void CmdExit_Click()

```
private void CmdExit_Click( object sender, EventArgs e )
{
    // The form is unloaded and the program is ended.
    this.Close();
}
```

SUPPLEMENT

- this.Close() : Function of Visual Studio C#
  Unload the form.

7 COMMUNICATION TRANSMISSION

7.1 Outline

MOTOCOM32.DLL is a transmission library that controls the data transmission function of the YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, the YASNAC XRC, MRC, ERC, and ERC2 on a personal computer. This library is composed in the form of Microsoft Windows DLL (Dynamic Link Library).

NOTE MOTOCOM32.DLL is located below the MOTOCOM32 installation directory. When a transmission application is created, copy this file to the same directory as the application. MOTOCOM.H and MOTOCOM32.LIB files are provided in the MOTOCOM32 installation directory. Use these files when a transmission application is created in C-language.

Transmission library has the following functions.

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

7.2 File Data Transmission Function

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

BscDownload
BscDownloadEx
BscUpload
BscUploadEx

■ BscDownload

FUNCTION	Sends a specified file to the robot controller.
FORMAT	_declspec(dllexport) short APIENTRY BscDownLoad(short nCid,char *fname);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *fname File name to be sent.
	OUT (Return) None
	Return Value 0 : Normal completion Others : Transmission error
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	“BscDownloadEx” “BscUpload” “BscUploadEx”

■ BscDownloadEx

FUNCTION	Sends a specified file to the robot controller. A directory where the sending file exists can be specified.
FORMAT	_declspec(dllexport) short APIENTRY BscDownLoadEx(short nCid,char *fname, char *path, BOOL nFlg);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *fname File name to be sent *path Diretory path of sending source data nFlg TRUE : Changes the directory temporarily and restores it at the end. FALSE : Changes the directory and completes the processing.
	OUT (Return) None
	Return Value 0 : Normal completion Others : Transmission error
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	“BscDownload” “BscUpload” “BscUploadEx”

■ BscUpload

FUNCTION	Receives a specified file from the robot controller.
FORMAT	_declspec(dllexport) short APIENTRY BscUpLoad(short nCid,char *fname);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *fname File name to be received
	OUT (Return) None
	Return Value 0 : Normal completion Others : Receiving error
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	“BscUploadEx” “BscDownload” “BscDownloadEx”

■ BscUploadEx

FUNCTION	Receives a specified file from the robot controller. The directory where the file is send to can be specified.
FORMAT	_declspec(dllexport) short APIENTRY BscUpLoadEx(short nCid,char *fname, char *path, BOOL nFlg);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *fname File name to be received *path Diretory path of sending source data nFlg TRUE : Changes the directory temporarily and restores it at the end. FALSE : Changes the directory and completes the processing.
	OUT (Return) None
	Return Value 0 : Normal completion Others : Receiving error
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	“BscUpload” “BscDownload” “BscDownloadEx”

7.3 Robot Control Function

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

Status Read

BscFindFirst
BscFindFirstMaster
BscFindNext
BscFindNextMaster
BscGetCtrlGroup
BscGetCtrlGroupXrc
BscGetCtrlGroupDX
BscDownload
BscDownloadEx
BscGetError
BscGetError2
BscGetFirstAlarm
BscGetFirstAlarmS
BscGetNextAlarm
BscGetNextAlarmS
BscGetStatus
BscGetUFrame
BscGetVarData
BscGetVarData2
BscHostGetVarData
BscHostGetVarDataM
BscGetVarDataEx
BscIsAlarm
BscIsCtrlGroup
BscIsCtrlGroupXrc
BscIsCtrlGroupDX
BscIsCycle
BscIsError
BscIsErrorCode
BscIsHold
BscIsJobLine
BscIsJobName
BscIsJobStep
BscIsLoc
BscGetPulsePos
BscIsPlayMode
BscIsRemoteMode
BscIsRobotPos
BscGetCartPos
BscIsServo
BscIsTaskInf
BscIsTaskInfXrc
BscIsTeachMode
BscJobWait
BscReadAlarmS
BscGetTorque
BscGetMaxTorque
BscGetEncTmp
BscGetSysTime

System Control

BscCancel
BscChangeTask
BscContinueJob
BscConvertJobP2R
BscConvertJobR2P
BscDeleteJob
BscHoldOff
BscHoldOn
BscHostPutVarData
BscHostPutVarDataM
BscPutVarDataEx
BscImov
BscImovEx
BscImovEx2
BscMDSP
BscMov
BscMovEx
BscMovEx2
BscMovj
BscMovjEx
BscMovl
BscMovlEx
BscOPLock
BscOPUnLock
BscPMov
BscPMovEx
BscPMovj
BscPMovjEx
BscPMovl
BscPMovlEx
BscPutUFrame
BscPutUFrameEx2
BscPutVarData
BscPutVarData2
BscStartJob
BscSelectJob
BscSelectMode
BscSelLoopCycle
BscSelOneCycle
BscSelStepCycle
BscSetLineNumber
BscSetMasterJob
BscReset
BscSetCtrlGroup
BscSetCtrlGroupXrc
BscSetCtrlGroupDX
BscServoOff
BscServoOn
BscUpload
BscUploadEx

■ BscFindFirst

FUNCTION	Reads the first job name from the all job list registered at the present time.
FORMAT	_declspec(dllexport) short APIENTRY BscFindFirst(short nCid,char *fname,short size);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *fname First job name storage pointer size Job name storage area size
	OUT (Return) *fname First job name storage pointer
	Return Value -1 : No job -2 : Internal error (memory allocation error) -3 : Internal error (memory lock error) -4 : Other errors 0 : Job found
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	“BscFindNext”

■ BscFindFirstMaster

FUNCTION	Reads the first job name from the job list that belongs to the target job.
FORMAT	_declspec(dllexport) short APIENTRY BscFindFirstMaster(short nCid,char *fname,short size);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *fname First job name storage pointer size Job name storage area size
	OUT (Return) *fname First job name storage pointer
	Return Value -1 : No job -2 : Internal error (memory allocation error) -3 : Internal error (memory lock error) -4 : Other errors 0 : Job found
REMARKS	**Call Condition** The BscSelectJob function must be called up and the target job name must be selected before executing this function.
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	“BscFindNextMaster” "BscSelectJob"

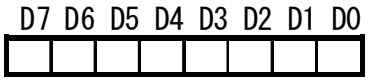
■ BscFindNext

FUNCTION	Reads the next job name registered at the present time.
FORMAT	_declspec(dllexport) short APIENTRY BscFindNext(short nCid,char *fname,short size);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *fname N-th job name storage pointer size Job name storage area size
	OUT (Return) *fname N-th job name storage pointer
	Return Value -1 : No next job 0 : Next job found
REMARKS	**Call Condition** The BscFindFirst function must be called up before executing this function.
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	“BscFindFirst”

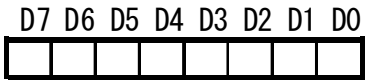
■ BscFindNextMaster

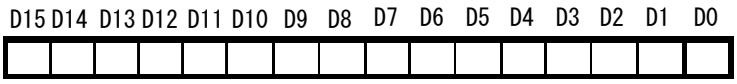
FUNCTION	Reads the next job name in the job list that belongs to the target job.
FORMAT	_declspec(dllexport) short APIENTRY BscFindNextMaster(short nCid,char *fname,short size);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *fname N-th job name storage pointer size Job name storage area size
	OUT (Return) *fname N-th job name storage pointer
	Return Value -1 : No next job 0 : Next job found
REMARKS	**Call Condition** The BscFindFirstMaster function must be called up before executing this function.
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	“BscFindFirstMaster”

■ BscGetCtrlGroup

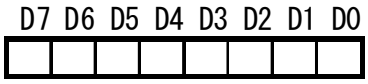
FUNCTION	Reads control group and task information.
FORMAT	_declspec(dllexport) short APIENTRY BscGetCtrlGroup(short nCid,short *groupinf,short *taskinf);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *groupinf Control group information storage pointer *taskinf Task information storage pointer
	OUT (Return) *groupinf Control group information storage pointer *taskinf Task information storage pointer
	Return Value 0 : Normal completion Others : Error codes
REMARKS	**Restrictions** This function is effective only for transmission with the MRC. Refer to the BscGetCtrlGroupDX for transmission with the YRC1000/YRC1000micro/DX200/DX100. Refer to the BscGetCtrlGroupXrc for transmission with the FS100/ NX100/XRC.
	Control Group Information The control group information is represented by bit data in decimals.  D0 : R1 (Robot1) D1 : R2 (Robot2) D2 : S1 (Station1) D3 : S2 (Station2) D4 : S3 (Station3) D5 : S4 (Station4) D6 : S5 (Station5) D7 : S6 (Station6)
	Task Information The task information is represented as follows. 0 : Master task 1 : Sub 1 task 2 : Sub 2 task "0" is returned if independent control is not allowed in the system.
CONTROLLER	MRC (Serial Port, Ethernet)
REFERENCE	"BscGetCtrlGroupDX" "BscSetCtrlGroupDX" "BscIsCtrlGroupDX" "BscGetCtrlGroupXrc" "BscSetCtrlGroupXrc" "BscIsCtrlGroupXrc" "BscIsTaskInfXrc" "BscSetCtrlGroup" "BscIsCtrlGroup" "BscIsTaskInf" "BscChangeTask"

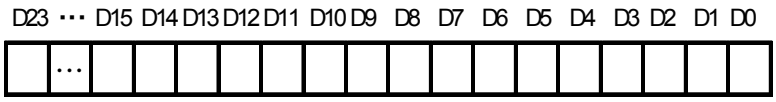
■ BscGetCtrlGroupXrc

FUNCTION	Reads control group and task information.
FORMAT	_declspec(dllexport) short APIENTRY BscGetCtrlGroupXrc(short nCid,short *groupinf,short *stationinf,short *taskinf);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *groupinf Control group information storage pointer (robot axis) *stationinf Control group information storage pointer (station axis) *taskinf Task information storage pointer
	OUT (Return) *groupinf Control group information storage pointer (robot axis) *stationinf Control group information storage pointer (station axis) *taskinf Task information storage pointer
	Return Value 0 : Normal completion Others : Error codes
REMARKS	**Restrictions** This function is effective for transmission with the FS100/NX100/XRC. Refer to the BscGetCtrlGroupDX for transmission with the YRC1000/YRC1000micro/DX200/DX100. Refer to BscGetCtrl-Group for transmission with the MRC.
	Control Group Information (Robot Axis) The control group information is represented by bit data in decimals.  D0 : R1 (Robot1) D1 : R2 (Robot2) D2 : R3 (Robot3) D3 : R4 (Robot4)

	Control Group Information (Station Axis) The control group information is represented by bit data in decimals.  D0：S1（Station1） D1：S2（Station2） D2：S3（Station3） D3：S4（Station4） D4：S5（Station5） D5：S6（Station6） D6：S7（Station7） D7：S8（Station8） D8：S9（Station9） D9：S10（Station10） D10：S11（Station11） D11：S12（Station12）
REMARKS	**Task Information** The task information is represented as follows. 0 : Master task 1 : Sub 1 task 2 : Sub 2 task 3 : Sub 3 task 4 : Sub 4 task 5 : Sub 5 task 6 : Sub 6 task 7 : Sub 7 task "0" is returned if independent control is not allowed in the system.
CONTROLLER	FS100, NX100, XRC (Serial Port, Ethernet)
REFERENCE	"BscGetCtrlGroupDX" "BscSetCtrlGroupDX" "BscIsCtrlGroupDX" “BscSetCtrlGroupXrc” “BscIsCtrlGroupXrc” “BscIsTaskInfXrc” “BscGetCtrlGroup“ “BscSetCtrlGroup” “BscIsCtrlGroup” “BscIsTaskInf” “BscChangeTask”

■ BscGetCtrlGroupDX

FUNCTION	Reads control group and task information.
FORMAT	_declspec(dllexport) short APIENTRY BscGetCtrlGroupDX(short nCid,long *groupinf,long *stationinf,short *taskinf);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *groupinf Control group information storage pointer (robot axis) *stationinf Control group information storage pointer (station axis) *taskinf Task information storage pointer
	OUT (Return) *groupinf Control group information storage pointer (robot axis) *stationinf Control group information storage pointer (station axis) *taskinf Task information storage pointer
	Return Value 0 : Normal completion Others : Error codes
REMARKS	**Restrictions** This function is effective for transmission with the YRC1000/ YRC1000micro/DX200/DX100. Refer to BscGetCtrlGroupXrc for transmission with the FS100/ NX100/XRC. Refer to BscGetCtrlGroup for transmission with the MRC.
	Control Group Information (Robot Axis) The control group information is represented by bit data in decimals.  D0 : R1 (Robot1) D1 : R2 (Robot2) D2 : R3 (Robot3) D3 : R4 (Robot4) : : D7 : R8 (Robot8)

REMARKS	**Control Group Information (Station Axis)** The control group information is represented by bit data in decimals.  D0 : S1 (Station1) D1 : S2 (Station2) D2 : S3 (Station3) D3 : S4 (Station4) D4 : S5 (Station5) D5 : S6 (Station6) D6 : S7 (Station7) D7 : S8 (Station8) D8 : S9 (Station9) D9 : S10 (Station10) D10 : S11 (Station11) D11 : S12 (Station12) ⋮ D23 : S24 (Station24)
	Task Information The task information is represented as follows. 0 : Master task 1 : Sub 1 task 2 : Sub 2 task 3 : Sub 3 task 4 : Sub 4 task 5 : Sub 5 task 6 : Sub 6 task 7 : Sub 7 task "0" is returned if independent control is not allowed in the system.
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100 (Serial Port, Ethernet)
REFERENCE	"BscSetCtrlGroupDX" "BscIsCtrlGroupDX" "BscGetCtrlGroupXrc" "BscSetCtrlGroupXrc" "BscIsCtrlGroupXrc" "BscIsTaskInfXrc" "BscGetCtrlGroup" "BscSetCtrlGroup" "BscIsCtrlGroup" "BscIsTaskInf" "BscChangeTask"

■ BscGetError

FUNCTION	Reads an error code or alarm code.
FORMAT	_declspec(dllexport) short APIENTRY BscGetError(short nCid);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number
	OUT (Return) None
	Return Value -1 : Acquisition Failure 0 : No error Others : Error codes
REMARKS	**Restrictions** This function is effective for transmission with the ERC. Refer to BscGetError2 for transmission with the YRC1000/ YRC1000micro/DX200/DX100/FS100/NX100/XRC/MRC.
CONTROLLER	ERC (Serial Port)
REFERENCE	“BscGetFirstAlarm” “BscGetNextAlarm” “BscIsAlarm” “BscIsError” “BscIsErrorCode”

■ BscGetError2

FUNCTION	Reads an error code or alarm code.
FORMAT	_declspec(dllexport) short APIENTRY BscGetError2(short nCid);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number
	OUT (Return) None
	Return Value -1 : Acquisition Failure 0 : No error Others : Error codes
REMARKS	**Restrictions** This function is effective for transmission with the YRC1000/ YRC1000micro/DX200/DX100/FS100/NX100/XRC/MRC. Refer to BscGetError for transmission with the ERC.
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, NX100, XRC, MRC (Serial Port, Ethernet)
REFERENCE	“BscGetFirstAlarm” “BscGetNextAlarm” “BscIsAlarm” “BscIsError” “BscIsErrorCode”

■ BscGetFirstAlarm

FUNCTION	Reads an alarm code and returns the alarm code and alarm data.
FORMAT	_declspec(dllexport) short APIENTRY BscGetFirstAlarm(short nCid,short *data);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *data Alarm data storage pointer
	OUT (Return) *data Alarm data storage pointer
	Return Value 0 : No alarm Others : Alarm code numbers
REMARKS	**Call Condition** The BscGetError2 function must be called up before executing this function.
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	“BscGetError2” “BscGetNextAlarm” “BscIsAlarm”

■ BscGetFirstAlarmS

FUNCTION	Reads an alarm code and returns the alarm code, alarm data and alarm message.
FORMAT	_declspec(dllexport) short APIENTRY BscGetFirstAlarmS(short nCid,short *data,char *msg);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *data Alarm data storage pointer *msg Alarm message storage pointer
	OUT (Return) *data Alarm data storage pointer *msg Alarm message storage pointer
	Return Value 0 : No alarm Others : Alarm code numbers
REMARKS	**Call Condition** The BscReadAlarmS function must be called up before executing this function.
	Restrictions This function is effective for transmission with the YRC1000/ YRC1000micro/DX200/DX100/FS100/NX100.
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100 (Serial Port, Ethernet)
REFERENCE	“BscReadAlarmS” “BscGetNextAlarm” “BscIsAlarm”

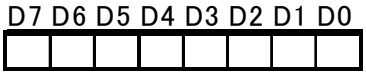
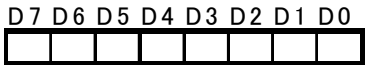
■ BscGetNextAlarm

FUNCTION	Reads the next alarm code and alarm data.
FORMAT	_declspec(dllexport) short APIENTRY BscGetNextAlarm(short nCid,short *data);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *data Alarm data storage pointer
	OUT (Return) *data Alarm data storage pointer
	Return Value 0 : No alarm Others : Alarm code numbers
REMARKS	**Call Condition** The BscGetFirstAlarm function must be called up before executing this function.
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	“BscGetError2” “BscGetFirstAlarm” “BscIsAlarm”

■ BscGetNextAlarmS

FUNCTION	Reads the next alarm code, alarm data and alarm message.
FORMAT	_declspec(dllexport) short APIENTRY BscGetNextAlarmS(short nCid,short *data,char *msg);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *data Alarm data storage pointer *msg Alarm message storage pointer
	OUT (Return) *data Alarm data storage pointer *msg Alarm message storage pointer
	Return Value 0 : No alarm Others : Alarm code numbers
REMARKS	**Call Condition** The BscGetFirstAlarmS function must be called up before executing this function.
	Restrictions This function is effective for transmission with the YRC1000/ YRC1000micro/DX200/DX100/FS100/NX100.
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100 (Serial Port, Ethernet)
REFERENCE	“BscReadAlarmS” “BscGetFirstAlarmS” “BscIsAlarm”

■ BscGetStatus

FUNCTION	Reads the status information.
FORMAT	_declspec(dllexport) short APIENTRY BscGetStatus(short nCid,short *d1,short *d2);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *d1 Data 1 storage pointer *d2 Data 2 storage pointer
	OUT (Return) *d1 Data 1 storage pointer *d2 Data 2 storage pointer
	Return Value -1 : Acquisition Failure Others : Normal completion
REMARKS	**Data 1** Data 1 are represented by bit data in decimals.  D0 : Step D1 : 1-cycle D2 : Auto operation D3 : Operating D4 : Operation at safe speed D5 : Teach * D6 : Play * D7 : Command remote * * : Effective only for YRC1000, YRC1000micro, DX200, DX100, NX100,XRC and MRC.
	Data 2 Data 2 are represented by bit data in decimals.  D0 : Hold (YRC1000/YRC1000micro/DX200/DX100/NX100 XRC/MRC: Playback box hold, ERC:Panel hold) D1 : Hold (YRC1000/YRC1000micro /DX200/DX100/NX100 XRC/MRC: Programming pendant hold, ERC: T-BOX hold) D2 : Hold (External hold) D3 : Hold (Command hold) D4 : Alarm occurred D5 : Error occurred D6 : Servo ON
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)

REFERENCE	“"BscStatus"” “BscIsAlarm” “BscIsCycle” “BscIsHold” “BscIsPlayMode” “BscIsRemoteMode” “BscIsServo” “BscIsTeachMode”

■ BscGetUFrame

FUNCTION	Reads specified user frame data.
FORMAT	_declspec(dllexport) short APIENTRY BscGetUFrame (short nCid,char *ufname,double *p);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *ufname Storage pointer of user coordinate name *p User coordinate data storage pointer
	OUT (Return) *p User coordinate data storage pointer
	Return Value -1 : User coordinate name error 0 : Normal completion Others : Error codes
REMARKS	**Restrictions** This function does not support the robot with 7 axes or more.
	User Coordinate Name The following coordinate names correspond to the user coordinate numbers.

User Coordinate Name	Specified Name
User coordinate 1	UF1
User coordinate 2	UF2
User coordinate 3	UF3
:	:
User coordinate 60	UF60
User coordinate 61	UF61
User coordinate 62	UF62
User coordinate 63	UF63

* User coordinate numbers 9 to 63 are effective for YRC1000/ YRC1000micro/DX200/DX100.
* User coordinate numbers 9 to 16 are effective for FS100.
* User coordinate numbers 9 to 24 are effective for NX100/XRC/ MRC.

REMARKS

Variable type

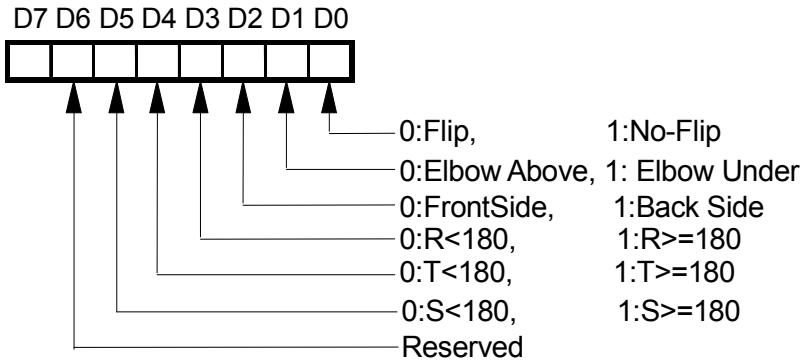
Coordinate values of the user coordinate system specified with the user coordinate number are assigned to the user coordinate data as follows.

Variables	Coordinate System	Meaning
P[0]	ORG	X-axis coordinate (unit: mm, effective down to 3 decimal places
P[1]		Y-axis coordinate (unit: mm, effective down to 3 decimal places)
P[2]		Z-axis coordinate (unit: mm, effective down to 3 decimal places)
P[3]		Wrist angle Rx (unit:°, effective down to 2 decimal places) *1
P[4]		Wrist angle Ry (unit:°, effective down to 2 decimal places) *1
P[5]		Wrist angle Rz (unit:°, effective down to 2 decimal places) *1
P[6]		Form
P[7]	XX	X-axis coordinate (unit: mm, effective down to 3 decimal places
P[8]		Y-axis coordinate (unit: mm, effective down to 3 decimal places)
P[9]		Z-axis coordinate (unit: mm, effective down to 3 decimal places)
P[10]		Wrist angle Rx (unit:°, effective down to 2 decimal places) *1
P[11]		Wrist angle Ry (unit:°, effective down to 2 decimal places) *1
P[12]		Wrist angle Rz (unit:°, effective down to 2 decimal places) *1
P[13]		Form
P[14]	XY	X-axis coordinate (unit: mm, effective down to 3 decimal places
P[15]		Y-axis coordinate (unit: mm, effective down to 3 decimal places)
P[16]		Z-axis coordinate (unit: mm, effective down to 3 decimal places)
P[17]		Wrist angle Rx (unit:°, effective down to 2 decimal places) *1
P[18]		Wrist angle Ry (unit:°, effective down to 2 decimal places) *1
P[19]		Wrist angle Rz (unit:°, effective down to 2 decimal places) *1
P[20]		Form
P[21]	Tool number (0 to 23)	
P[22]	7th axis pulse number (mm for traveling axis)	
P[23]	8th axis pulse number (mm for traveling axis)	
P[24]	9th axis pulse number (mm for traveling axis)	
P[25]	10th axis pulse number	
P[26]	11th axis pulse number	
P[27]	12th axis pulse number	

*1 YRC1000, YRC1000micro, DX200, DX100 is effective down to 4 decimal places.

Form

The form data are represented by bit data in decimals.



* With the ERC or ERC2, the data from D3 to D7 are disregarded.
* With the MRC or MRC2, the data D6 and D7 are disregarded.

CONTROLLER

YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet)
ERC (Serial Port)

REFERENCE	“BscPutUFrame”

■ BscGetVarData

FUNCTION	Receives variables.
FORMAT	_declspec(dllexport) short APIENTRY BscGetVarData(short nCid,short type,short varno,double *p);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number type Variable type varno Variable number *p Head pointer to the numeric variable storage area
	OUT (Return) *p Head pointer to the numeric variable storage area
	Return Value 0 : Normal completion Others : Error codes
REMARKS	**Restrictions** This function is effective only for transmission with the YRC1000/ YRC1000micro/DX200/DX100/FS100/NX100/XRC/MRC.
	Variable Types The variable types are represented as follows. 0 : Byte type 1 : Integer type 2 : Double-precision type 3 : Real type 4 : Robot axis position type 5 : Base axis position type 6 : Station axis position type (pulse type only)

REMARKS

Content of the numeric variable storage area

Depending on the variable type, the numeric variable storage area contains the number of values indicated below.

Variable Type Number	Data Type (Pulse/ XYZ)	Number of values	Content				
			P[0]	P[1]	P[2]	P[3]	P[4]
0	-	1	Byte				
1	-	1	Integer	-	-	-	-
2	-	1	Double	-	-	-	-
3	-	1	Real	-	-	-	-
4	Pulse	8	0	S-axis Pulses	L-axis Pulses	U-axis Pulses	R-axis Pulses
4	XYZ	10	1	Coordinate Type	X-axis (mm)	Y-axis (mm)	Z-axis (mm)
5	Pulse	8	0	Base Axis-1 Pulses	Base Axis-2 Pulses	Base Axis-3 Pulses	Base Axis-4 Pulses
5	XYZ	10	1	Coordinate Type	X-axis (mm)	Y-axis (mm)	Z-axis (mm)
6	Pulse	8	0	Station Axis-1 Pulses	Station Axis-2 Pulses	Station Axis-3 Pulses	Station Axis-4 Pulses

Variable Type Number	Data Type (Pulse/ XYZ)	Number of values	Content				
			P[5]	P[6]	P[7]	P[8]	P[9]
0	-	1					
1	-	1	-	-	-	-	-
2	-	1	-	-	-	-	-
3	-	1	-	-	-	-	-
4	Pulse	8	B-axis Pulses	T-axis Pulses	Tool Number		
4	XYZ	10	Rx Angle(deg)	Ry Angle(deg)	Rz Angle(deg)	Form	Tool Number
5	Pulse	8	Base Axis-5 Pulses	Base Axis-6 Pulses	Tool Number		
5	XYZ	10	Rx Angle(deg)	Ry Angle(deg)	Rz Angle(deg)	Form	Tool Number
6	Pulse	8	Station Axis-5 Pulses	Station Axis-6 Pulses	Tool Number		

The robot axis position and base axis position type variables include the pulse type and XYZ type, according to the first return value.
The station axis position type variable contains the pulse type only.
See the following for details on the coordinate system types and form.

Coordinate Types

YRC1000, YRC1000micro, DX200, DX100

The following coordinate names correspond to the coordinate type data.

Coordinate type	Coordinate name	Coordinate type	Coordinate name
0	Base coordinate	:	:
1	Robot coordinate	:	:
2	User coordinate 1	63	User coordinate 62
3	User coordinate 2	64	User coordinate 63
:	:	65	Tool coordinate
:	:	66	Master tool coordinate

REMARKS

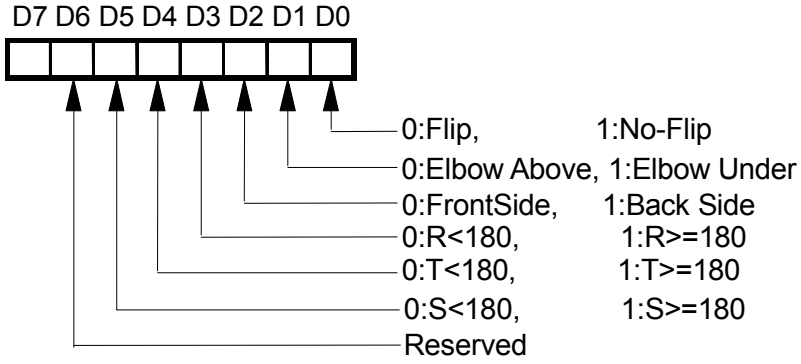
Coordinate Types
FS100
The following coordinate names correspond to the coordinate type data.

Coordinate type	Coordinate name	Coordinate type	Coordinate name
0	Base coordinate	10	User coordinate 9
1	Robot coordinate	11	User coordinate 10
2	User coordinate 1	12	User coordinate 11
3	User coordinate 2	13	User coordinate 12
4	User coordinate 3	14	User coordinate 13
5	User coordinate 4	15	User coordinate 14
6	User coordinate 5	16	User coordinate 15
7	User coordinate 6	17	User coordinate 16
8	User coordinate 7	18	Tool coordinate
9	User coordinate 8	19	Master tool coordinate

Coordinate Types
NX100/XRC/MRC
The following coordinate names correspond to the coordinate type data.

Coordinate type	Coordinate name	Coordinate type	Coordinate name
0	Base coordinate	14	User coordinate 13
1	Robot coordinate	15	User coordinate 14
2	User coordinate 1	16	User coordinate 15
3	User coordinate 2	17	User coordinate 16
4	User coordinate 3	18	User coordinate 17
5	User coordinate 4	19	User coordinate 18
6	User coordinate 5	20	User coordinate 19
7	User coordinate 6	21	User coordinate 20
8	User coordinate 7	22	User coordinate 21
9	User coordinate 8	23	User coordinate 22
10	User coordinate 9	24	User coordinate 23
11	User coordinate 10	25	User coordinate 24
12	User coordinate 11	26	Tool coordinate
13	User coordinate 12	27	Master tool coordinate

Form
The form data are represented by bit data in decimals.



* With the MRC or MRC2, the data of D5 and D7 are disregarded.

CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet)
REFERENCE	"BscPutVarData" "BscPutVarData2"

■ BscGetVarData2

FUNCTION	Receives variables. (robot with 7 axes or more)
FORMAT	_declspec(dllexport) short APIENTRY BscGetVarData2(short nCid,short type,short varno,double *p);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number type Variable type varno Variable number *p Head pointer to the numeric variable storage area
	OUT (Return) *p Head pointer to the numeric variable storage area
	Return Value 0 : Normal completion Others : Error codes
REMARKS	**Restrictions** This function is effective only for transmission with the YRC1000/ YRC1000micro/DX200/DX100/FS100/NX100/XRC.
	Variable Types The variable types are represented as follows. 0 : Byte type 1 : Integer type 2 : Double-precision type 3 : Real type 4 : Robot axis position type 5 : Base axis position type 6 : Station axis position type (pulse type only)

REMARKS

Content of the numeric variable storage area

Depending on the variable type, the numeric variable storage area contains the number of values indicated below.

Variable Type Number	Data Type (Pulse/ XYZ)	Number of values	Content				
			P[0]	P[1]	P[2]	P[3]	P[4]
0	-	1	Byte				
1	-	1	Integer	-	-	-	-
2	-	1	Double	-	-	-	-
3	-	1	Real	-	-	-	-
4	Pulse	10	0	S-axis Pulses	L-axis Pulses	U-axis Pulses	R-axis Pulses
4	XYZ	10	1	Coordinate Type	X-axis (mm)	Y-axis (mm)	Z-axis (mm)
5	Pulse	8	0	Base Axis-1 Pulses	Base Axis-2 Pulses	Base Axis-3 Pulses	Base Axis-4 Pulses
5	XYZ	10	1	Coordinate Type	X-axis (mm)	Y-axis (mm)	Z-axis (mm)
6	Pulse	8	0	Station Axis-1 Pulses	Station Axis-2 Pulses	Station Axis-3 Pulses	Station Axis-4 Pulses

Variable Type Number	Data Type (Pulse/ XYZ)	Number of values	Content				
			P[5]	P[6]	P[7]	P[8]	P[9]
0	-	1					
1	-	1	-	-	-	-	-
2	-	1	-	-	-	-	-
3	-	1	-	-	-	-	-
4	Pulse	10	B-axis Pulses	T-axis Pulses	7thaxis Pulses	8thaxis Pulses	Tool Number
4	XYZ	10	Rx Angle(deg)	Ry Angle(deg)	Rz Angle(deg)	Form	Tool Number
5	Pulse	8	Base Axis-5 Pulses	Base Axis-6 Pulses	Tool Number	-	-
5	XYZ	10	Rx Angle(deg)	Ry Angle(deg)	Rz Angle(deg)	Form	Tool Number
6	Pulse	8	Station Axis-5 Pulses	Station Axis-6 Pulses	Tool Number	-	-

The robot axis position and base axis position type variables include the pulse type and XYZ type, according to the first return value.
The station axis position type variable contains the pulse type only.
See the following for details on the coordinate system types and form.

Coordinate Types

YRC1000, YRC1000micro, DX200, DX100

The following coordinate names correspond to the coordinate type data.

Coordinate type	Coordinate name	Coordinate type	Coordinate name
0	Base coordinate	:	:
1	Robot coordinate	:	:
2	User coordinate 1	63	User coordinate 62
3	User coordinate 2	64	User coordinate 63
:	:	65	Tool coordinate
:	:	66	Master tool coordinate

REMARKS

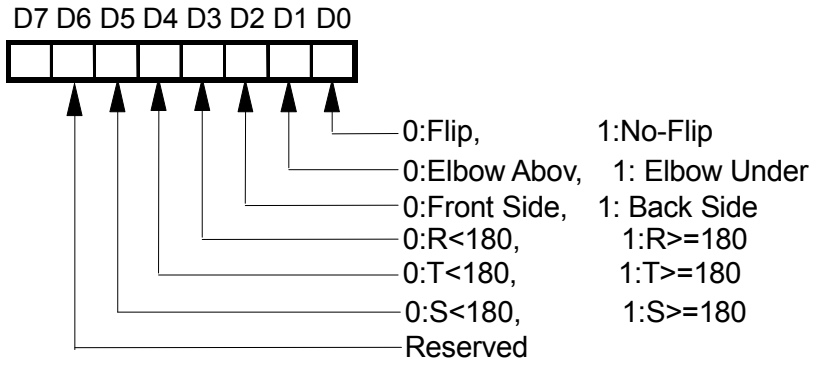
Coordinate Types
FS100
The following coordinate names correspond to the coordinate type data.

Coordinate type	Coordinate name	Coordinate type	Coordinate name
0	Base coordinate	10	User coordinate 9
1	Robot coordinate	11	User coordinate 10
2	User coordinate 1	12	User coordinate 11
3	User coordinate 2	13	User coordinate 12
4	User coordinate 3	14	User coordinate 13
5	User coordinate 4	15	User coordinate 14
6	User coordinate 5	16	User coordinate 15
7	User coordinate 6	17	User coordinate 16
8	User coordinate 7	18	Tool coordinate
9	User coordinate 8	19	Master tool coordinate

Coordinate Types
NX100/XRC
The following coordinate names correspond to the coordinate type data.

Coordinate type	Coordinate name	Coordinate type	Coordinate name
0	Base coordinate	14	User coordinate 13
1	Robot coordinate	15	User coordinate 14
2	User coordinate 1	16	User coordinate 15
3	User coordinate 2	17	User coordinate 16
4	User coordinate 3	18	User coordinate 17
5	User coordinate 4	19	User coordinate 18
6	User coordinate 5	20	User coordinate 19
7	User coordinate 6	21	User coordinate 20
8	User coordinate 7	22	User coordinate 21
9	User coordinate 8	23	User coordinate 22
10	User coordinate 9	24	User coordinate 23
11	User coordinate 10	25	User coordinate 24
12	User coordinate 11	26	Tool coordinate
13	User coordinate 12	27	Master tool coordinate

Form
The form data are represented by bit data in decimals.



CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC (Serial Port, Ethernet)
REFERENCE	"BscPutVarData" "BscPutVarData2"

■ BscHostGetVarData

FUNCTION	Receives variables.
FORMAT	_declspec(dllexport) short APIENTRY BscHostGetVarData(short nCid,short type,short varno,double *p,char *str);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number type Variable Types varno Variable number *p Head pointer to the numeric variable storage area *str Head pointer to the character variable storage area
	OUT (Return) *p Head pointer to the numeric variable storage area *str Head pointer to the character variable storage area
	Return Value 0 : Normal completion Others : Error codes
REMARKS	**Restrictions** This function is effective only for transmission with the YRC1000/ YRC1000micro/DX200/DX100/FS100/NX100/XRC/MRC. String variables can only be used with the YRC1000/YRC1000micro/DX200/DX100/FS100 or NX100 ver3.0 or later.
	Variable Types The variable types are represented as follows. 0 : Byte type 1 : Integer type 2 : Double-precision type 3 : Real type 4 : Robot axis position type 5 : Base axis position type 6 : Station axis position type (pulse type only) 7 : String type

REMARKS

Content of the numeric variable storage area
Depending on the variable type, the numeric variable storage area contains the number of values indicated below.

Variable Type Number	Data Type (Pulse/ XYZ)	Number of values	Content				
			P[0]	P[1]	P[2]	P[3]	P[4]
0	-	1	Byte				
1	-	1	Integer	-	-	-	-
2	-	1	Double	-	-	-	-
3	-	1	Real	-	-	-	-
4	Pulse	8	0	S-axis Pulses	L-axis Pulses	U-axis Pulses	R-axis Pulses
4	XYZ	10	1	Coordinate Type	X-axis (mm)	Y-axis (mm)	Z-axis (mm)
5	Pulse	8	0	Base Axis-1 Pulses	Base Axis-2 Pulses	Base Axis-3 Pulses	Base Axis-4 Pulses
5	XYZ	10	1	Coordinate Type	X-axis (mm)	Y-axis (mm)	Z-axis (mm)
6	Pulse	8	0	Station Axis-1 Pulses	Station Axis-2 Pulses	Station Axis-3 Pulses	Station Axis-4 Pulses

Variable Type Number	Data Type (Pulse/ XYZ)	Number of values	Content				
			P[5]	P[6]	P[7]	P[8]	P[9]
0	-	1					
1	-	1	-	-	-	-	-
2	-	1	-	-	-	-	-
3	-	1	-	-	-	-	-
4	Pulse	8	B-axis Pulses	T-axis Pulses	Tool Number	-	-
4	XYZ	10	Rx Angle(deg)	Ry Angle(deg)	Rz Angle(deg)	Form	Tool Number
5	Pulse	8	Base Axis-5 Pulses	Base Axis-6 Pulses	Tool Number	-	-
5	XYZ	10	Rx Angle(deg)	Ry Angle(deg)	Rz Angle(deg)	Form	Tool Number
6	Pulse	8	Station Axis-5 Pulses	Station Axis-6 Pulses	Tool Number	-	-

The robot axis position and base axis position type variables include the pulse type and XYZ type, according to the first return value.
The station axis position type variable contains the pulse type only.

See below for details on the coordinate system types and form.

Content of the character variable storage area

Variable Type Number	Data Type (Pulse / XYZ)	Number of Values	Content
			str
7	—	16	String

NOTE When this function is used to receive a string type variable make sure that the character variable storage area is allocated for 17 characters.
Declaration in Visual Basic: Dim S_Variable As String *17
Declaration in C++: char S_Variable[17]

REMARKS

Coordinate Types
YRC1000, YRC1000micro, DX200, DX100
The following coordinate names correspond to the coordinate type data.

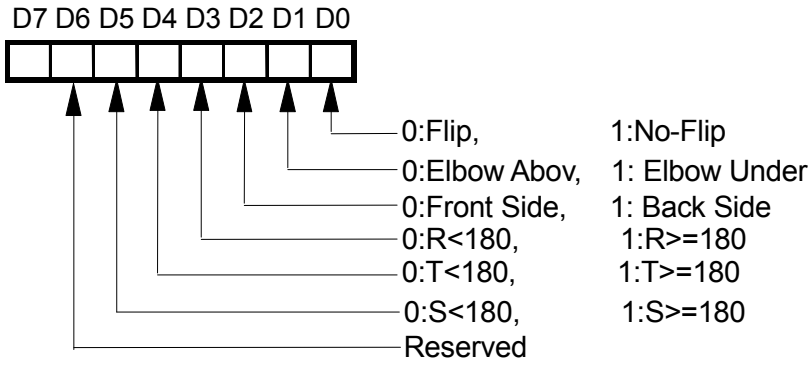
Coordinate type	Coordinate name	Coordinate type	Coordinate name
0	Base coordinate	:	:
1	Robot coordinate	:	:
2	User coordinate 1	63	User coordinate 62
3	User coordinate 2	64	User coordinate 63
:	:	65	Tool coordinate
:	:	66	Master tool coordinate

Coordinate Types
FS100
The following coordinate names correspond to the coordinate type data.

Coordinate type	Coordinate name	Coordinate type	Coordinate name
0	Base coordinate	10	User coordinate 9
1	Robot coordinate	11	User coordinate 10
2	User coordinate 1	12	User coordinate 11
3	User coordinate 2	13	User coordinate 12
4	User coordinate 3	14	User coordinate 13
5	User coordinate 4	15	User coordinate 14
6	User coordinate 5	16	User coordinate 15
7	User coordinate 6	17	User coordinate 16
8	User coordinate 7	18	Tool coordinate
9	User coordinate 8	19	Master tool coordinate

Coordinate Types
NX100/XRC/MRC
The following coordinate names correspond to the coordinate type data.

Coordinate type	Coordinate name	Coordinate type	Coordinate name
0	Base coordinate	14	User coordinate 13
1	Robot coordinate	15	User coordinate 14
2	User coordinate 1	16	User coordinate 15
3	User coordinate 2	17	User coordinate 16
4	User coordinate 3	18	User coordinate 17
5	User coordinate 4	19	User coordinate 18
6	User coordinate 5	20	User coordinate 19
7	User coordinate 6	21	User coordinate 20
8	User coordinate 7	22	User coordinate 21
9	User coordinate 8	23	User coordinate 22
10	User coordinate 9	24	User coordinate 23
11	User coordinate 10	25	User coordinate 24
12	User coordinate 11	26	Tool coordinate
13	User coordinate 12	27	Master tool coordinate

	Form The form data are represented by bit data in decimals.  * With the MRC or MRC2, the data of D5 and D7 are disregarded.
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet)
REFERENCE	"BscHostPutVarData"

■ BscHostGetVarDataM

FUNCTION	Receives multiple variables at the same time.
FORMAT	_declspec(dllexport) short APIENTRY BscHostGetVarDataM(short nCid,short type,short varno,short num,double *p);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number type Variable Types varno Variable number num Number of variables *p Head pointer to the numeric variable storage area
	OUT (Return) *p Head pointer to the numeric variable storage area
	Return Value 0 : Normal completion Others : Error codes
REMARKS	**Restrictions** This function is effective only for transmission with the YRC1000/ YRC1000micro/DX200/DX100/FS100/NX100.
	Variable Types The variable types are represented as follows. 0 : Byte type 1 : Integer type 2 : Double-precision type 3 : Real type
	Variable Designation Method The variable information transmitted is composed of the number of values (num) requested of the specified variable type, beginning with the value of the specified variable number (varno) followed by the values of subsequent variables.
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100 (Serial Port, Ethernet)
REFERENCE	"BscHostPutVarDataM"

■ BscGetVarDataEx

FUNCTION	Receives variables. (robots with 7 axes or more)
FORMAT	_declspec(dllexport) short APIENTRY BscGetVarDataEx(short nCid,short type,short varno,double *p,char *str,short *axisNum);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number type Variable Types varno Variable number *p Head pointer to the numeric variable storage area *str Head pointer to the character variable storage area *axisNum Number of axis (pointer)
	OUT (Return) *p Head pointer to the numeric variable storage area *str Head pointer to the character variable storage area *axisNum Number of axis (pointer)
	Return Value 0 : Normal completion Others : Error codes
REMARKS	**Restrictions** This function is effective only for transmission with the YRC1000/ YRC1000micro/DX200/DX100/FS100/NX100/XRC/MRC. String variables can only be used with the YRC1000/YRC1000micro/DX200/DX100/FS100 or NX100 ver3.0 or later.
	Variable Types The variable types are represented as follows. 0 : Byte type 1 : Integer type 2 : Double-precision type 3 : Real type 4 : Robot axis position type 5 : Base axis position type 6 : Station axis position type (pulse type only) 7 : String type

REMARKS

Content of the numeric variable storage area
Depending on the variable type, the numeric variable storage area contains the number of values indicated below.

Variable Type Number	Data Type (Pulse/ XYZ)	Number of values	Content				
			P[0]	P[1]	P[2]	P[3]	P[4]
0	-	1	Byte	-	-	-	-
1	-	1	Integer	-	-	-	-
2	-	1	Double	-	-	-	-
3	-	1	Real	-	-	-	-
4	Pulse	9	0	S-axis Pulses	L-axis Pulses	U-axis Pulses	R-axis Pulses
4	XYZ	11	1	Coordinate Type	X-axis (mm)	Y-axis (mm)	Z-axis (mm)
5	Pulse	8	0	Base Axis-1 Pulses	Base Axis-2 Pulses	Base Axis-3 Pulses	Base Axis-4 Pulses
5	XYZ	10	1	Coordinate Type	X-axis (mm)	Y-axis (mm)	Z-axis (mm)
6	Pulse	8	0	Station Axis-1 Pulses	Station Axis-2 Pulses	Station Axis-3 Pulses	Station Axis-4 Pulses

Variable Type Number	Data Type (Pulse/ XYZ)	Number of values	Content					
			P[5]	P[6]	P[7]	P[8]	P[9]	P[10]
0	-	1	-	-	-	-	-	-
1	-	1	-	-	-	-	-	-
2	-	1	-	-	-	-	-	-
3	-	1	-	-	-	-	-	-
4	Pulse	9	B-axis Pulses	T-axis Pulses	E-axis Pulses	Tool Number	-	-
4	XYZ	11	Rx Angle(deg)	Ry Angle(deg)	Rz Angle (deg)	Re (deg)	Form	Tool Number
5	Pulse	8	Base Axis-5 Pulses	Base Axis-6 Pulses	Tool Number	-	-	-
5	XYZ	10	Rx Angle (deg)	Ry Angle(deg)	Rz Angle (deg)	Form	Tool Number	-
6	Pulse	8	Station Axis-5 Pulses	Station Axis-6 Pulses	Tool Number	-	-	-

The robot axis position and base axis position type variables include the pulse type and XYZ type, according to the first return value.
The station axis position type variable contains the pulse type only.

See below for details on the coordinate system types and form.

Content of the character variable storage area

Variable Type Number	Data Type (Pulse / XYZ)	Number of Values	Content
			str
7	—	16	String

NOTE

When this function is used to receive a string type variable make sure that the character variable storage area is allocated for 17 characters.
Declaration in Visual Basic: Dim S_Variable As String *17
Declaration in C++: char S_Variable[17]

REMARKS

Coordinate Types
YRC1000, YRC1000micro, DX200, DX100
The following coordinate names correspond to the coordinate type data.

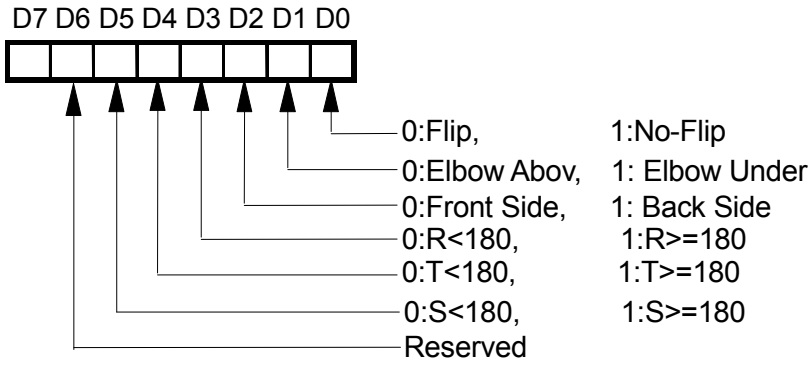
Coordinate type	Coordinate name	Coordinate type	Coordinate name
0	Base coordinate	:	:
1	Robot coordinate	:	:
2	User coordinate 1	63	User coordinate 62
3	User coordinate 2	64	User coordinate 63
:	:	65	Tool coordinate
:	:	66	Master tool coordinate

Coordinate Types
FS100
The following coordinate names correspond to the coordinate type data.

Coordinate type	Coordinate name	Coordinate type	Coordinate name
0	Base coordinate	10	User coordinate 9
1	Robot coordinate	11	User coordinate 10
2	User coordinate 1	12	User coordinate 11
3	User coordinate 2	13	User coordinate 12
4	User coordinate 3	14	User coordinate 13
5	User coordinate 4	15	User coordinate 14
6	User coordinate 5	16	User coordinate 15
7	User coordinate 6	17	User coordinate 16
8	User coordinate 7	18	Tool coordinate
9	User coordinate 8	19	Master tool coordinate

Coordinate Types
NX100/XRC/MRC
The following coordinate names correspond to the coordinate type data.

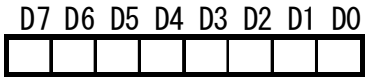
Coordinate type	Coordinate name	Coordinate type	Coordinate name
0	Base coordinate	14	User coordinate 13
1	Robot coordinate	15	User coordinate 14
2	User coordinate 1	16	User coordinate 15
3	User coordinate 2	17	User coordinate 16
4	User coordinate 3	18	User coordinate 17
5	User coordinate 4	19	User coordinate 18
6	User coordinate 5	20	User coordinate 19
7	User coordinate 6	21	User coordinate 20
8	User coordinate 7	22	User coordinate 21
9	User coordinate 8	23	User coordinate 22
10	User coordinate 9	24	User coordinate 23
11	User coordinate 10	25	User coordinate 24
12	User coordinate 11	26	Tool coordinate
13	User coordinate 12	27	Master tool coordinate

	Form The form data are represented by bit data in decimals.  * With the MRC or MRC2, the data of D5 and D7 are disregarded.
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet)
REFERENCE	"BscPutVarDataEx"

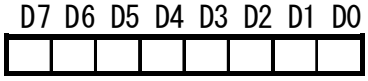
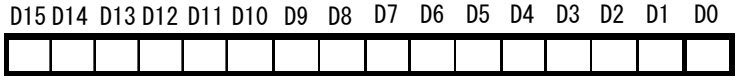
■ BscIsAlarm

FUNCTION	Reads alarm status.
FORMAT	_declspec(dllexport) short APIENTRY BscIsAlarm(short nCid);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number
	OUT (Return) None
	Return Value -1 : Acquisition Failure 0 : No alarm 1 : Alarm
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC(Serial Port)
REFERENCE	"BscGetError2" "BscGetFirstAlarm" "BscGetNextAlarm" "BscGetStatus" "BscIsCycle" "BscIsError" "BscIsHold" "BscIsPlayMode" "BscIsRemoteMode" "BscIsServo" "BscIsTeachMode"

■ BscIsCtrlGroup

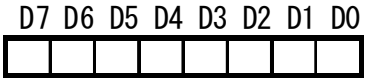
FUNCTION	Reads control group information.
FORMAT	_declspec(dllexport) short APIENTRY BscIsCtrlGroup(short nCid);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number
	OUT (Return) None
	Return Value -1 : Acquisition Failure Others : Control group information
REMARKS	**Restrictions** This function is effective only for transmission with MRC. Refer to BscIsCtrlGroupDX for transmission with YRC1000/ YRC1000micro/DX200/DX100. Refer to BscIsCtrlGroupXrc for transmission with FS100/NX100/ XRC.
	Control Group Information The control group information is represented by bit data in decimals.  D0 : R1 (Robot1) D1 : R2 (Robot2) D2 : S1 (Station1) D3 : S2 (Station2) D4 : S3 (Station3) D5 : S4 (Station4) D6 : S5 (Station5) D7 : S6 (Station6)
CONTROLLER	MRC (Serial Port, Ethernet)
REFERENCE	"BscGetCtrlGroupDX" "BscSetCtrlGroupDX" "BscIsCtrlGroupDX" "BscIsCtrlGroupXrc" "BscGetCtrlGroupXrc" "BscSetCtrlGroupXrc" "BscIsTaskInfXrc" "BscGetCtrlGroup" "BscSetCtrlGroup" "BscIsTaskInf" "BscChangeTask"

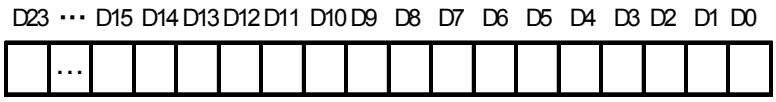
■ BscIsCtrlGroupXrc

FUNCTION	Reads control group information.
FORMAT	_declspec(dllexport) short APIENTRY BscIsCtrlGroupXrc(short nCid,short *robtask,short *stattask);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *robtask Control group information storage pointer (robot axis) *stattask Control group information storage pointer (station axis)
	OUT (Return) *robtask Control group information storage pointer (robot axis) *stattask Control group information storage pointer (station axis)
	Return Value -1 : Acquisition Failure 0 : Normal completion
REMARKS	**Restrictions** This function is effective only for transmission with FS100/NX100/ XRC. Refer to BscIsCtrlGroupDX for transmission with YRC1000/ YRC1000micro/DX200/DX100. Refer to BscIsCtrlGroup for transmission with MRC.
	Control Group Information (Robot Axis) The control group information is represented by bit data in decimals.  D0 : R1 (Robot1) D1 : R2 (Robot2) D2 : R3 (Robot3) D3 : R4 (Robot4)
	Control Group Information (Station Axis) The control group information is represented by bit data in decimals.  D0 : S1 (Station1) D1 : S2 (Station2) D2 : S3 (Station3) D3 : S4 (Station4) D4 : S5 (Station5) D5 : S6 (Station6) D6 : S7 (Station7) D7 : S8 (Station8) D8 : S9 (Station9) D9 : S10 (Station10) D10 : S11 (Station11) D11 : S12 (Station12)

CONTROLLER	FS100, NX100, XRC (Serial Port, Ethernet)
REFERENCE	"BscGetCtrlGroupDX" "BscSetCtrlGroupDX" "BscIsCtrlGroupDX" "BscGetCtrlGroupXrc" "BscSetCtrlGroupXrc" "BscIsTaskInfXrc" "BscIsCtrlGroup" "BscGetCtrlGroup" "BscSetCtrlGroup" "BscIsTaskInf" "BscChangeTask"

■ BscIsCtrlGroupDX

FUNCTION	Reads control group information.
FORMAT	_declspec(dllexport) short APIENTRY BscIsCtrlGroupDX(short nCid,long *robtask,long *stattask);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *robtask Control group information storage pointer (robot axis) *stattask Control group information storage pointer (station axis)
	OUT (Return) *robtask Control group information storage pointer (robot axis) *stattask Control group information storage pointer (station axis)
	Return Value -1 : Acquisition Failure 0 : Normal completion
REMARKS	**Restrictions** This function is effective only for transmission with YRC1000/ YRC1000micro/DX200/DX100. Refer to BscIsCtrlGroupXrc for transmission with FS100/NX100/ XRC. Refer to BscIsCtrlGroup for transmission with MRC.
	Control Group Information (Robot Axis) The control group information is represented by bit data in decimals.  D0 : R1 (Robot1) D1 : R2 (Robot2) D2 : R3 (Robot3) D3 : R4 (Robot4) : : D7 : R8 (Robot8)

REMARKS	**Control Group Information (Station Axis)** The control group information is represented by bit data in decimals.  D0 : S1 (Station1) D1 : S2 (Station2) D2 : S3 (Station3) D3 : S4 (Station4) D4 : S5 (Station5) D5 : S6 (Station6) D6 : S7 (Station7) D7 : S8 (Station8) D8 : S9 (Station9) D9 : S10 (Station10) D10 : S11 (Station11) D11 : S12 (Station12) : : D23 : S24 (Station24)
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100 (Serial Port, Ethernet)
REFERENCE	"BscGetCtrlGroupDX" "BscSetCtrlGroupDX" "BscIsCtrlGroupXrc" "BscGetCtrlGroupXrc" "BscSetCtrlGroupXrc" "BscIsTaskInfXrc" "BscIsCtrlGroup" "BscGetCtrlGroup" "BscSetCtrlGroup" "BscIs-TaskInf" "BscChangeTask"

■ BscIsCycle

FUNCTION	Reads playback mode information.
FORMAT	_declspec(dllexport) short APIENTRY BscIsCycle(short nCid);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number
	OUT (Return) None
	Return Value -1 : Acquisition Failure 0 : Step mode 1 : 1-cycle mode 2 : Auto mode
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	"BscGetStatus" "BscIsAlarm" "BscIsError" "BscIsHold" "BscIsPlayMode" "BscIsRemoteMode" "BscIsServo" "BscIsTeachMode"

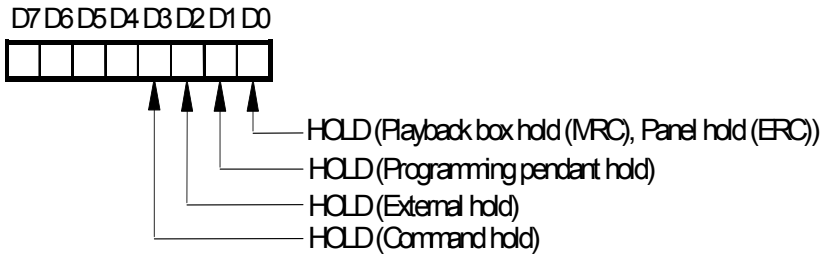
■ BscIsError

FUNCTION	Reads error status.
FORMAT	_declspec(dllexport) short APIENTRY BscIsError(short nCid);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number
	OUT (Return) None
	Return Value -1 : Acquisition failure 0 : No error 1 : Error found
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	"BscGetError2" "BscGetStatus" "BscIsAlarm" "BscIsCycle" "BscIsErrorCode" "BscIsHold" "BscIsPlayMode" "BscIsRemoteMode" "BscIsServo" "BscIsTeachMode"

■ BscIsErrorCode

FUNCTION	Reads the error code.
FORMAT	_declspec(dllexport) short APIENTRY BscIsErrorCode(short nCid);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number
	OUT (Return) None
	Return Value 0 : No error Others : Error codes
	Call Condition Before executing this function, the existence of an error must be confirmed by calling the BscIsError function.
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	"BscGetError2" "BscIsError"

■ BscIsHold

FUNCTION	Reads hold status.
FORMAT	_declspec(dllexport) short APIENTRY BscIsHold(short nCid);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number
	OUT (Return) None
	Return Value -1 : Acquisition Failure 0 : Not held Others : See below
REMARKS	**Hold Status** The hold status data are represented by bit data in decimals. 
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	"BscGetStatus" "BscIsAlarm" "BscIsCycle" "BscIsError" "BscIsPlayMode" "BscIsRemoteMode" "BscIsServo" "BscIsTeachMode"

■ BsclsJobLine

FUNCTION	Reads the current job line number.
FORMAT	_declspec(dllexport) short APIENTRY BsclsJobLine(short nCid);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number
	OUT (Return) None
	Return Value -1 : Acquisition Failure Others : Line numbers
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	"BsclsJobName" "BsclsJobStep"

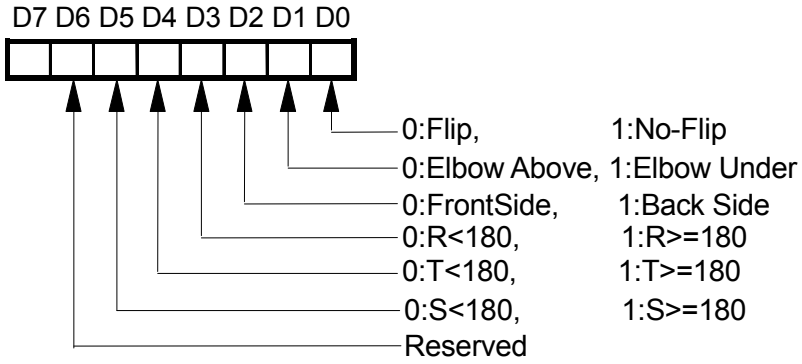
■ BscIsJobName

FUNCTION	Reads the current job name.
FORMAT	_declspec(dllexport) short APIENTRY BscIsJobName(short nCid,char *jobname,short size);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *jobname Job name storage pointer size Job name storage area size
	OUT (Return) *jobname Job name storage pointer
	Return Value -1 : Acquisition Failure 0 : Normal completion
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	"BscIsJobLine" "BscIsJobStep"

■ BscIsJobStep

FUNCTION	Reads the current job step number.
FORMAT	_declspec(dllexport) short APIENTRY BscIsJobStep(short nCid);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number
	OUT (Return) None
	Return Value -1 : Acquisition Failure Others : Step numbers
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	"BscIsJobName" "BscIsJobLine"

■ BsclsLoc

FUNCTION	Reads the current robot position in pulse or XYZ frame system.
FORMAT	_declspec(dllexport) short APIENTRY BsclsLoc(short nCid,short ispulse,short *rconf,double *p);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number ispulse 0 : Cartesian coordinate system (Effective only for MRC and ERC.) 1 : Joint coordinate system *rconf Form storage pointer *p Head pointer to the current position storage area
	OUT (Return) *rconf Form storage pointer *p Head pointer to the current position storage area
	Return Value -1 : Acquisition Failure 0 : Normal completion
REMARKS	**From** The form data are represented by bit data in decimals.  * With the ERC or ERC2, the data from D3 to D7 are disregarded. * With the MRC or MRC2, the data D5 and D7 are disregarded.

REMARKS	Current Position The current position data are as follows when the joint coordinate system or Cartesian coordinate system are specified.

	Joint coordinate system	Cartesian coordinate system
P[0]	S-axis pulse number	X-axis coordinate (unit: mm)
P[1]	L-axis pulse number	Y-axis coordinate (unit: mm)
P[2]	U-axis pulse number	Z-axis coordinate (unit: mm)
P[3]	R-axis pulse number	Wrist angle Rx (unit: °)
P[4]	B-axis pulse number	Wrist angle Ry (unit: °)
P[5]	T-axis pulse number	Wrist angle Rz (unit: °)
P[6]	7th axis pulse number	7th axis pulse number (mm for traveling axis)
P[7]	8th axis pulse number	8th axis pulse number (mm for traveling axis)
P[8]	9th axis pulse number	9th axis pulse number (mm for traveling axis)
P[9]	10th axis pulse number	10th axis pulse number
P[10]	11th axis pulse number	11th axis pulse number
P[11]	12th axis pulse number	12th axis pulse number

CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	"BscIsRobotPos"

■ BscGetPulsePos

FUNCTION	Reads the current robot position in pulse frame system.
FORMAT	_declspec(dllexport) short APIENTRY BscGetPulsePos(short nCid,double *p);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *p Head pointer to the current position storage area
	OUT (Return) *p Head pointer to the current position storage area
	Return Value -1 : Acquisition Failure 0 : Normal completion
REMARKS	Current Position The current position data are as follows when the joint coordinate system or Cartesian coordinate system are specified.
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC (Serial Port, Ethernet)
REFERENCE	"BscGetCartPos"

	Joint coordinate system	Robots with 7 axes
P[0]	S-axis pulse number	S-axis pulse number
P[1]	L-axis pulse number	L-axis pulse number
P[2]	U-axis pulse number	U-axis pulse number
P[3]	R-axis pulse number	R-axis pulse number
P[4]	B-axis pulse number	B-axis pulse number
P[5]	T-axis pulse number	T-axis pulse number
P[6]	7th axis pulse number	E-axis pulse number
P[7]	8th axis pulse number	8th axis pulse number
P[8]	9th axis pulse number	9th axis pulse number
P[9]	10th axis pulse number	10th axis pulse number
P[10]	11th axis pulse number	11th axis pulse number
P[11]	12th axis pulse number	12th axis pulse number
P[12]	-	13th axis pulse number

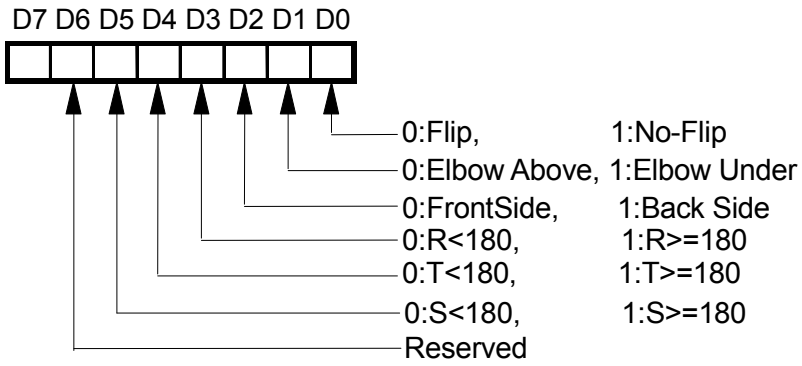
■ BscIsPlayMode

FUNCTION	Reads the operation mode.
FORMAT	_declspec(dllexport) short APIENTRY BscIsPlayMode(short nCid);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number
	OUT (Return) None
	Return Value -1 : Acquisition Failure 0 : Not operating 1 : Operating 2 : Operating at safe speed
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet)
REFERENCE	"BscGetStatus" "BscIsAlarm" "BscIsCycle" "BscIsError" "BscIsRemoteMode" "BscIsServo" "BscIsTeachMode"

■ BscIsRemoteMode

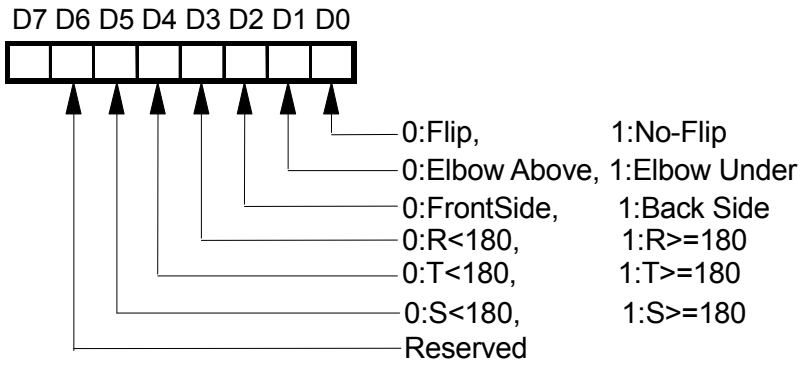
FUNCTION	Reads the command remote mode status.
FORMAT	_declspec(dllexport) short APIENTRY BscIsRemoteMode(short nCid);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number
	OUT (Return) None
	Return Value -1 : Acquisition Failure 0 : Not command remote mode 1 : Command remote mode
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet)
REFERENCE	"BscGetStatus" "BscIsAlarm" "BscIsCycle" "BscIsError" "BscIsHold" "BscIsPlayMode" "BscIsServo" "BscIsTeachMode"

■ BscIsRobotPos

FUNCTION	Reads the current robot position in a specified frame system. The existence of the external axis can also be specified.
FORMAT	_declspec(dllexport) short APIENTRY BscIsRobotPos(short nCid,char *framename,int isex,short *rconf,short *toolno,double *p);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *framename Coordinate name; BASE : Base coordinate system; ROBOT : Robot coordinate system; UF1 : User coordinate system1... isex 0 : No external axis, 1 : With external axis *rconf Form storage pointer *toolno Tool number storage pointer *p Head pointer to the current position storage area
	OUT (Return) *rconf Form storage pointer *toolno Tool number storage pointer *p Head pointer to the current position storage area
	Return Value -1 : Acquisition Failure 0 : Normal completion
REMARKS	**Form** The form data are represented by bit data in decimals.  * With the ERC or ERC2, the data from D3 to D7 are disregarded. * With the MRC or MRC2, the data D5 and D7 are disregarded.

<table>
<tr><td>REMARKS</td><td>Current Position
The current position data are as follows when the joint coordinate system or Cartesian coordinate system is specified.
<table><tr><th></th><th>Current position in the specified coordinate system</th></tr><tr><td>P[0]</td><td>X-axis coordinate (unit: mm)</td></tr><tr><td>P[1]</td><td>Y-axis coordinate (unit: mm)</td></tr><tr><td>P[2]</td><td>Z-axis coordinate (unit: mm)</td></tr><tr><td>P[3]</td><td>Wrist angle Rx (unit: °)</td></tr><tr><td>P[4]</td><td>Wrist angle Ry (unit: °)</td></tr><tr><td>P[5]</td><td>Wrist angle Rz (unit: °)</td></tr><tr><td>P[6]</td><td>7th axis pulse number (mm for traveling axis)</td></tr><tr><td>P[7]</td><td>8th axis pulse number (mm for traveling axis)</td></tr><tr><td>P[8]</td><td>9th axis pulse number (mm for traveling axis)</td></tr><tr><td>P[9]</td><td>10th axis pulse number</td></tr><tr><td>P[10]</td><td>11th axis pulse number</td></tr><tr><td>P[11]</td><td>12th axis pulse number</td></tr></table></td></tr>
<tr><td>CONTROLLER</td><td>YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet)
ERC (Serial Port)</td></tr>
<tr><td>REFERENCE</td><td>"BsclsLoc"</td></tr>
</table>

■ BscGetCartPos

FUNCTION	Reads the current robot position in a specified frame system. The existence of the external axis can also be specified.
FORMAT	_declspec(dllexport) short APIENTRY BscGetCartPos(short nCid,char *framename,int isex,long *rconf,short *toolno,double *p);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *framename Coordinate name; BASE : Base coordinate system; ROBOT : Robot coordinate system; UF1 : User coordinate system1... isex 0 : No external axis, 1 : With external axis *rconf Form storage pointer *toolno Tool number storage pointer *p Head pointer to the current position storage area
	OUT (Return) *rconf Form storage pointer *toolno Tool number storage pointer *p Head pointer to the current position storage area
	Return Value -1 : Acquisition Failure 0 : Normal completion
REMARKS	**Form** The form data are represented by bit data in decimals.  * With the ERC or ERC2, the data from D3 to D7 are disregarded. * With the MRC or MRC2, the data D5 and D7 are disregarded.

REMARKS	**Current Position** The current position data are as follows when the joint coordinate system or Cartesian coordinate system is specified.

	Current position in the specified coordinate system	Robots with 7 axes
P[0]	X-axis coordinate (unit: mm)	X-axis coordinate (unit: mm)
P[1]	Y-axis coordinate (unit: mm)	Y-axis coordinate (unit: mm)
P[2]	Z-axis coordinate (unit: mm)	Z-axis coordinate (unit: mm)
P[3]	Wrist angle Rx (unit: °)	Wrist angle Rx (unit: °)
P[4]	Wrist angle Ry (unit: °)	Wrist angle Ry (unit: °)
P[5]	Wrist angle Rz (unit: °)	Wrist angle Rz (unit: °)
P[6]	7th axis pulse number (mm for traveling axis)	Re (unit: °)
P[7]	8th axis pulse number (mm for traveling axis)	8th axis pulse number (mm for traveling axis)
P[8]	9th axis pulse number (mm for traveling axis)	9th axis pulse number (mm for traveling axis)
P[9]	10th axis pulse number	10th axis pulse number (mm for traveling axis)
P[10]	11th axis pulse number	11th axis pulse number
P[11]	12th axis pulse number	12th axis pulse number
P[12]	-	13th axis pulse number

CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	"BscGetPulsePos"

■ BscIsServo

FUNCTION	Reads the servo status.
FORMAT	_declspec(dllexport) short APIENTRY BscIsServo(short nCid);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number
	OUT (Return) None
	Return Value -1 : Acquisition Failure 0 : Servo OFF 1 : Servo ON
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	"BscGetStatus" "BscIsAlarm" "BscIsCycle" "BscIsError" "BscIsHold" "BscIsPlayMode" "BscIsRemoteMode" "BscIsTeachMode" "BscServoOn" "BscServoOff"

■ BscIsTaskInf

FUNCTION	Reads task information.
FORMAT	_declspec(dllexport) short APIENTRY BscIsTaskInf(short nCid);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number
	OUT (Return) None
	Return Value -1 : Acquisition Failure Others : Task information
REMARKS	**Restrictions** This function is effective only for transmission with the MRC. Refer to BscIsTaskInfXrc for transmission with the YRC1000/ YRC1000micro/DX200/DX100/FS100/NX100/XRC.
	Task Information The task information is represented as follows. 0 : Master task 1 : Sub 1 task 2 : Sub 2 task "0" is returned when independent control is not allowed in the system.
CONTROLLER	MRC (Serial Port, Ethernet)
REFERENCE	"BscIsTaskInfXrc" "BscGetCtrlGroupXrc" "BscSetCtrlGroupXrc" "BscIsCtrlGroupXrc" "BscGetCtrlGroup" "BscSetCtrlGroup" "BscIsCtrlGroup" "BscChangeTask"

■ BscIsTaskInfXrc

FUNCTION	Reads task information.
FORMAT	_declspec(dllexport) short APIENTRY BscIsTaskInfXrc(short nCid);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number
	OUT (Return) None
	Return Value -1 : Acquisition Failure Others : Error codes
REMARKS	**Restrictions** This function is effective for transmission with the YRC1000/ YRC1000micro/DX200/DX100/FS100/NX100/XRC. Refer to BscIsTaskInf for transmission with the MRC.
	Task Information The task information is represented as follows. 0 : Master task 1 : Sub 1 task 2 : Sub 2 task 3 : Sub 3 task 4 : Sub 4 task 5 : Sub 5 task 6 : Sub 6 task 7 : Sub 7 task "0" is returned if independent control is not allowed in the system.
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC (Serial Port, Ethernet)
REFERENCE	"BscGetCtrlGroupXrc" "BscSetCtrlGroupXrc" "BscIsCtrlGroupXrc" "BscIsTaskInf" "BscGetCtrlGroup" "BscSetCtrlGroup" "BscIsCtrlGroup" "BscChangeTask"

■ BscIsTeachMode

FUNCTION	Reads whether in the teach mode or play mode.
FORMAT	_declspec(dllexport) short APIENTRY BscIsTeachMode(short nCid);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number
	OUT (Return) None
	Return Value -1 : Acquisition Failure 0 : Teach mode 1 : Play mode
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet)
REFERENCE	"BscGetStatus" "BscIsAlarm" "BscIsCycle" "BscIsError" "BscIsHold" "BscIsPlayMode" "BscIsRemoteMode" "BscIsServo"

■ BscJobWait

FUNCTION	Waits for job completion until the robot stops or specified time expires.
FORMAT	_declspec(dllexport) short APIENTRY BscJobWait(short nCid,short time);
ARGUMENTS	I**N (Transfer)** nCid Communication handler ID number time Wait time (-1 : Unlimited; 0 to 32767seconds)
	OUT (Return) None
	Return Value -2 : Abnormal completion -1 : Operation incomplete 0 : Operation completed Others : Error codes
REMARKS	**Cause of Incomplete Operation** These causes are considered for imcomplete operation. * The robot was stopped via teach pendant or by external hold. * The robot was stopped by alarm. * The robot was stopped by emergency stop. * Time expired
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)

■ BscReadAlarmS

FUNCTION	Reads the error code, error data and error message.
FORMAT	_declspec(dllexport) short APIENTRY BscReadAlarmS(short nCid,short *data,char *msg);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *data Error data storage pointer *msg Error message storage pointer
	OUT (Return) *data Error data storage pointer *msg Error message storage pointer
	Return Value -1 : Error code acquisition failure 0 : No Error Others : Error codes numbers
REMARKS	**Restrictions** This function is effective for transmission with the YRC1000/ YRC1000micro/DX200/DX100/FS100/NX100.
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100 (Serial Port, Ethernet)
REFERENCE	"BscGetFirstAlarmS" "BscGetNextAlarmS" "BscGetError2"

■ BscGetTorque

FUNCTION	Reads torque value.
FORMAT	_declspec(dllexport) BscGetTorque(short nCid,double *p);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *p Head pointer to the torque value storage area
	OUT (Return) *p Head pointer to the torque value storage area
	Return Value -1 : Acquisition Failure 0 : Normal completion
REMARKS	**Restrictions** This function is effective only for transmission with YRC1000, YRC1000micro, DX200 verDN1.60-00 or later.
	Torque Value The torque value data are as follows when the joint coordinate system.

	Joint coordinate system	Robots with 7 axes
P[0]	S-axis motor torque	S-axis motor torque
P[1]	L-axis motor torque	L-axis motor torque
P[2]	U-axis motor torque	U-axis motor torque
P[3]	R-axis motor torque	R-axis motor torque
P[4]	B-axis motor torque	B-axis motor torque
P[5]	T-axis motor torque	T-axis motor torque
P[6]	7th axis motor torque	E-axis motor torque
P[7]	8th axis motor torque	8th axis motor torque
P[8]	9th axis motor torque	9th axis motor torque
P[9]	10th axis motor torque	10th axis motor torque
P[10]	11th axis motor torque	11th axis motor torque
P[11]	12th axis motor torque	12th axis motor torque
P[12]	-	13th axis motor torque

CONTROLLER	YRC1000, YRC1000micro, DX200 verDN1.60-00 or later.
REFERENCE	"BscGetMaxTorque"

■ BscGetMaxTorque

FUNCTION	Reads max torque value.
FORMAT	_declspec(dllexport) BscGetMaxTorque(short nCid,double *p);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *p Head pointer to the max torque value storage area
	OUT (Return) *p Head pointer to the max torque value storage area
	Return Value -1 : Acquisition Failure 0 : Normal completion
REMARKS	**Restrictions** This function is effective only for transmission with YRC1000, YRC1000micro, DX200 verDN1.60-00 or later.
	Max Torque Value The max torque value data are as follows when the joint coordinate system.

	Joint coordinate system	Robots with 7 axes
P[0]	S-axis max motor torque	S-axis max motor torque
P[1]	L-axis max motor torque	L-axis max motor torque
P[2]	U-axis max motor torque	U-axis max motor torque
P[3]	R-axis max motor torque	R-axis max motor torque
P[4]	B-axis max motor torque	B-axis max motor torque
P[5]	T-axis max motor torque	T-axis max motor torque
P[6]	7th axis max motor torque	E-axis max motor torque
P[7]	8th axis max motor torque	8th axis max motor torque
P[8]	9th axis max motor torque	9th axis max motor torque
P[9]	10th axis max motor torque	10th axis max motor torque
P[10]	11th axis max motor torque	11th axis max motor torque
P[11]	12th axis max motor torque	12th axis max motor torque
P[12]	-	13th axis max motor torque

CONTROLLER	YRC1000, YRC1000micro, DX200 verDN1.60-00 or later.
REFERENCE	"BscGetTorque"

■ BscGetEncTmp

FUNCTION	Reads encoder temperature.
FORMAT	_declspec(dllexport) BscGetEncTmp(short nCid,double *p);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *p Head pointer to the encoder temperature value storage area
	OUT (Return) *p Head pointer to the encoder temperature value storage area
	Return Value -1 : Acquisition Failure 0 : Normal completion
REMARKS	**Restrictions** This function is effective only for transmission with YRC1000, YRC1000micro, DX200 verDN1.60-00 or later.
	Encoder Temperature The encoder temperature value data are as follows when the joint coordinate system.

	Joint coordinate system	Robots with 7 axes
P[0]	S-axis encoder temperature	S-axis encoder temperature
P[1]	L-axis encoder temperature	L-axis encoder temperature
P[2]	U-axis encoder temperature	U-axis encoder temperature
P[3]	R-axis encoder temperature	R-axis encoder temperature
P[4]	B-axis encoder temperature	B-axis encoder temperature
P[5]	T-axis encoder temperature	T-axis encoder temperature
P[6]	7th axis encoder temperature	E-axis encoder temperature
P[7]	8th axis encoder temperature	8th axis encoder temperature
P[8]	9th axis encoder temperature	9th axis encoder temperature
P[9]	10th axis encoder temperature	10th axis encoder temperature
P[10]	11th axis encoder temperature	11th axis encoder temperature
P[11]	12th axis encoder temperature	12th axis encoder temperature
P[12]	-	13th axis encoder temperature

CONTROLLER	YRC1000, YRC1000micro, DX200 verDN1.60-00 or later.

■ BscGetSysTime

FUNCTION	Reads system monitoring time.
FORMAT	_declspec(dllexport) BscGetSysTime(short nCid,char *contpwr, char *svpwr, char *playback, char *mov, char *ope);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *contpwr Head pointer to the controller power time value storage area * svpwr Head pointer to the servo power time value storage area * playback Head pointer to the playback time value storage area * mov Head pointer to the moving time value storage area * ope Head pointer to the operating time value storage area
	OUT (Return) *contpwr Head pointer to the controller power time value storage area * svpwr Head pointer to the servo power time value storage area * playback Head pointer to the playback time value storage area * mov Head pointer to the moving time value storage area * ope Head pointer to the operating time value storage area
	Return Value -1 : Acquisition Failure 0 : Normal completion
REMARKS	**Restrictions** This function is effective only for transmission with YRC1000, YRC1000micro, DX200 verDN1.60-00 or later.
	System Monitoring Time Output formats of each times are follows. HHHHH:MM'SS HHHHH=Hour, MM=Minute, SS=Second
CONTROLLER	YRC1000, YRC1000micro, DX200 verDN1.60-00 or later.

■ BscCancel

FUNCTION	Cancels an error.
FORMAT	_declspec(dllexport) short APIENTRY BscCancel(short nCid);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number
	OUT (Return) None
	Return Value 0 : Normal completion Others : Error codes
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	"BscReset"

■ BscChangeTask

FUNCTION	Changes a task.
FORMAT	_declspec(dllexport) short APIENTRY BscChangeTask(short nCid,short task);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number task Specified task number
	OUT (Return) None
	Return Value 0 : Normal completion Others : Error codes
REMARKS	**Restrictions** This function is effective only for transmission with the YRC1000/ YRC1000micro/DX200/DX100/FS100/NX100/XRC/MRC. When the power supply of robot controller is started up, a master task is selected as the control task. This function cannot be used in a system where independent control is not enabled.
	Specified Task Number The task number is represented as follows. 0 : Task 1 : Sub 1 task 2 : Sub 2 task 3 : Sub 3 task 4 : Sub 4 task 5 : Sub 5 task 6 : Sub 6 task 7 : Sub 7 task * Specified task number 3 to 7 are only for the YRC1000 and YRC1000micro and DX200 and DX100 and NX100 and XRC. * Specified task number 3 to 5 are only for the FS100.
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet)
REFERENCE	"BscGetCtrlGroupXrc" "BscSetCtrlGroupXrc" "BscIsCtrlGroupXrc" "BscIsTaskInfXrc" "BscGetCtrlGroup" "BscSetCtrlGroup" "BscIsCtrlGroup" "BscIsTaskInf"

■ BscContinueJob

FUNCTION	Starts job. (Execution starts from the current line of the current job.)
FORMAT	_declspec(dllexport) short APIENTRY BscContinueJob(short nCid);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number
	OUT (Return) None
	Return Value 0 : Normal completion Others : Error codes
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	"BscStartJob"

■ BscConvertJobP2R

FUNCTION	Converts a pulse job to a relative job in a specified frame system.
FORMAT	_declspec(dllexport) short APIENTRY BscConvertJobP2R(short nCid,char *name,char *framename);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *name Job name storage pointer *framename Coordinate name: BASE : Base coordinate, ROBOT : Robot coordinate, UF1 : User coordinate1 ...
	OUT (Return) None
	Return Value 0 : Normal completion Others : Error codes
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	"BscConvertJobR2P"

■ BscConvertJobR2P

FUNCTION	Converts a relative job in a specified frame system to a pulse job.
FORMAT	_declspec(dllexport) short APIENTRY BscConvertJobR2P(short nCid,char *name,short cv_type,char *var_no);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *name Job name storage pointer cv_type Conversion method var_no Reference position variable number
	OUT (Return) None
	Return Value 0 : Normal completion Others : Error codes
REMARKS	**Restrictions** This function is effective only for transmission with the YRC1000/ YRC1000micro/DX200/DX100/FS100/NX100/XRC/MRC.
	Conversion Method The conversion method is represented as follows. 0 : Previous step regarded (B-axis sign constant) 1 : Type regarded 2 : Previous step regarded (Minimum R-axis movement)
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet)
REFERENCE	"BscConvertJobP2R"

■ BscDeleteJob

FUNCTION	Deletes a job.
FORMAT	_declspec(dllexport) short APIENTRY BscDeleteJob(short nCid);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number
	OUT (Return) None
	Return Value 0 : Normal completion 1 : Cannot delete Otherss : Error codes
REMARKS	**Call Condition** The BscSelectJob function must be called up and the job name to be deleted must be specified before executing this function.
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	"BscSelectJob" "BscSetMasterJob"

■ BscHoldOff

FUNCTION	Sets hold-OFF.
FORMAT	_declspec(dllexport) short APIENTRY BscHoldOff(short nCid);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number
	OUT (Return) None
	Return Value 0 : Normal completion Others : Error codes
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	BscHoldOn

■ BscHoldOn

FUNCTION	Sets hold-ON.
FORMAT	_declspec(dllexport) short APIENTRY BscHoldOn(short nCid);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number
	OUT (Return) None
	Return Value 0 : Normal completion Others : Error codes
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	"BscHoldOff"

■ BscHostPutVarData

FUNCTION	Sets variables.
FORMAT	_declspec(dllexport) short APIENTRY BscHostPutVarData(short nCid,short type,short varno,double *p, char *str);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number type Variable Types varno Variable number *p Head pointer to the numeric variable storage area *str Head pointer to the character variable storage area
	OUT (Return) None
	Return Value 0 : Normal completion Others : Error codes
REMARKS	**Restrictions** This function is effective only for transmission with the YRC1000/ YRC1000micro/DX200/DX100/FS100/NX100/XRC/MRC. String variables can only be used with the YRC1000 or YRC1000micro or DX200 or DX100 or FS100 or NX100 ver3.0 or later.
	Variable Types The variable types are represented as follows. 0 : Byte type 1 : Integer type 2 : Double-precision type 3 : Real type 4 : Robot axis position type 5 : Base axis position type 6 : Station axis position type (pulse type only) 7 : String type

REMARKS

Content of the numeric variable storage area

Depending on the variable type, the numeric variable storage area contains the number of values indicated below.

Variable Type Number	Data Type (Pulse/ XYZ)	Number of values	Content				
			P[0]	P[1]	P[2]	P[3]	P[4]
0	-	1	Byte				
1	-	1	Integer	-	-	-	-
2	-	1	Double	-	-	-	-
3	-	1	Real	-	-	-	-
4	Pulse	8	0	S-axis Pulses	L-axis Pulses	U-axis Pulses	R-axis Pulses
4	XYZ	10	1	Coordinate Type	X-axis (mm)	Y-axis (mm)	Z-axis (mm)
5	Pulse	8	0	Base Axis-1 Pulses	Base Axis-2 Pulses	Base Axis-3 Pulses	Base Axis-4 Pulses
5	XYZ	10	1	Coordinate Type	X-axis (mm)	Y-axis (mm)	Z-axis (mm)
6	Pulse	8	0	Station Axis-1 Pulses	Station Axis-2 Pulses	Station Axis-3 Pulses	Station Axis-4 Pulses

Variable Type Number	Data Type (Pulse/ XYZ)	Number of values	Content				
			P[5]	P[6]	P[7]	P[8]	P[9]
0	-	1					
1	-	1	-	-	-	-	-
2	-	1	-	-	-	-	-
3	-	1	-	-	-	-	-
4	Pulse	8	B-axis Pulses	T-axis Pulses	Tool Number	-	-
4	XYZ	10	Rx Angle(deg)	Ry Angle(deg)	Rz Angle(deg)	Form	Tool Number
5	Pulse	8	Base Axis-5 Pulses	Base Axis-6 Pulses	Tool Number	-	-
5	XYZ	10	Rx Angle(deg)	Ry Angle(deg)	Rz Angle(deg)	Form	Tool Number
6	Pulse	8	Station Axis-5 Pulses	Station Axis-6 Pulses	Tool Number	-	-

The robot axis position and base axis position type variables include the pulse type and XYZ type, according to the first return value. The station axis position type variable contains the pulse type only. See below for details on the coordinate system types and form.

Content of the character variable storage area

Variable Type Number	Data Type (Pulse / XYZ)	Number of Values	Content
			str
7	—	16	String

NOTE

When this function is used to receive a string type variable make sure that the character variable storage area is allocated for 17 characters.

Declaration in Visual Basic: Dim S_Variable As String *17

Declaration in C++: char S_Variable[17]

REMARKS

Coordinate Types
YRC1000, YRC1000micro, DX200, DX100
The following coordinate names correspond to the coordinate type data.

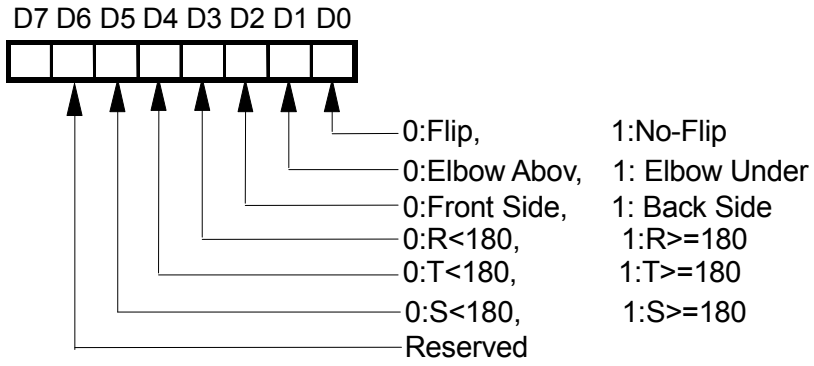
Coordinate type	Coordinate name	Coordinate type	Coordinate name
0	Base coordinate	:	:
1	Robot coordinate	:	:
2	User coordinate 1	63	User coordinate 62
3	User coordinate 2	64	User coordinate 63
:	:	65	Tool coordinate
:	:	66	Master tool coordinate

Coordinate Types
FS100
The following coordinate names correspond to the coordinate type data.

Coordinate type	Coordinate name	Coordinate type	Coordinate name
0	Base coordinate	10	User coordinate 9
1	Robot coordinate	11	User coordinate 10
2	User coordinate 1	12	User coordinate 11
3	User coordinate 2	13	User coordinate 12
4	User coordinate 3	14	User coordinate 13
5	User coordinate 4	15	User coordinate 14
6	User coordinate 5	16	User coordinate 15
7	User coordinate 6	17	User coordinate 16
8	User coordinate 7	18	Tool coordinate
9	User coordinate 8	19	Master tool coordinate

Coordinate Types
NX100/XRC/MRC
The following coordinate names correspond to the coordinate type data.

Coordinate type	Coordinate name	Coordinate type	Coordinate name
0	Base coordinate	14	User coordinate 13
1	Robot coordinate	15	User coordinate 14
2	User coordinate 1	16	User coordinate 15
3	User coordinate 2	17	User coordinate 16
4	User coordinate 3	18	User coordinate 17
5	User coordinate 4	19	User coordinate 18
6	User coordinate 5	20	User coordinate 19
7	User coordinate 6	21	User coordinate 20
8	User coordinate 7	22	User coordinate 21
9	User coordinate 8	23	User coordinate 22
10	User coordinate 9	24	User coordinate 23
11	User coordinate 10	25	User coordinate 24
12	User coordinate 11	26	Tool coordinate
13	User coordinate 12	27	Master tool coordinate

	Form The form data are represented by bit data in decimals.  * With the MRC or MRC2, the data of D5 and D7 are disregarded.
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet)
REFERENCE	"BscHostGetVarData"

■ BscHostPutVarDataM

FUNCTION	Sets multiple variables at the same time.
FORMAT	_declspec(dllexport) short APIENTRY BscHostPutVarDataM(short nCid,short type,short varno, short num, double *p);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number type Variable Types varno Variable number num Number of variables *p Head pointer to the numeric variable storage area
	OUT (Return) None
	Return Value 0 : Normal completion Others : Error codes
REMARKS	**Restrictions** This function is effective only for transmission with the YRC1000/ YRC1000micro/DX200/DX100/FS100/NX100.
	Variable Types The variable types are represented as follows. 0 : Byte type 1 : Integer type 2 : Double-precision type 3 : Real type
	Variable Designation Method The variable information transmitted is composed of the number of values (num) requested of the specified variable type, beginning with the value of the specified variable number (varno) followed by the values of subsequent variables.
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100 (Serial Port, Ethernet)
REFERENCE	"BscHostGetVarDataM"

■ BscPutVarDataEx

FUNCTION	Sets variables.(robots with 7 axes or more)
FORMAT	_declspec(dllexport) short APIENTRY BsctPutVarDataEx(short nCid,short type,short varno,double *p, char *str,short axisNum);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number type Variable Types varno Variable number *p Head pointer to the numeric variable storage area *str Head pointer to the character variable storage area axisNum Number of axis
	OUT (Return) None
	Return Value 0 : Normal completion Others : Error codes
REMARKS	**Restrictions** This function is effective only for transmission with the YRC1000/ YRC1000micro/DX200/DX100/FS100/NX100/XRC/MRC. String variables can only be used with the YRC1000/YRC1000micro/DX200/DX100/FS100 or NX100 ver3.0 or later.
	Variable Types The variable types are represented as follows. 0 : Byte type 1 : Integer type 2 : Double-precision type 3 : Real type 4 : Robot axis position type 5 : Base axis position type 6 : Station axis position type (pulse type only) 7 : String type

REMARKS

Content of the numeric variable storage area

Depending on the variable type, the numeric variable storage area contains the number of values indicated below.

Variable Type Number	Data Type (Pulse/ XYZ)	Number of values	Content				
			P[0]	P[1]	P[2]	P[3]	P[4]
0	-	1	Byte				
1	-	1	Integer	-	-	-	-
2	-	1	Double	-	-	-	-
3	-	1	Real	-	-	-	-
4	Pulse	9	0	S-axis Pulses	L-axis Pulses	U-axis Pulses	R-axis Pulses
4	XYZ	11	1	Coordinate Type	X-axis (mm)	Y-axis (mm)	Z-axis (mm)
5	Pulse	8	0	Base Axis-1 Pulses	Base Axis-2 Pulses	Base Axis-3 Pulses	Base Axis-4 Pulses
5	XYZ	10	1	Coordinate Type	X-axis (mm)	Y-axis (mm)	Z-axis (mm)
6	Pulse	8	0	Station Axis-1 Pulses	Station Axis-2 Pulses	Station Axis-3 Pulses	Station Axis-4 Pulses

Variable Type Number	Data Type (Pulse/ XYZ)	Number of values	Content					
			P[5]	P[6]	P[7]	P[8]	P[9]	P[10]
0	-	1						
1	-	1	-	-	-	-	-	-
2	-	1	-	-	-	-	-	-
3	-	1	-	-	-	-	-	-
4	Pulse	9	B-axis Pulses	T-axis Pulses	E-axis Pulses	Tool Number	-	-
4	XYZ	11	Rx Angle(deg)	Ry Angle(deg)	Rz Angle(deg)	Re (deg)	Form	Tool Number
5	Pulse	8	Base Axis-5 Pulses	Base Axis-6 Pulses	Tool Number	-	-	-
5	XYZ	10	Rx Angle(deg)	Ry Angle(deg)	Rz Angle(deg)	Form	Tool Number	-
6	Pulse	8	Station Axis-5 Pulses	Station Axis-6 Pulses	Tool Number	-	-	-

The robot axis position and base axis position type variables include the pulse type and XYZ type, according to the first return value.
The station axis position type variable contains the pulse type only.
See below for details on the coordinate system types and form.

Content of the character variable storage area

Variable Type Number	Data Type (Pulse / XYZ)	Number of Values	Content
			str
7	—	16	String

NOTE When this function is used to receive a string type variable make sure that the character variable storage area is allocated for 17 characters.
Declaration in Visual Basic: Dim S_Variable As String *17
Declaration in C++: char S_Variable[17]

REMARKS

Coordinate Types
YRC1000, YRC1000micro, DX200, DX100
The following coordinate names correspond to the coordinate type data.

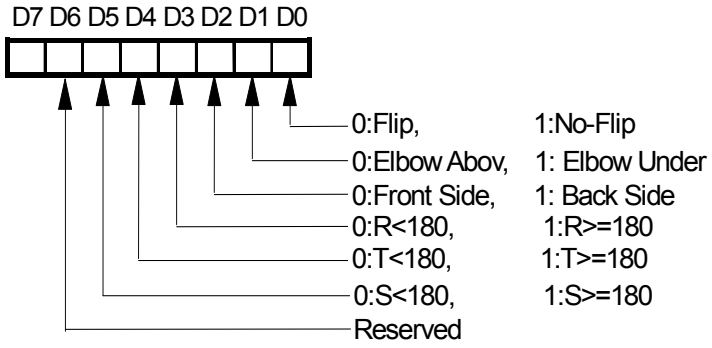
Coordinate type	Coordinate name	Coordinate type	Coordinate name
0	Base coordinate	:	:
1	Robot coordinate	:	:
2	User coordinate 1	63	User coordinate 62
3	User coordinate 2	64	User coordinate 63
:	:	65	Tool coordinate
:	:	66	Master tool coordinate

Coordinate Types
FS100
The following coordinate names correspond to the coordinate type data.

Coordinate type	Coordinate name	Coordinate type	Coordinate name
0	Base coordinate	10	User coordinate 9
1	Robot coordinate	11	User coordinate 10
2	User coordinate 1	12	User coordinate 11
3	User coordinate 2	13	User coordinate 12
4	User coordinate 3	14	User coordinate 13
5	User coordinate 4	15	User coordinate 14
6	User coordinate 5	16	User coordinate 15
7	User coordinate 6	17	User coordinate 16
8	User coordinate 7	18	Tool coordinate
9	User coordinate 8	19	Master tool coordinate

Coordinate Types
NX100/XRC/MRC
The following coordinate names correspond to the coordinate type data.

Coordinate type	Coordinate name	Coordinate type	Coordinate name
0	Base coordinate	14	User coordinate 13
1	Robot coordinate	15	User coordinate 14
2	User coordinate 1	16	User coordinate 15
3	User coordinate 2	17	User coordinate 16
4	User coordinate 3	18	User coordinate 17
5	User coordinate 4	19	User coordinate 18
6	User coordinate 5	20	User coordinate 19
7	User coordinate 6	21	User coordinate 20
8	User coordinate 7	22	User coordinate 21
9	User coordinate 8	23	User coordinate 22
10	User coordinate 9	24	User coordinate 23
11	User coordinate 10	25	User coordinate 24
12	User coordinate 11	26	Tool coordinate
13	User coordinate 12	27	Master tool coordinate

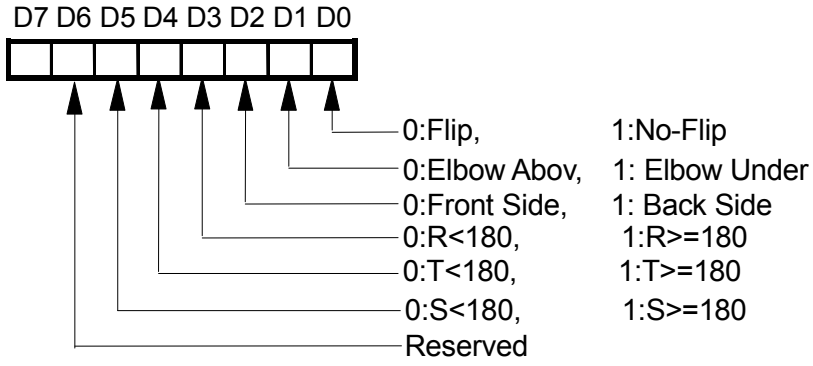
REMARKS	**Form** The form data are represented by bit data in decimals.  * With the MRC or MRC2, the data of D5 and D7 are disregarded.
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet)
REFERENCE	"BscGetVarDataEx"

■ BsclmovFUNCTION

FUNCTION	Moves robot with linear motion form the current position for the increment value in a specified frame system.
FORMAT	_declspec(dllexport) short APIENTRY BsclmovFUNCTION(short nCid,char *vtype,double spd,char *framename,short toolno,double *p);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *vtype Move speed selection; V : Control point; VR : Position angular spd Move speed (0.1 to ****.*mm/s, 0.1 to ***.* ° /s) *framename Coordinate name; BASE : Base coordinate; ROBOT : Robot coordinate; UF1 : User coordinate1... TOOL : Tool coordinate (Only for NX100/XRC/MRC) toolno Tool number *p Head pointer to the target position storage area
	OUT (Return) None
	Return Value 0 : Normal completion Others : Error codes
REMARKS	**Target Position** The target position is represented as follows.

	Target position in the specified coordinate system
P[0]	X-axis coordinate (unit: mm)
P[1]	Y-axis coordinate (unit: mm)
P[2]	Z-axis coordinate (unit: mm)
P[3]	Wrist angle Rx (unit: °)
P[4]	Wrist angle Ry (unit: °)
P[5]	Wrist angle Rz (unit: °)
P[6]	7th axis pulse number (mm for traveling axis)
P[7]	8th axis pulse number (mm for traveling axis)
P[8]	9th axis pulse number (mm for traveling axis)
P[9]	10th axis pulse number
P[10]	11th axis pulse number
P[11]	12th axis pulse number

* Set "0" for data P[7] to P[11] if the system has no external axis.

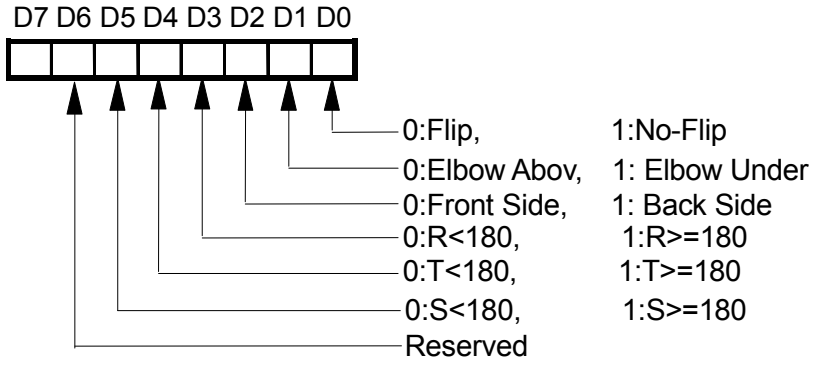
REMARKS	**Form** The form data are represented by bit data in decimals.  * With the ERC or ERC2, the data from D3 to D7 are disregarded. * With the MRC or MRC2, the data D5 and D7 are disregarded.
CONTROLLER	NX100, XRC, MRC (Serial Port, Ethernet), ERC (Serial Port) * Refer to the BscImovEx2 for the YRC1000/YRC1000micro/ DX200/DX100/FS100.
REFERENCE	"BscMov" "BscMovEx" "BscMovl" "BscImovEx2"

■ BsclmovEx

FUNCTION	Moves robot with linear motion form the current position for the increment value in a specified frame system.(robots with 7 axes or more)
FORMAT	_declspec(dllexport) short APIENTRY BsclmovEx(short nCid,char *vtype,double spd,char *framename,short toolno,double *p,short axisNum);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *vtype Move speed selection; V : Control point; VR : Position angular spd Move speed (0.1 to ****.*mm/s, 0.1 to ***.* ° /s) *framename Coordinate name; BASE : Base coordinate; ROBOT : Robot coordinate; UF1 : User coordinate1... TOOL : Tool coordinate toolno Tool number *p Head pointer to the target position storage area axisNum Number of axis
	OUT (Return) None
	Return Value 0 : Normal completion Others : Error codes
REMARKS	**Restrictions** This function is to keep compatibility. Please use BsclmovEx2.
	Target Position The target position is represented as follows.

	Target position in the specified coordinate system	Robots with 7 axes
P[0]	X-axis coordinate (unit: mm)	X-axis coordinate (unit: mm)
P[1]	Y-axis coordinate (unit: mm)	Y-axis coordinate (unit: mm)
P[2]	Z-axis coordinate (unit: mm)	Z-axis coordinate (unit: mm)
P[3]	Wrist angle Rx (unit: °)	Wrist angle Rx (unit: °)
P[4]	Wrist angle Ry (unit: °)	Wrist angle Ry (unit: °)
P[5]	Wrist angle Rz (unit: °)	Wrist angle Rz (unit: °)
P[6]	7th axis pulse number (mm for traveling axis)	Re (unit: °)
P[7]	8th axis pulse number (mm for traveling axis)	8th axis pulse number (mm for traveling axis)
P[8]	9th axis pulse number (mm for traveling axis)	9th axis pulse number (mm for traveling axis)
P[9]	10th axis pulse number	10th axis pulse number (mm for traveling axis)
P[10]	11th axis pulse number	11th axis pulse number
P[11]	12th axis pulse number	12th axis pulse number
P[12]	-	13th axis pulse number

* Set "0" for data P[6] to P[11] if the system has no external axis.
Set "0" for data P[7] to P[12] if the system has robots with 7 axes.

REMARKS	**Form** The form data are represented by bit data in decimals. 
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100 (Serial Port, Ethernet)
REFERENCE	"BscMovEx2" "BscMovjEx" "BscMovlEx"

■ BscImovEx2

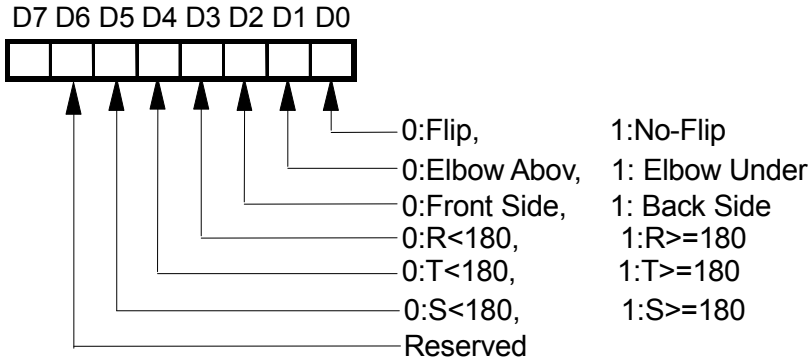
FUNCTION	Moves robot with linear motion form the current position for the increment value in a specified frame system.(robots with 7 axes or more)
FORMAT	_declspec(dllexport) short APIENTRY BscImovEx(short nCid,short ctype,char *vtype,double spd,char *framename,short toolno,double *p,short axisNum);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number ctype Controller selection; 0 : ERC, 1 : MRC, 2 : XRC, 3 : NX100 , 4 : YRC1000, YRC1000micro, DX200, DX100, 5 : FS100 *vtype Move speed selection; V : Control point; VR : Position angular spd Move speed (0.1 to ****.*mm/s, 0.1 to ***.* ° /s) *framename Coordinate name; BASE : Base coordinate; ROBOT : Robot coordinate; UF1 : User coordinate1... TOOL : Tool coordinate (Only for YRC1000/YRC1000micro/ DX200/DX100/FS100/ NX100/XRC/MRC) toolno Tool number *p Head pointer to the target position storage area axisNum Number of robot axes
	OUT (Return) None
	Return Value 0 : Normal completion Others : Error codes
REMARKS	**Restrictions** YRC1000 and YRC1000micro and DX200 and DX100 and FS100 only can be used robot with 7 axes or more.

REMARKS	**Target Position** The target position is represented as follows.

	Target position in the specified coordinate system	Robots with 7 axes
P[0]	X-axis coordinate (unit: mm)	X-axis coordinate (unit: mm)
P[1]	Y-axis coordinate (unit: mm)	Y-axis coordinate (unit: mm)
P[2]	Z-axis coordinate (unit: mm)	Z-axis coordinate (unit: mm)
P[3]	Wrist angle Rx (unit: °)	Wrist angle Rx (unit: °)
P[4]	Wrist angle Ry (unit: °)	Wrist angle Ry (unit: °)
P[5]	Wrist angle Rz (unit: °)	Wrist angle Rz (unit: °)
P[6]	7th axis pulse number (mm for traveling axis)	Re (unit: °)
P[7]	8th axis pulse number (mm for traveling axis)	8th axis pulse number (mm for traveling axis)
P[8]	9th axis pulse number (mm for traveling axis)	9th axis pulse number (mm for traveling axis)
P[9]	10th axis pulse number	10th axis pulse number (mm for traveling axis)
P[10]	11th axis pulse number	11th axis pulse number
P[11]	12th axis pulse number	12th axis pulse number
P[12]	-	13th axis pulse number

* Set "0" for data P[6] to P[11] if the system has no external axis.
Set "0" for data P[7] to P[12] if the system has robots with 7 axes.

Form
The form data are represented by bit data in decimals.



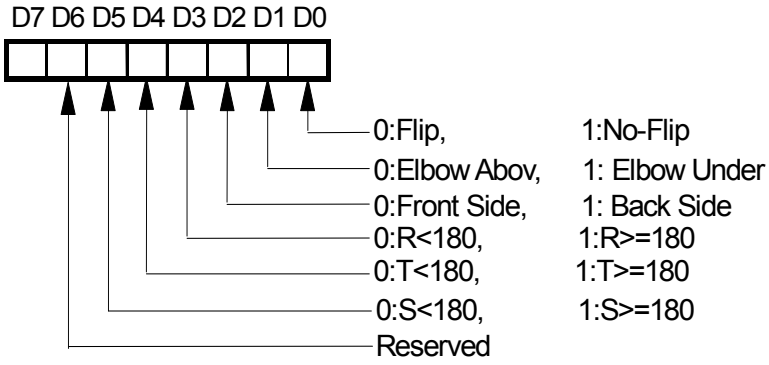
* With the ERC or ERC2, the data from D3 to D7 are disregarded.
* With the MRC or MRC2, the data D5 and D7 are disregarded.

CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC(Serial Port, Ethernet), ERC(Serial Port)
REFERENCE	"BscMovEx2" "BscMovjEx" "BscMovlEx"

■ BscMDSP

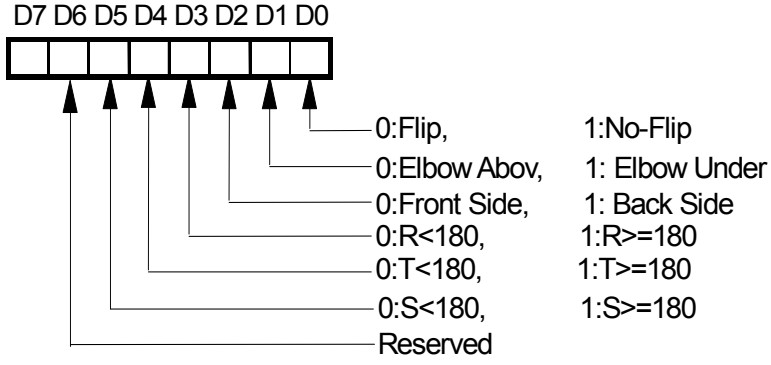
FUNCTION	Sends message data.
FORMAT	_declspec(dllexport) short APIENTRY BscMDSP(short nCid,char *ptr);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *ptr Message storage pointer
	OUT (Return) None
	Return Value 0 : Normal completion Others : Error codes
REMARKS	**Number of message characters** The character string for a message is restricted as follows. With YRC1000/YRC1000micro/DX200/DX100/FS100/NX100/XRC/MRC; Character string with 30 characters maximum. With ERC; Character string with 28 characters maximum.
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)

■ BscMov

FUNCTION	Moves robot with specified motion from the current position to a target position in a specified frame system.
FORMAT	_declspec(dllexport) short APIENTRY BscMov(short nCid,char *movtype,char *vtype,double spd,char *framename,short rconf,short toolno,double *p);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *movtype Motion type; MOVJ : Joint; MOVL : Linear; IMOV : Linear (incrementa value) *vtype Move speed selection; V : Control point; VR : Position angular VJ : Joint speed spd Move speed (0.1 to ***.* mm/s, 0.1 to ***.*° /s, 0.001 to 100.00%) *framename Coordinate name; BASE : Base coordinate; ROBOT : Robot coordinate; UF1 : User coordinate1... TOOL : Tool coordinate (Only for NX100/XRC/MRC with motion type "IMOV.") rconf Form toolno Tool number *p Head pointer to the target position storage area
	OUT (Return) None
	Return Value 0 : Normal completion Others : Error codes
REMARKS	**Form** The form data are represented by bit data in decimals.  * With the ERC or ERC2, the data from D3 to D7 are disregarded. * With the MRC or MRC2, the data D5 and D7 are disregarded.

<table>
<tr><td>REMARKS</td><td>Target Position
The target position data are represented as follows.
<table><tr><th></th><th>Target position in the specified coordinate system</th></tr><tr><td>P[0]</td><td>X-axis coordinate (unit: mm)</td></tr><tr><td>P[1]</td><td>Y-axis coordinate (unit: mm)</td></tr><tr><td>P[2]</td><td>Z-axis coordinate (unit: mm)</td></tr><tr><td>P[3]</td><td>Wrist angle Rx (unit: °)</td></tr><tr><td>P[4]</td><td>Wrist angle Ry (unit: °)</td></tr><tr><td>P[5]</td><td>Wrist angle Rz (unit: °)</td></tr><tr><td>P[6]</td><td>7th axis pulse number (mm for traveling axis)</td></tr><tr><td>P[7]</td><td>8th axis pulse number (mm for traveling axis)</td></tr><tr><td>P[8]</td><td>9th axis pulse number (mm for traveling axis)</td></tr><tr><td>P[9]</td><td>10th axis pulse number</td></tr><tr><td>P[10]</td><td>11th axis pulse number</td></tr><tr><td>P[11]</td><td>12th axis pulse number</td></tr></table>*Set "0" for data P[7] to P[11] if the system has no external axis.</td></tr>
<tr><td>CONTROLLER</td><td>NX100, XRC, MRC (Serial Port, Ethernet), ERC (Serial Port)
* Refer to the BscMovEx2 for the YRC1000/YRC1000micro/DX200/ DX100/FS100.</td></tr>
<tr><td>REFERENCE</td><td>"BscMovEx" "BscMovl" "BscImov" "BscMovEx2"</td></tr>
</table>

■ BscMovEx

FUNCTION	Moves robot with specified motion from the current position to a target position in a specified frame system.(robots with 7 axes or more)
FORMAT	_declspec(dllexport) short APIENTRY BscMovEx(short nCid,char *movtype,char *vtype,double spd,char *framename,long rconf,short toolno,double *p,short axisNum);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *movtype Motion type; MOVJ : Joint; MOVL : Linear; IMOV : Linear (incrementa value) *vtype Move speed selection; V : Control point; VR : Position angular VJ : Joint speed spd Move speed (0.1 to ***.* mm/s, 0.1 to ***.*° /s, 0.01 to 100.00 %) *framename Coordinate name; BASE : Base coordinate; ROBOT : Robot coordinate; UF1 : User coordinate1... TOOL : Tool coordinate (Only motion type "IMOV.") rconf Form toolno Tool number *p Head pointer to the target position storage area axisNum Number of axis
	OUT (Return) None
	Return Value 0 : Normal completion Others : Error codes
REMARKS	**Restrictions** This function is to keep compatibility. Please use BscMovEx2.
	Form The form data are represented by bit data in decimals. 

REMARKS	**Target Position** The target position data are represented as follows. *Set "0" for data P[6] to P[11] if the system has no external axis. Set "0" for data P[7] to P[12] if the system has robots with 7 axes.
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100 (Serial Port, Ethernet)
REFERENCE	"BscMovjEx" "BscMovlEx" "BscImovEx"

	Target position in the specified coordinate system	Robots with 7 axes
P[0]	X-axis coordinate (unit: mm)	X-axis coordinate (unit: mm)
P[1]	Y-axis coordinate (unit: mm)	Y-axis coordinate (unit: mm)
P[2]	Z-axis coordinate (unit: mm)	Z-axis coordinate (unit: mm)
P[3]	Wrist angle Rx (unit: °)	Wrist angle Rx (unit: °)
P[4]	Wrist angle Ry (unit: °)	Wrist angle Ry (unit: °)
P[5]	Wrist angle Rz (unit: °)	Wrist angle Rz (unit: °)
P[6]	7th axis pulse number (mm for traveling axis)	Re (unit: °)
P[7]	8th axis pulse number (mm for traveling axis)	8th axis pulse number (mm for traveling axis)
P[8]	9th axis pulse number (mm for traveling axis)	9th axis pulse number (mm for traveling axis)
P[9]	10th axis pulse number	10th axis pulse number (mm for traveling axis)
P[10]	11th axis pulse number	11th axis pulse number
P[11]	12th axis pulse number	12th axis pulse number
P[12]	-	13th axis pulse number

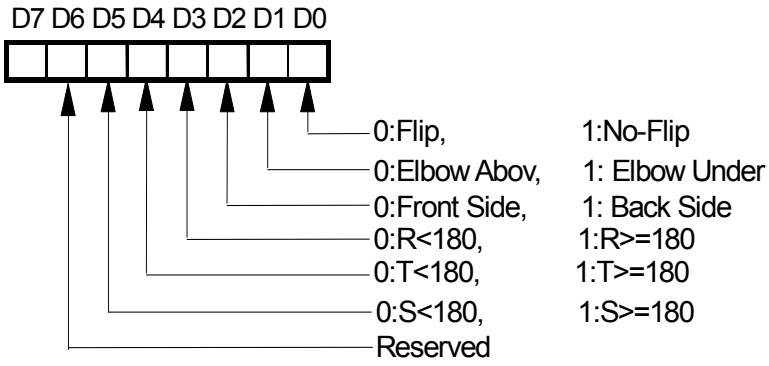
■ BscMovEx2

FUNCTION	Moves robot with specified motion from the current position to a target position in a specified frame system.(robots with 7 axes or more)
FORMAT	_declspec(dllexport) short APIENTRY BscMovEx(short nCid,short ctype,char *movtype,char *vtype,double spd,char *framename,long rconf,short toolno,double *p,short axisNum);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number ctype Controller selection; 0 : ERC, 1 : MRC, 2 : XRC, 3 : NX100 , 4 : YRC1000, YRC1000micro, DX200, DX100, 5 : FS100 *movtype Motion type; MOVJ : Joint; MOVL : Linear; IMOV : Linear (incrementa value) *vtype Move speed selection; V : Control point; VR : Position angular VJ : Joint speed spd Move speed (0.1 to ***.* mm/s, 0.1 to ***.*° /s, 0.01 to 100.00 %) *framename Coordinate name; BASE : Base coordinate; ROBOT : Robot coordinate; UF1 : User coordinate1... TOOL : Tool coordinate (Only for YRC1000/YRC1000micro/ DX200/DX100/FS100/NX100/XRC/ MRC with motion type "IMOV.") rconf Form toolno Tool number *p Head pointer to the target position storage area axisNum Number of robot axes
	OUT (Return) None
	Return Value 0 : Normal completion Others : Error codes
REMARKS	**Restrictions** YRC1000 and YRC1000micro and DX200 and DX100 and FS100 only can be used robot with 7 axes or more.

REMARKS

Form

The form data are represented by bit data in decimals.



* With the ERC or ERC2, the data from D3 to D7 are disregarded.
* With the MRC or MRC2, the data D5 and D7 are disregarded.

Target Position

The target position data are represented as follows.

	Target position in the specified coordinate system	Robots with 7 axes
P[0]	X-axis coordinate (unit: mm)	X-axis coordinate (unit: mm)
P[1]	Y-axis coordinate (unit: mm)	Y-axis coordinate (unit: mm)
P[2]	Z-axis coordinate (unit: mm)	Z-axis coordinate (unit: mm)
P[3]	Wrist angle Rx (unit: °)	Wrist angle Rx (unit: °)
P[4]	Wrist angle Ry (unit: °)	Wrist angle Ry (unit: °)
P[5]	Wrist angle Rz (unit: °)	Wrist angle Rz (unit: °)
P[6]	7th axis pulse number (mm for traveling axis)	Re (unit: °)
P[7]	8th axis pulse number (mm for traveling axis)	8th axis pulse number (mm for traveling axis)
P[8]	9th axis pulse number (mm for traveling axis)	9th axis pulse number (mm for traveling axis)
P[9]	10th axis pulse number	10th axis pulse number (mm for traveling axis)
P[10]	11th axis pulse number	11th axis pulse number
P[11]	12th axis pulse number	12th axis pulse number
P[12]	-	13th axis pulse number

*Set "0" for data P[6] to P[11] if the system has no external axis.
Set "0" for data P[7] to P[12] if the system has robots with 7 axes.

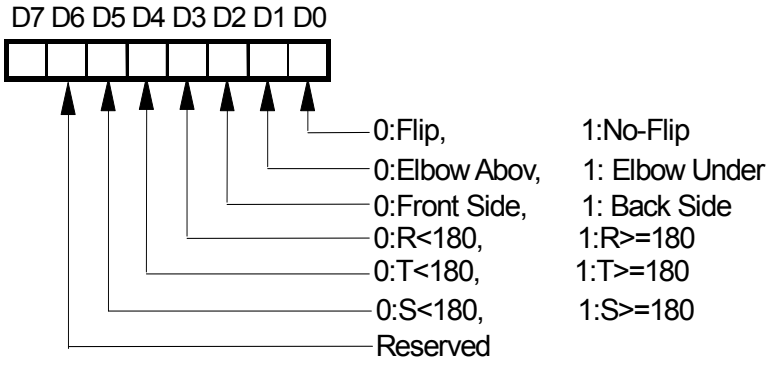
CONTROLLER

YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC(Serial Port, Ethernet), ERC(Serial Port)

REFERENCE

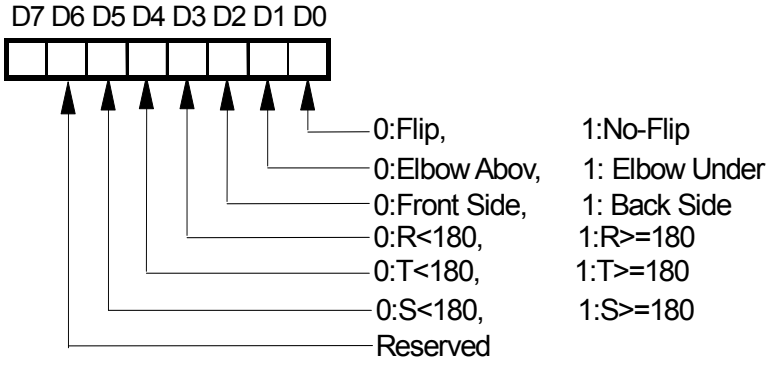
"BscMovjEx" "BscMovlEx" "BscImovEx2"

■ BscMovj

FUNCTION	Moves robot with joint motion to a target position in a specified frame system.
FORMAT	_declspec(dllexport) short APIENTRY BscMovj(short nCid,double spd,char *framename,short rconf,short toolno,double *p);
ARGUMENTS	IN (Transfer) nCid Communication handler ID number spd Move speed (0.01 to 100.0%) *framename Coordinate name; BASE : Base coordinate; ROBOT : Robot coordinate; UF1 : User coordinate1... rconf Form toolno Tool number *p Head pointer to the target position storage area
	OUT (Return) None
	Return Value 0 : Normal completion Others : Error codes
REMARKS	From The form data are represented by bit data in decimals.  * With the ERC or ERC2, the data from D3 to D7 are disregarded. * With the MRC or MRC2, the data D5 and D7 are disregarded.

<table>
<tr><td>REMARKS</td><td>Target Position
The target positions are represented as follows.
<table><tr><th></th><th>Target position in the specified coordinate system</th></tr><tr><td>P[0]</td><td>X-axis coordinate (unit: mm)</td></tr><tr><td>P[1]</td><td>Y-axis coordinate (unit: mm)</td></tr><tr><td>P[2]</td><td>Z-axis coordinate (unit: mm)</td></tr><tr><td>P[3]</td><td>Wrist angle Rx (unit: °)</td></tr><tr><td>P[4]</td><td>Wrist angle Ry (unit: °)</td></tr><tr><td>P[5]</td><td>Wrist angle Rz (unit: °)</td></tr><tr><td>P[6]</td><td>7th axis pulse number (mm for traveling axis)</td></tr><tr><td>P[7]</td><td>8th axis pulse number (mm for traveling axis)</td></tr><tr><td>P[8]</td><td>9th axis pulse number (mm for traveling axis)</td></tr><tr><td>P[9]</td><td>10th axis pulse number</td></tr><tr><td>P[10]</td><td>11th axis pulse number</td></tr><tr><td>P[11]</td><td>12th axis pulse number</td></tr></table>*Set "0" for data P[7] to P[11] if the system has no external axis.</td></tr>
<tr><td>CONTROLLER</td><td>YRC1000, YRC1000micro, DX200, DX100, FS100,NX100, XRC, MRC (Serial Port, Ethernet)
ERC (Serial Port)
* Refer to the BscMovjEx for the YRC1000/YRC1000micro/DX200/ DX100/FS100 with 7 axes or more.</td></tr>
<tr><td>REFERENCE</td><td>"BscMov" "BscMovl" "BscImov"</td></tr>
</table>

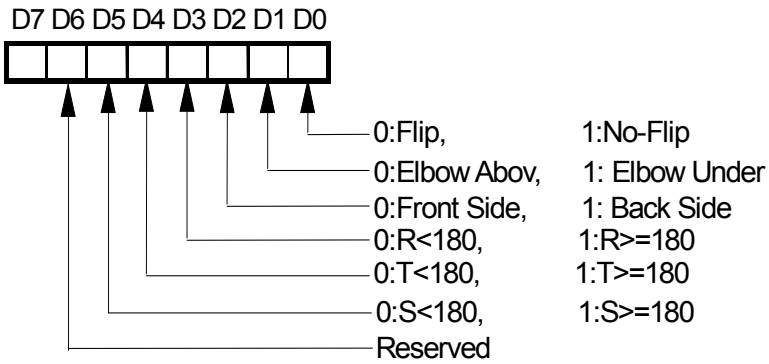
■ BscMovjEx

FUNCTION	Moves robot with joint motion to a target position in a specified frame system. (robots with 7 axes or more)
FORMAT	_declspec(dllexport) short APIENTRY BscMovjEx(short nCid,double spd,char *framename,long rconf,short toolno,double *p,short axisNum);
ARGUMENTS	IN (Transfer) nCid Communication handler ID number spd Move speed (0.01 to 100.0%) *framename Coordinate name; BASE : Base coordinate; ROBOT : Robot coordinate; UF1 : User coordinate1... rconf Form toolno Tool number *p Head pointer to the target position storage area axisNum Number of axis
	OUT (Return) None
	Return Value 0 : Normal completion Others : Error codes
REMARKS	From The form data are represented by bit data in decimals. 

REMARKS	**Target Position** The target positions are represented as follows. *Set "0" for data P[6] to P[11] if the system has no external axis. Set "0" for data P[7] to P[12] if the system has robots with 7 axes.
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100 (Serial Port, Ethernet)
REFERENCE	"BscMovEx2" "BscMovlEx" "BscImovEx2"

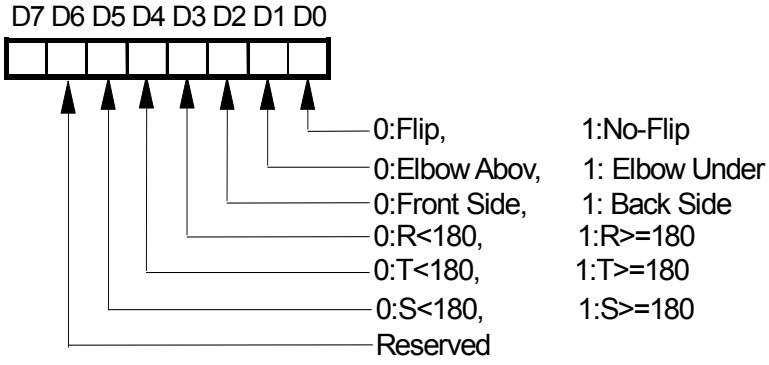
	Target position in the specified coordinate system	Robots with 7 axes
P[0]	X-axis coordinate (unit: mm)	X-axis coordinate (unit: mm)
P[1]	Y-axis coordinate (unit: mm)	Y-axis coordinate (unit: mm)
P[2]	Z-axis coordinate (unit: mm)	Z-axis coordinate (unit: mm)
P[3]	Wrist angle Rx (unit: °)	Wrist angle Rx (unit: °)
P[4]	Wrist angle Ry (unit: °)	Wrist angle Ry (unit: °)
P[5]	Wrist angle Rz (unit: °)	Wrist angle Rz (unit: °)
P[6]	7th axis pulse number (mm for traveling axis)	Re (unit: °)
P[7]	8th axis pulse number (mm for traveling axis)	8th axis pulse number (mm for traveling axis)
P[8]	9th axis pulse number (mm for traveling axis)	9th axis pulse number (mm for traveling axis)
P[9]	10th axis pulse number	10th axis pulse number (mm for traveling axis)
P[10]	11th axis pulse number	11th axis pulse number
P[11]	12th axis pulse number	12th axis pulse number
P[12]	-	13th axis pulse number

■ BscMovl

FUNCTION	Moves robot with linear motion to a target position in a specified frame system.
FORMAT	_declspec(dllexport) short APIENTRY BscMovl(short nCid,char *vtype,double spd,char *framename,short rconf,short toolno,double *p);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *vtype Move speed selection; V : Control point; VR : Position angular spd Move speed (0.1 to ****.* mm/s, 0.1 to ***.*° /s) *framename Coordinate name; BASE : Base coordinate; ROBOT : Robot coordinate; UF1 : User coordinate1... rconf Form toolno Tool number *p Head pointer to the target position storage area
	OUT (Return) None
	Return Value 0 : Normal completion Others : Error codes
REMARKS	**Form** The form data are represented by bit data in decimals.  * With the ERC or ERC2, the data from D3 to D7 are disregarded. * With the MRC or MRC2, the data D5 and D7 are disregarded.

<table>
<tr><td>REMARKS</td><td>Target Position
The target positions are represented as follows.
<table><tr><th></th><th>Target position in the specified coordinate system</th></tr><tr><td>P[0]</td><td>X-axis coordinate (unit: mm)</td></tr><tr><td>P[1]</td><td>Y-axis coordinate (unit: mm)</td></tr><tr><td>P[2]</td><td>Z-axis coordinate (unit: mm)</td></tr><tr><td>P[3]</td><td>Wrist angle Rx (unit: °)</td></tr><tr><td>P[4]</td><td>Wrist angle Ry (unit: °)</td></tr><tr><td>P[5]</td><td>Wrist angle Rz (unit: °)</td></tr><tr><td>P[6]</td><td>7th axis pulse number (mm for traveling axis)</td></tr><tr><td>P[7]</td><td>8th axis pulse number (mm for traveling axis)</td></tr><tr><td>P[8]</td><td>9th axis pulse number (mm for traveling axis)</td></tr><tr><td>P[9]</td><td>10th axis pulse number</td></tr><tr><td>P[10]</td><td>11th axis pulse number</td></tr><tr><td>P[11]</td><td>12th axis pulse number</td></tr></table>*Set "0" for data P[7] to P[11] if the system has no external axis.</td></tr>
<tr><td>CONTROLLER</td><td>YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet)
ERC (Serial Port)
* Refer to the BscMovlEx for the YRC1000/YRC1000micro/DX200/ DX100/FS100 with 7 axes or more.</td></tr>
<tr><td>REFERENCE</td><td>"BscMov" "BscMovEx" "BscImov"</td></tr>
</table>

■ BscMovlEx

FUNCTION	Moves robot with linear motion to a target position in a specified frame system. (robots with 7 axes or more)
FORMAT	_declspec(dllexport) short APIENTRY BscMovlEx(short nCid,char *vtype,double spd,char *framename,long rconf,short toolno,double *p,short axisNum);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *vtype Move speed selection; V : Control point; VR : Position angular spd Move speed (0.1 to ****.* mm/s, 0.1 to ***.*° /s) *framename Coordinate name; BASE : Base coordinate; ROBOT : Robot coordinate; UF1 : User coordinate1... rconf Form toolno Tool number *p Head pointer to the target position storage area axisNum Number of axis
	OUT (Return) None
	Return Value 0 : Normal completion Others : Error codes
REMARKS	**Form** The form data are represented by bit data in decimals. 

REMARKS	**Target Position** The target positions are represented as follows. <table><tr><th></th><th>Target position in the specified coordinate system</th><th>Robots with 7 axes</th></tr><tr><td>P[0]</td><td>X-axis coordinate (unit: mm)</td><td>X-axis coordinate (unit: mm</td></tr><tr><td>P[1]</td><td>Y-axis coordinate (unit: mm)</td><td>Y-axis coordinate (unit: mm</td></tr><tr><td>P[2]</td><td>Z-axis coordinate (unit: mm)</td><td>Z-axis coordinate (unit: mm</td></tr><tr><td>P[3]</td><td>Wrist angle Rx (unit: °)</td><td>Wrist angle Rx (unit: °)</td></tr><tr><td>P[4]</td><td>Wrist angle Ry (unit: °)</td><td>Wrist angle Ry (unit: °)</td></tr><tr><td>P[5]</td><td>Wrist angle Rz (unit: °)</td><td>Wrist angle Rz (unit: °)</td></tr><tr><td>P[6]</td><td>7th axis pulse number (mm for traveling axis)</td><td>Re (unit: °)</td></tr><tr><td>P[7]</td><td>8th axis pulse number (mm for traveling axis)</td><td>8th axis pulse number (mm for traveling axis)</td></tr><tr><td>P[8]</td><td>9th axis pulse number (mm for traveling axis)</td><td>9th axis pulse number (mm for traveling axis)</td></tr><tr><td>P[9]</td><td>10th axis pulse number</td><td>10th axis pulse number (mm for traveling axis)</td></tr><tr><td>P[10]</td><td>11th axis pulse number</td><td>11th axis pulse number</td></tr><tr><td>P[11]</td><td>12th axis pulse number</td><td>12th axis pulse number</td></tr><tr><td>P[12]</td><td>-</td><td>13th axis pulse number</td></tr></table>*Set "0" for data P[6] to P[11] if the system has no external axis. Set "0" for data P[7] to P[12] if the system has robots with 7 axes.
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100 (Serial Port, Ethernet)
REFERENCE	"BscMovEx2" "BscMovjEx" "BscImovEx2"

■ BscOPLock

FUNCTION	Interlocks the robot.
FORMAT	_declspec(dllexport) short APIENTRY BscOPLock(short nCid);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number
	OUT (Return) None
	Return Value 0 : Normal completion Others : Error codes
REMARKS	**Interlock Status** Once interlock is set, all but the following are interlocked.
	With YRC1000/YRC1000micro/DX200/DX100/FS100/NX100 * Emergency stop, hold from the programming pendant. * Input signals except I/O mode change, external start, external servo ON, cycle change, I/O prohibited, P.P/PANEL prohibited, and master call Interlock cannot be accomplished when the programming pendant is in the editing mode, or when file access is operating by other functions.
	With XRC * Hold from the programming pendant. * Hold, emergency stop from the playback box. * Input signal other than I/O 404x, 405x and 409x. (including external hold, external servo OFF) Interlock cannot be accomplished when the programming pendant is in the editing mode, or when file access is operating by other functions.
	With MRC * Hold from the programming pendant. * Hold, emergency stop from the playback box. * Input signal other than I/O 404x and 405x. (including external hold, external servo OFF) Interlock cannot be accomplished when the programming pendant is in the editing mode, or when file access is operating by other functions.
	With ERC * Start and hold buttons of panel operation. * Emergency stop button of panel operation. * Servo power ON button of panel operation.

CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	"BscOPUnLock"

■ BscOPUnLock

FUNCTION	Releases the robot interlocked status.
FORMAT	_declspec(dllexport) short APIENTRY BscOPUnLock(short nCid);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number
	OUT (Return) None
	Return Value 0 : Normal completion Others : Error codes
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	"BscOPLock"

■ BscPMov

FUNCTION	Moves robot to a specified pulse position.
FORMAT	_declspec(dllexport) short APIENTRY BscPMov(short nCid,char *movtype,char *vtype,double spd,short toolno,double *p);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *movvtype Motion type; MOVJ : Joint; MOVL : Linear *vtype Move speed selection; V : Control point; VR : Position angular VJ : Joint speed spd Move speed (0.1 to ****.* mm/s, 0.1 to ***.*° /s, 0.01 to 100.00%) toolno Tool number *p Head pointer to the target position storage area
	OUT (Return) None
	Return Value 0 : Normal completion Others : Error codes
REMARKS	**Target Position** The target position data are represented as follows.

	Target position in units of pulses
P[0]	S-axis pulse number
P[1]	L-axis pulse number
P[2]	U-axis pulse number
P[3]	R-axis pulse number
P[4]	B-axis pulse number
P[5]	T-axis pulse number
P[6]	7th axis pulse number
P[7]	8th axis pulse number
P[8]	9th axis pulse number
P[9]	10th axis pulse number
P[10]	11th axis pulse number
P[11]	12th axis pulse number

*Set "0" for data P[7] to P[11] if the system has no external axis.

CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port) * Refer to the BscPMovEx for the YRC1000/YRC1000micro/DX200/ DX100/FS100 with 7 axes or more.
REFERENCE	"BscPMovj" "BscPMovl"

■ BscPMovEx

FUNCTION	Moves robot to a specified pulse position. (robots with 7 axes or more)
FORMAT	_declspec(dllexport) short APIENTRY BscPMovEx(short nCid,char *movtype,char *vtype,double spd,short toolno,double *p,short axisNum);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *movvtype Motion type; MOVJ : Joint, MOVL : Linear *vtype Move speed selection; V : TCP speed VR : Play speed of the posture VJ : Joint speed spd Move speed (0.1 to ****.* mm/s, 0.1 to ***.*° /s, 0.01 to 100.00%) toolno Tool number *p Head pointer to the target position storage area axisNum Number of axis
	OUT (Return) None
	Return Value 0 : Normal completion Others : Error codes
REMARKS	**Target Position** The target position data are represented as follows.

	Target position in units of pulses	Robots with 7 axes
P[0]	S-axis pulse number	S-axis pulse number
P[1]	L-axis pulse number	L-axis pulse number
P[2]	U-axis pulse number	U-axis pulse number
P[3]	R-axis pulse number	R-axis pulse number
P[4]	B-axis pulse number	B-axis pulse number
P[5]	T-axis pulse number	T-axis pulse number
P[6]	7th axis pulse number	E-axis pulse number
P[7]	8th axis pulse number	8th axis pulse number
P[8]	9th axis pulse number	9th axis pulse number
P[9]	10th axis pulse number	10th axis pulse number
P[10]	11th axis pulse number	11th axis pulse number
P[11]	12th axis pulse number	12th axis pulse number
P[12]	-	13th axis pulse number

*Set "0" for data P[6] to P[11] if the system has no external axis.
Set "0" for data P[7] to P[12] if the system has robots with 7 axes.

CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100 (Serial Port, Ethernet)
REFERENCE	"BscPMovjEx" "BscPMovlEx"

■ BscPMovj

FUNCTION	Moves robot to a specified pulse position with joint motion.
FORMAT	_declspec(dllexport) short APIENTRY BscPMovj(short nCid,double spd,short toolno,double *p);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number spd Move speed (0.01 to 100.00%) toolno Tool number *p Head pointer to the target position storage area
	OUT (Return) None
	Return Value 0 : Normal completion Others : Error codes
REMARKS	**Target Position** The target position data are represented as follows.

	Target position in units of pulses
P[0]	S-axis pulse number
P[1]	L-axis pulse number
P[2]	U-axis pulse number
P[3]	R-axis pulse number
P[4]	B-axis pulse number
P[5]	T-axis pulse number
P[6]	7th axis pulse number
P[7]	8th axis pulse number
P[8]	9th axis pulse number
P[9]	10th axis pulse number
P[10]	11th axis pulse number
P[11]	12th axis pulse number

*Set "0" for data P[7] to P[11] if the system has no external axis.

CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port) * Refer to the BscPMovjEx for the YRC1000/YRC1000micro/ DX200/DX100/FS100 with 7 axes or more.
REFERENCE	"BscPMovl" "BscPMov"

■ BscPMovjEx

FUNCTION	Moves robot to a specified pulse position with joint motion. (robots with 7 axes or more)
FORMAT	_declspec(dllexport) short APIENTRY BscPMovjEx(short nCid,double spd,short toolno,double *p,short axisNum);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number spd Move speed (0.01 to 100.00%) toolno Tool number *p Head pointer to the target position storage area axisNum Number of axis
	OUT (Return) None
	Return Value 0 : Normal completion Others : Error codes
REMARKS	**Target Position** The target position data are represented as follows.

	Target position in units of pulses	Robots with 7 axes
P[0]	S-axis pulse number	S-axis pulse number
P[1]	L-axis pulse number	L-axis pulse number
P[2]	U-axis pulse number	U-axis pulse number
P[3]	R-axis pulse number	R-axis pulse number
P[4]	B-axis pulse number	B-axis pulse number
P[5]	T-axis pulse number	T-axis pulse number
P[6]	7th axis pulse number	E-axis pulse number
P[7]	8th axis pulse number	8th axis pulse number
P[8]	9th axis pulse number	9th axis pulse number
P[9]	10th axis pulse number	10th axis pulse number
P[10]	11th axis pulse number	11th axis pulse number
P[11]	12th axis pulse number	12th axis pulse number
P[12]	-	13th axis pulse number

*Set "0" for data P[6] to P[11] if the system has no external axis.
Set "0" for data P[7] to P[12] if the system has robots with 7 axes.

CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100 (Serial Port, Ethernet)
REFERENCE	"BscPMovlEx" "BscPMovEx"

■ BscPMovl

FUNCTION	Moves robot to a specified pulse position with linear motion.
FORMAT	_declspec(dllexport) short APIENTRY BscPMovl(short nCid,char *vtype,double spd,short toolno,double *p);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *vtype Move speed selection; V : Control point; VR : Position angular spd Move speed (0.1 to ****.*mm/s, 0.1 to ***.*° /s) toolno Tool number *p Head pointer to the target position storage area
	OUT (Return) None
	Return Value 0 : Normal completion Others : Error codes
REMARKS	**Target Position** The target position data are represented as follows.

	Target position in units of pulses
P[0]	S-axis pulse number
P[1]	L-axis pulse number
P[2]	U-axis pulse number
P[3]	R-axis pulse number
P[4]	B-axis pulse number
P[5]	T-axis pulse number
P[6]	7th axis pulse number
P[7]	8th axis pulse number
P[8]	9th axis pulse number
P[9]	10th axis pulse number
P[10]	11th axis pulse number
P[11]	12th axis pulse number

*Set "0" for data P[7] to P[11] if the system has no external axis.

CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port) * Refer to the BscPMovlEx for the YRC1000/YRC1000micro/ DX200/DX100/FS100 with 7 axes or more.
REFERENCE	"BscPMovj" "BscPMov"

■ BscPMovlEx

FUNCTION	Moves robot to a specified pulse position with linear motion.
FORMAT	_declspec(dllexport) short APIENTRY BscPMovlEx(short nCid,char *vtype,double spd,short toolno,double *p,short axisNum);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *vtype Move speed selection; V : Control point; VR : Position angular spd Move speed (0.1 to ****.*mm/s, 0.1 to ***.*° /s) toolno Tool number *p Head pointer to the target position storage area axisNum Number of axis
	OUT (Return) None
	Return Value 0 : Normal completion Others : Error codes
REMARKS	**Target Position** The target position data are represented as follows.

	Target position in units of pulses	Robots with 7 axes
P[0]	S-axis pulse number	S-axis pulse number
P[1]	L-axis pulse number	L-axis pulse number
P[2]	U-axis pulse number	U-axis pulse number
P[3]	R-axis pulse number	R-axis pulse number
P[4]	B-axis pulse number	B-axis pulse number
P[5]	T-axis pulse number	T-axis pulse number
P[6]	7th axis pulse number	E-axis pulse number
P[7]	8th axis pulse number	8th axis pulse number
P[8]	9th axis pulse number	9th axis pulse number
P[9]	10th axis pulse number	10th axis pulse number
P[10]	11th axis pulse number	11th axis pulse number
P[11]	12th axis pulse number	12th axis pulse number
P[12]	-	13th axis pulse number

*Set "0" for data P[6] to P[11] if the system has no external axis.
Set "0" for data P[7] to P[12] if the system has robots with 7 axes.

CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100 (Serial Port, Ethernet)
REFERENCE	"BscPMovjEx" "BscPMovEx"

■ BscPutUFrame

FUNCTION	Sets a specified user frame data.
FORMAT	_declspec(dllexport) short APIENTRY BscPutUFrame(short nCid,char *ufname,double *p);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *ufname Storage pointer of the user coordinate name to be written in. *p Head pointer to the user coordinate data storage area
	OUT (Return) None
	Return Value 0 : Normal completion Others : Error codes
REMARKS	**Restrictions** This function does not support the robot with 7 axes or more.
	User Coordinate Name The following coordinate names correspond to the user coordinate numbers.

User Coordinate Name	Specified Name
User coordinate 1	UF1
User coordinate 2	UF2
User coordinate 3	UF3
:	:
User coordinate 21	UF21
User coordinate 22	UF22
User coordinate 23	UF23
User coordinate 24	UF24

* User coordinate numbers 9 to 24 are effective only for NX100/ XRC/MRC.

REMARKS

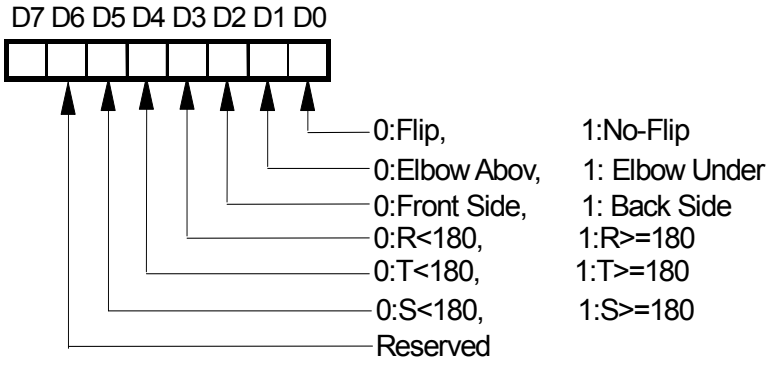
Variable Types

Coordinate values of the user coordinate system specified with the user coordinate number are assigned to the user coordinate data as follows.

Variables	Coordinate System	Meaning
P[0]	ORG	X-axis coordinate (unit: mm, effective down to 3 decimal places
P[1]		Y-axis coordinate (unit: mm, effective down to 3 decimal places)
P[2]		Z-axis coordinate (unit: mm, effective down to 3 decimal places)
P[3]		Wrist angle Rx (unit:°, effective down to 2 decimal places)
P[4]		Wrist angle Ry (unit:°, effective down to 2 decimal places)
P[5]		Wrist angle Rz (unit:°, effective down to 2 decimal places)
P[6]		Form
P[7]	XX	X-axis coordinate (unit: mm, effective down to 3 decimal places
P[8]		Y-axis coordinate (unit: mm, effective down to 3 decimal places)
P[9]		Z-axis coordinate (unit: mm, effective down to 3 decimal places)
P[10]		Wrist angle Rx (unit:°, effective down to 2 decimal places)
P[11]		Wrist angle Ry (unit:°, effective down to 2 decimal places)
P[12]		Wrist angle Rz (unit:°, effective down to 2 decimal places)
P[13]		Form
P[14]	XY	X-axis coordinate (unit: mm, effective down to 3 decimal places
P[15]		Y-axis coordinate (unit: mm, effective down to 3 decimal places)
P[16]		Z-axis coordinate (unit: mm, effective down to 3 decimal places)
P[17]		Wrist angle Rx (unit:°, effective down to 2 decimal places)
P[18]		Wrist angle Ry (unit:°, effective down to 2 decimal places)
P[19]		Wrist angle Rz (unit:°, effective down to 2 decimal places)
P[20]		Form
P[21]	Tool number	
P[22]	7th axis pulse number (mm for traveling axis)	
P[23]	8th axis pulse number (mm for traveling axis)	
P[24]	9th axis pulse number (mm for traveling axis)	
P[25]	10th axis pulse number	
P[26]	11th axis pulse number	
P[27]	12th axis pulse number	

Form

The form data are represented by bit data in decimals.



* With the ERC or ERC2, the data from D3 to D7 are disregarded.
* With the MRC or MRC2, the data D5 and D7 are disregarded.

CONTROLLER: NX100, XRC, MRC (Serial Port, Ethernet), ERC (Serial Port)

REFERENCE: "BscGetUFrame" "BscPutUFrameEx2"

■ BscPutUFrameEx2

FUNCTION	Sets a specified user frame data.
FORMAT	_declspec(dllexport) short APIENTRY BscPutUFrame(short nCid,short ctype,char *ufname,double *p);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number ctype Controller selection; 0 : ERC, 1 : MRC, 2 : XRC, 3 : NX100 , 4 : YRC1000, YRC1000micro,DX200, DX100, 5 : FS100 *ufname Storage pointer of the user coordinate name to be written in. *p Head pointer to the user coordinate data storage area
	OUT (Return) None
	Return Value 0 : Normal completion Others : Error codes
REMARKS	**Restrictions** This function does not support the robot with 7 axes or more.
	User Coordinate Name The following coordinate names correspond to the user coordinate numbers.

User Coordinate Name	Specified Name
User coordinate 1	UF1
User coordinate 2	UF2
User coordinate 3	UF3
:	:
User coordinate 61	UF61
User coordinate 62	UF62
User coordinate 63	UF63
User coordinate 64	UF64

* User coordinate numbers 9 to 64 are effective only for YRC1000/ YRC1000micro/DX200/DX100.
* User coordinate numbers 9 to 16 are effective only for FS100.
* User coordinate numbers 9 to 24 are effective only for NX100/ XRC/MRC.

REMARKS

Variable Types

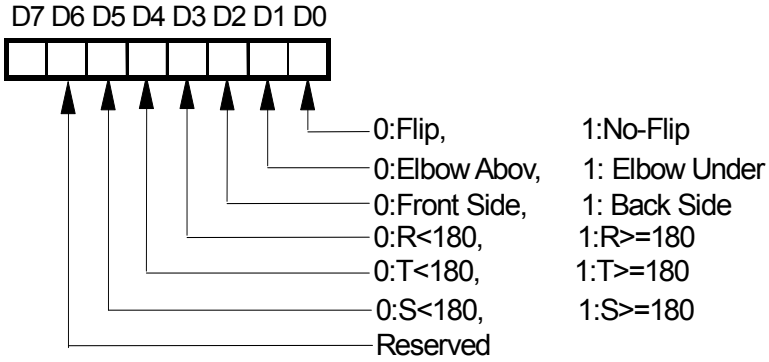
Coordinate values of the user coordinate system specified with the user coordinate number are assigned to the user coordinate data as follows.

Variables	Coordinate System	Meaning
P[0]	ORG	X-axis coordinate (unit: mm, effective down to 3 decimal places
P[1]		Y-axis coordinate (unit: mm, effective down to 3 decimal places
P[2]		Z-axis coordinate (unit: mm, effective down to 3 decimal places
P[3]		Wrist angle Rx (unit:°, effective down to 2 decimal places) *1
P[4]		Wrist angle Ry (unit:°, effective down to 2 decimal places) *1
P[5]		Wrist angle Rz (unit:°, effective down to 2 decimal places) *1
P[6]		Form
P[7]	XX	X-axis coordinate (unit: mm, effective down to 3 decimal places
P[8]		Y-axis coordinate (unit: mm, effective down to 3 decimal places
P[9]		Z-axis coordinate (unit: mm, effective down to 3 decimal places
P[10]		Wrist angle Rx (unit:°, effective down to 2 decimal places) *1
P[11]		Wrist angle Ry (unit:°, effective down to 2 decimal places) *1
P[12]		Wrist angle Rz (unit:°, effective down to 2 decimal places) *1
P[13]		Form
P[14]	XY	X-axis coordinate (unit: mm, effective down to 3 decimal places
P[15]		Y-axis coordinate (unit: mm, effective down to 3 decimal places
P[16]		Z-axis coordinate (unit: mm, effective down to 3 decimal places
P[17]		Wrist angle Rx (unit:°, effective down to 2 decimal places) *1
P[18]		Wrist angle Ry (unit:°, effective down to 2 decimal places) *1
P[19]		Wrist angle Rz (unit:°, effective down to 2 decimal places) *1
P[20]		Form
P[21]	Tool number	
P[22]	7th axis pulse number (mm for traveling axis)	
P[23]	8th axis pulse number (mm for traveling axis)	
P[24]	9th axis pulse number (mm for traveling axis)	
P[25]	10th axis pulse number	
P[26]	11th axis pulse number	
P[27]	12th axis pulse number	

*1 YRC1000, YRC1000micro, DX200, DX100, FS100 is effective down to 4 decimal places.

Form

The form data are represented by bit data in decimals.



* With the ERC or ERC2, the data from D3 to D7 are disregarded.
* With the MRC or MRC2, the data D5 and D7 are disregarded.

CONTROLLER

YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet)
ERC (Serial Port)

REFERENCE	"BscGetUFrame" "BscPutUFrame"

■ BscPutVarData

FUNCTION	Sets variable data.
FORMAT	_declspec(dllexport) short APIENTRY BscPutVarData(short nCid,short type,short varno,double *p);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number type Variable Types varno Variable number *p Head pointer to the numeric variable storage area
	OUT (Return) None
	Return Value 0 : Normal completion Others : Error codes
REMARKS	**Restrictions** This function is effective only for transmission with the YRC1000/ YRC1000micro/DX200/DX100/FS100/NX100/XRC/MRC.
	Variable Types The variable types are represented as follows. 0 : Byte type 1 : Integer type 2 : Double-precision type 3 : Real type 4 : Robot axis position type 5 : Base axis position type 6 : Station axis position type (pulse type only)

REMARKS

Content of the numeric variable storage area

Depending on the variable type, the numeric variable storage area contains the number of values indicated below.

Variable Type Number	Data Type (Pulse/ XYZ)	Number of values	Content				
			P[0]	P[1]	P[2]	P[3]	P[4]
0	-	1	Byte				
1	-	1	Integer	-	-	-	-
2	-	1	Double	-	-	-	-
3	-	1	Real	-	-	-	-
4	Pulse	8	0	S-axis Pulses	L-axis Pulses	U-axis Pulses	R-axis Pulses
4	XYZ	10	1	Coordinate Type	X-axis (mm)	Y-axis (mm)	Z-axis (mm)
5	Pulse	8	0	Base Axis-1 Pulses	Base Axis-2 Pulses	Base Axis-3 Pulses	Base Axis-4 Pulses
5	XYZ	10	1	Coordinate Type	X-axis (mm)	Y-axis (mm)	Z-axis (mm)
6	Pulse	8	0	Station Axis-1 Pulses	Station Axis-2 Pulses	Station Axis-3 Pulses	Station Axis-4 Pulses

Variable Type Number	Data Type (Pulse/ XYZ)	Number of values	Content				
			P[5]	P[6]	P[7]	P[8]	P[9]
0	-	1					
1	-	1	-	-	-	-	-
2	-	1	-	-	-	-	-
3	-	1	-	-	-	-	-
4	Pulse	8	B-axis Pulses	T-axis Pulses	Tool Number	-	-
4	XYZ	10	Rx Angle(deg)	Ry Angle(deg)	Rz Angle(deg)	Form	Tool Number
5	Pulse	8	Base Axis-5 Pulses	Base Axis-6 Pulses	Tool Number	-	-
5	XYZ	10	Rx Angle(deg)	Ry Angle(deg)	Rz Angle(deg)	Form	Tool Number
6	Pulse	8	Station Axis-5 Pulses	Station Axis-6 Pulses	Tool Number	-	-

The robot axis position and base axis position type variables include the pulse type and XYZ type, according to the first return value.
The station axis position type variable contains the pulse type only.
See the following for details on the coordinate system types and form.

Coordinate Types

YRC1000, YRC1000micro, DX200, DX100

The following coordinate names correspond to the coordinate type data.

Coordinate type	Coordinate name	Coordinate type	Coordinate name
0	Base coordinate	:	:
1	Robot coordinate	:	:
2	User coordinate 1	63	User coordinate 62
3	User coordinate 2	64	User coordinate 63
:	:	65	Tool coordinate
:	:	66	Master tool coordinate

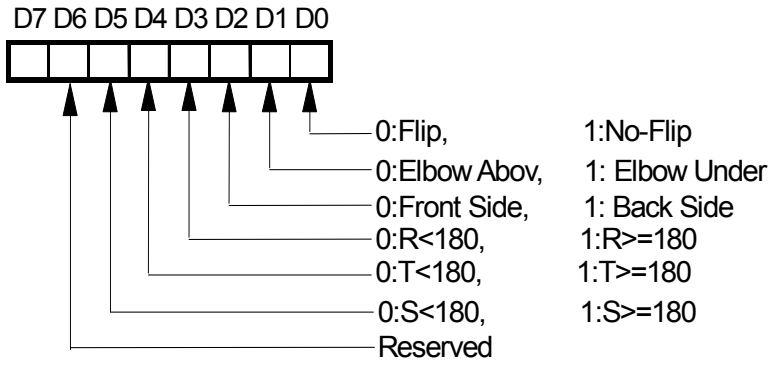
REMARKS

Coordinate Types
FS100
The following coordinate names correspond to the coordinate type data.

Coordinate type	Coordinate name	Coordinate type	Coordinate name
0	Base coordinate	10	User coordinate 9
1	Robot coordinate	11	User coordinate 10
2	User coordinate 1	12	User coordinate 11
3	User coordinate 2	13	User coordinate 12
4	User coordinate 3	14	User coordinate 13
5	User coordinate 4	15	User coordinate 14
6	User coordinate 5	16	User coordinate 15
7	User coordinate 6	17	User coordinate 16
8	User coordinate 7	18	Tool coordinate
9	User coordinate 8	19	Master tool coordinate

Coordinate Types
NX100/XRC/MRC
The following coordinate names correspond to the coordinate type data.

Coordinate type	Coordinate name	Coordinate type	Coordinate name
0	Base coordinate	14	User coordinate 13
1	Robot coordinate	15	User coordinate 14
2	User coordinate 1	16	User coordinate 15
3	User coordinate 2	17	User coordinate 16
4	User coordinate 3	18	User coordinate 17
5	User coordinate 4	19	User coordinate 18
6	User coordinate 5	20	User coordinate 19
7	User coordinate 6	21	User coordinate 20
8	User coordinate 7	22	User coordinate 21
9	User coordinate 8	23	User coordinate 22
10	User coordinate 9	24	User coordinate 23
11	User coordinate 10	25	User coordinate 24
12	User coordinate 11	26	Tool coordinate
13	User coordinate 12	27	Master tool coordinate

REMARKS	**Form** The form data are represented by bit data in decimals.  * With the MRC or MRC2, the data D5 and D7 are disregarded.
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet)
REFERENCE	"BscGetVarData" "BscGetVarData2"

■ BscPutVarData2

FUNCTION	Sets variable data. (robot with 7 axes or more)
FORMAT	_declspec(dllexport) short APIENTRY BscPutVarData2(short nCid,short type,short varno,double *p,short axisNum);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number type Variable Types varno Variable number *p Head pointer to the numeric variable storage area axisNum Number of axis
	OUT (Return) None
	Return Value 0 : Normal completion Others : Error codes
REMARKS	**Restrictions** This function is effective only for transmission with the YRC1000/ YRC1000micro/DX200/DX100/FS100/NX100/XRC/MRC.
	Variable Types The variable types are represented as follows. 0 : Byte type 1 : Integer type 2 : Double-precision type 3 : Real type 4 : Robot axis position type 5 : Base axis position type 6 : Station axis position type (pulse type only)

REMARKS

Content of the numeric variable storage area

Depending on the variable type, the numeric variable storage area contains the number of values indicated below.

Variable Type Number	Data Type (Pulse/ XYZ)	Number of values	Content				
			P[0]	P[1]	P[2]	P[3]	P[4]
0	-	1	Byte				
1	-	1	Integer	-	-	-	-
2	-	1	Double	-	-	-	-
3	-	1	Real	-	-	-	-
4	Pulse	10	0	S-axis Pulses	L-axis Pulses	U-axis Pulses	R-axis Pulses
4	XYZ	10	1	Coordinate Type	X-axis (mm)	Y-axis (mm)	Z-axis (mm)
5	Pulse	8	0	Base Axis-1 Pulses	Base Axis-2 Pulses	Base Axis-3 Pulses	Base Axis-4 Pulses
5	XYZ	10	1	Coordinate Type	X-axis (mm)	Y-axis (mm)	Z-axis (mm)
6	Pulse	8	0	Station Axis-1 Pulses	Station Axis-2 Pulses	Station Axis-3 Pulses	Station Axis-4 Pulses

Variable Type Number	Data Type (Pulse/ XYZ)	Number of values	Content				
			P[5]	P[6]	P[7]	P[8]	P[9]
0	-	1					
1	-	1	-	-	-	-	-
2	-	1	-	-	-	-	-
3	-	1	-	-	-	-	-
4	Pulse	10	B-axis Pulses	T-axis Pulses	7thaxis Pulses	8thaxis Pulses	Tool Number
4	XYZ	10	Rx Angle(deg)	Ry Angle(deg)	Rz Angle(deg)	Form	Tool Number
5	Pulse	8	Base Axis-5 Pulses	Base Axis-6 Pulses	Tool Number	-	-
5	XYZ	10	Rx Angle(deg)	Ry Angle(deg)	Rz Angle(deg)	Form	Tool Number
6	Pulse	8	Station Axis-5 Pulses	Station Axis-6 Pulses	Tool Number	-	-

The robot axis position and base axis position type variables include the pulse type and XYZ type, according to the first return value.
The station axis position type variable contains the pulse type only.
See the following for details on the coordinate system types and form.

Coordinate Types

YRC1000/YRC1000micro/DX200/DX100

The following coordinate names correspond to the coordinate type data.

Coordinate type	Coordinate name	Coordinate type	Coordinate name
0	Base coordinate	:	:
1	Robot coordinate	:	:
2	User coordinate 1	63	User coordinate 62
3	User coordinate 2	64	User coordinate 63
:	:	65	Tool coordinate
:	:	66	Master tool coordinate

REMARKS

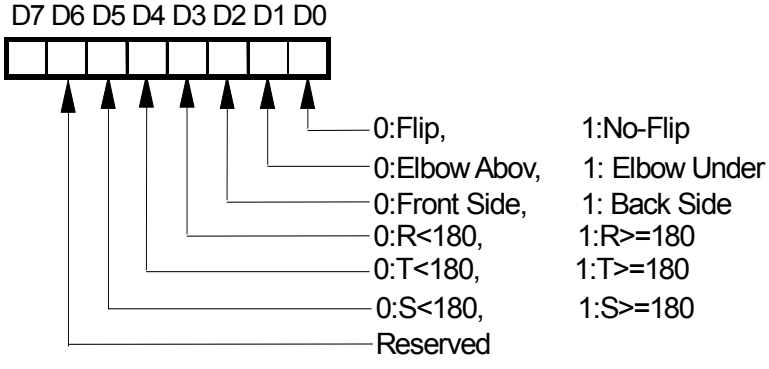
Coordinate Types
FS100
The following coordinate names correspond to the coordinate type data.

Coordinate type	Coordinate name	Coordinate type	Coordinate name
0	Base coordinate	10	User coordinate 9
1	Robot coordinate	11	User coordinate 10
2	User coordinate 1	12	User coordinate 11
3	User coordinate 2	13	User coordinate 12
4	User coordinate 3	14	User coordinate 13
5	User coordinate 4	15	User coordinate 14
6	User coordinate 5	16	User coordinate 15
7	User coordinate 6	17	User coordinate 16
8	User coordinate 7	18	Tool coordinate
9	User coordinate 8	19	Master tool coordinate

Coordinate Types
NX100/XRC
The following coordinate names correspond to the coordinate type data.

Coordinate type	Coordinate name	Coordinate type	Coordinate name
0	Base coordinate	14	User coordinate 13
1	Robot coordinate	15	User coordinate 14
2	User coordinate 1	16	User coordinate 15
3	User coordinate 2	17	User coordinate 16
4	User coordinate 3	18	User coordinate 17
5	User coordinate 4	19	User coordinate 18
6	User coordinate 5	20	User coordinate 19
7	User coordinate 6	21	User coordinate 20
8	User coordinate 7	22	User coordinate 21
9	User coordinate 8	23	User coordinate 22
10	User coordinate 9	24	User coordinate 23
11	User coordinate 10	25	User coordinate 24
12	User coordinate 11	26	Tool coordinate
13	User coordinate 12	27	Master tool coordinate

Form
The form data are represented by bit data in decimals.



CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC (Serial Port, Ethernet)
REFERENCE	"BscGetVarData" "BscGetVarData2"

■ BscStartJob

FUNCTION	Starts job. (A job to be started has the job name which is selected last.)
FORMAT	_declspec(dllexport) short APIENTRY BscStartJob(short nCid);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number
	OUT (Return) None
	Return Value 0 : Normal completion 1 : Current job name not specified Others : Error codes
REMARKS	**Call Condition** The BscSelectJob function must be called up and the current job name must be specified before executing this function. To restart a job during startup that has been held by the "BscHoldOn" function, release the hold by "BscHoldOff" function to call up the BscContinueJob function.
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	"BscContinueJob" "BscSelectJob"

■ BscSelectJob

FUNCTION	Selects a job.
FORMAT	_declspec(dllexport) short APIENTRY BscSelectJob(short nCid,char *name);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *name Job name storage pointer
	OUT (Return) None
	Return Value 0 : Normal completion Others : Error codes
REMARKS	**Job name** The character string for the job name is restricted as follows. Character string with 30 characters maximum. (8 characters that can be used in the MS-DOS) Specify "*" instead of the job name to select all the jobs.
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	"BscFindFirstMaster" "BscDeleteJob" "BscSetMasterJob" "BscSetLineNumber" "BscStartJob"

■ BscSelectMode

FUNCTION	Selects mode. (Teach or Play)
FORMAT	_declspec(dllexport) short APIENTRY BscSelectMode(short nCid,short mode);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number mode Selected mode
	OUT (Return) None
	Return Value 0 : Normal completion Others : Error codes
REMARKS	Selected Mode The selected mode is represented as follows. 1 : Teach 2 : Play
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)

■ BscSelLoopCycle

FUNCTION	Changes the cycle mode to auto mode.
FORMAT	_declspec(dllexport) short APIENTRY BscSelLoopCycle(short nCid);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number
	OUT (Return) None
	Return Value 0 : Normal completion Others : Error codes
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	"BscSelStepCycle" "BscSelOneCycle"

■ BscSelOneCycle

FUNCTION	Changes the cycle mode to 1-cycle mode.
FORMAT	_declspec(dllexport) short APIENTRY BscSelOneCycle(short nCid);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number
	OUT (Return) None
	Return Value 0 : Normal completion Others : Error codes
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	"BscSelStepCycle" "BscSelLoopCycle"

■ BscSelStepCycle

FUNCTION	Changes the cycle mode to step mode.
FORMAT	_declspec(dllexport) short APIENTRY BscSelStepCycle(short nCid);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number
	OUT (Return) None
	Return Value 0 : Normal completion Others : Error codes
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	"BscSelOneCycle" "BscSelLoopCycle"

■ BscSetLineNumber

FUNCTION	Sets a line number of current job.
FORMAT	_declspec(dllexport) short APIENTRY BscSetLineNumber(short nCid,short line);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number line Line number
	OUT (Return) None
	Return Value 0 : Normal completion Others : Error codes
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	"BscSelectJob"

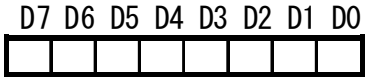
■ BscSetMasterJob

FUNCTION	Sets a job as a master job.
FORMAT	_declspec(dllexport) short APIENTRY BscSetMasterJob(short nCid);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number
	OUT (Return) None
	Return Value 0 : Normal completion Others : Error codes
REMARKS	**Call Condition** The BscSelectJob function must be called up and the registered job must be specified before executing this function.
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	"BscDeleteJob" "BscSelectJob"

■ BscReset

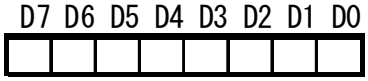
FUNCTION	Resets a robot alarm.
FORMAT	_declspec(dllexport) short APIENTRY BscReset(short nCid);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number
	OUT (Return) None
	Return Value 0 : Normal completion Others : Error codes
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	"BscCancel"

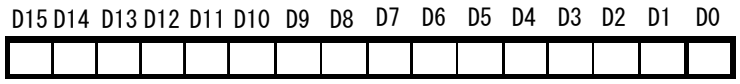
■ BscSetCtrlGroup

FUNCTION	Sets a control group.
FORMAT	_declspec(dllexport) short APIENTRY BscSetCtrlGroup(short nCid,short groupno);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number groupno Control group information to be set
	OUT (Return) None
	Return Value 0 : Normal completion Others : Error codes
REMARKS	**Restrictions** This function is effective only for transmission with the MRC. Refer to BscSetCtrlGroupDX for the transmission with the YRC1000/YRC1000micro/DX200/DX100. Refer to BscSetCtrlGroupXrc for the transmission with the FS100/NX100/XRC. When the power supply of robot controller is started up, robot 1, base 1, and station 1 (when a base and a stations exist) are specified. In a system with a base axis (such as travel axis), when the manipulator with this base axis is specified, this base axis is automatically specified. The following settings can not be made. －Selection of control axis which does not exist －Simultaneous specification of R1 and R2 －Specification of multiple number of stations
	Control Group Information The control group information is represented by bit data in decimals.  D0：R1（Robot 1） D1：R2（Robot 2） D2：S1（Station 1） D3：S2（Station 2） D4：S3（Station 3） D5：S4（Station 4） D6：S5（Station 5） D7：S6（Station 6）
CONTROLLER	MRC (Serial Port, Ethernet)

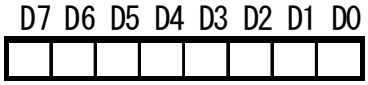
REFERENCE	"BscGetCtrlGroupDX" "BscSetCtrlGroupDX" "BscIsCtrlGroupDX" "BscSetCtrlGroupXrc" "BscGetCtrlGroupXrc" "BscIsCtrlGroupXrc" "BscIsTaskInfXrc" "BscGetCtrlGroup" "BscIsCtrlGroup" "BscIsTaskInf" "BscChangeTask"

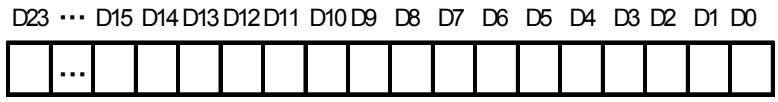
■ BscSetCtrlGroupXrc

FUNCTION	Sets a control group.
FORMAT	_declspec(dllexport) short APIENTRY BscSetCtrlGroupXrc(short nCid,short groupno1,short groupno2);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number groupno1 Control group information to be set (robot axis) groupno2 Control group information to be set (station axis)
	OUT (Return) groupno1 Control group information to be set (robot axis) groupno2 Control group information to be set (station axis)
	Return Value 0 : Normal completion Others : Error codes
REMARKS	**Restrictions** This function is effective only for transmission with the FS100/ NX100/XRC. Refer to BscSetCtrlGroupDX for the transmission with the YRC1000/YRC1000micro/DX200/DX100. Refer to BscSetCtrlGroup for transmission with the MRC. When the power supply of robot controller is started up, robot 1, base 1, and station 1 (when a base and a stations exist) are specified. In a system with a base axis (such as travel axis), when the manipulator with this base axis is specified, this base axis is automatically specified. The following settings can not be made. －Selection of control axis which does not exist －Simultaneous specification of R1 and R2 －Specification of multiple number of stations
	Control Group Information (Robot Axis) The control group information is represented by bit data in decimals.  D0：R1（Robot 1） D1：R2（Robot 2） D2：R3（Robot 3） D3：R4（Robot 4）

REMARKS	**Control Group Information (Station Axis)** The control group information is represented by bit data in decimals.  D0：S1（Station1） D1：S2（Station2） D2：S3（Station3） D3：S4（Station4） D4：S5（Station5） D5：S6（Station6） D6：S7（Station7） D7：S8（Station8） D8：S9（Station9） D9：S10（Station10） D10：S11（Station11） D11：S12（Station12）
CONTROLLER	FS100, NX100, XRC (Serial Port, Ethernet)
REFERENCE	"BscGetCtrlGroupDX" "BscSetCtrlGroupDX" "BscIsCtrlGroupDX" "BscGetCtrlGroupXrc" "BscIsCtrlGroupXrc" "BscIsTaskInfXrc" "BscSetCtrlGroup" "BscGetCtrlGroup" "BscIsCtrlGroup" "BscIsTaskInf" "BscChangeTask"

■ BscSetCtrlGroupDX

FUNCTION	Sets a control group.
FORMAT	_declspec(dllexport) short APIENTRY BscSetCtrlGroupDX(short nCid,long groupno1,long groupno2);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number groupno1 Control group information to be set (robot axis) groupno2 Control group information to be set (station axis)
	OUT (Return) groupno1 Control group information to be set (robot axis) groupno2 Control group information to be set (station axis)
	Return Value 0 : Normal completion Others : Error codes
REMARKS	**Restrictions** This function is effective only for transmission with the YRC1000/ YRC1000micro/DX200/DX100. Refer to BscSetCtrlGroupXrc for transmission with the FS100/ NX100/XRC. Refer to BscSetCtrlGroup for transmission with the MRC. When the power supply of robot controller is started up, robot 1, base 1, and station 1 (when a base and a stations exist) are specified. In a system with a base axis (such as travel axis), when the manipulator with this base axis is specified, this base axis is automatically specified. The following settings can not be made. －Selection of control axis which does not exist －Simultaneous specification of R1 and R2 －Specification of multiple number of stations
	Control Group Information (Robot Axis) The control group information is represented by bit data in decimals.  D0 : R1 (Robot 1) D1 : R2 (Robot 2) D2 : R3 (Robot 3) D3 : R4 (Robot 4) : : D7 : R8 (Robot 8)

REMARKS	**Control Group Information (Station Axis)** The control group information is represented by bit data in decimals.  D0 : S1 (Station1) D1 : S2 (Station2) D2 : S3 (Station3) D3 : S4 (Station4) D4 : S5 (Station5) D5 : S6 (Station6) D6 : S7 (Station7) D7 : S8 (Station8) D8 : S9 (Station9) D9 : S10 (Station10) D10 : S11 (Station11) D11 : S12 (Station12) ⋮ D23 : S24 (Station24)
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100 (Serial Port, Ethernet)
REFERENCE	"BscGetCtrlGroupDX" "BscIsCtrlGroupDX" "BscSetCtrlGroupXrc" "BscGetCtrlGroupXrc" "BscIsCtrlGroupXrc" "BscIsTaskInfXrc" "BscSetCtrlGroup" "BscGetCtrlGroup" "BscIsCtrlGroup" "BscIsTaskInf" "BscChangeTask"

■ BscServoOff

FUNCTION	Sets servo power supply OFF.
FORMAT	_declspec(dllexport) short APIENTRY BscServoOff(short nCid);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number
	OUT (Return) None
	Return Value 0 : Normal completion Others : Error codes
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	"BscServoOn" "BscIsServo"

■ BscServoOn

FUNCTION	Sets servo power supply ON.
FORMAT	_declspec(dllexport) short APIENTRY BscServoOn(short nCid);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number
	OUT (Return) None
	Return Value 0 : Normal completion Others : Error codes
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	"BscServoOff" "BscIsServo"

7.4 DCI Function

Job save, load, or variable load, save are automatically accomplished when the robot is under playback mode, by preparing the functions corresponding to the instructions.
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.

BscDCILoadSave
BscDCILoadSaveOnce
BscDCIGetPos
BscDCIGetPos2
BscDCIGetVarData
BscDCIGetVarDataEx
BscDCIPutPos
BscDCIPutPos2
BscDCIPutVarData
BscDCIPutVarDataEx

■ BscDCILoadSave

FUNCTION	Loads or saves a job with DCI instruction.
FORMAT	_declspec(dllexport) short APIENTRY BscDCILoadSave(short nCid,short timec);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number timec Waiting time for sending/receiving (sec)
	OUT (Return) None
	Return Value -1 : Failed to send/receive Others : Number of received jobs
REMARKS	**Number of Sending/Receiving** This function retries communication of the sending/receiving signal until the specified waiting time comes.
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	"BscDCILoadSaveOnce" "BscDCIGetPos" "BscDCIPutPos" "BscDCIGetPos2" "BscDCIPutPos2" "BscDCIGetVarData" "BscDCIPutVarData"

■ BscDCILoadSaveOnce

FUNCTION	Loads or saves a job with DCI instruction.
FORMAT	_declspec(dllexport) short APIENTRY BscDCILoadSaveO-nce(short nCid);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number
	OUT (Return) None
	Return Value -1 : Failed to send/receive Others : Number of received jobs
REMARKS	**Number of Sending/Receiving** This function waits indefinitely until sending/receiving request is sent from the robot. Communication is accomplished a single time when the request arrives.
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	"BscDCILoadSave" "BscDCIGetPos" "BscDCIPutPos" "BscDCIGetPos2" "BscDCIPutPos2" "BscDCIGetVarData" "BscDCIPutVarData"

■ BscDCIGetPos

FUNCTION	Gets a variable with DCI instruction.
FORMAT	_declspec(dllexport) short APIENTRY BscDCIGetPos(short nCid,short *type,short *rconf,double *p);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *type Variable type number (pointer) *rconf Form data (pointer) *p Head pointer to the numeric variable storage area
	OUT (Return) *rconf Form data (pointer) *p Head pointer to the numeric variable storage area
	Return Value -1 : Failed to send Others : Variable type number
REMARKS	**Variable Type Number** The variable type number is represented as follows.

Variable Contents	YRC1000/YRC1000micro/DX200 DX100/FS100/NX100/XRC/MRC	ERC
1	Byte type	
2	Integer type	
3	Double-precision type	
4	Real type	
5	Robot axis position type (pulse)	
6	Robot axis position type (XYZ)	
7	Base axis position type (pulse)	
8	Base axis position type (XYZ)	
9	Station axis position type (pulse)	-

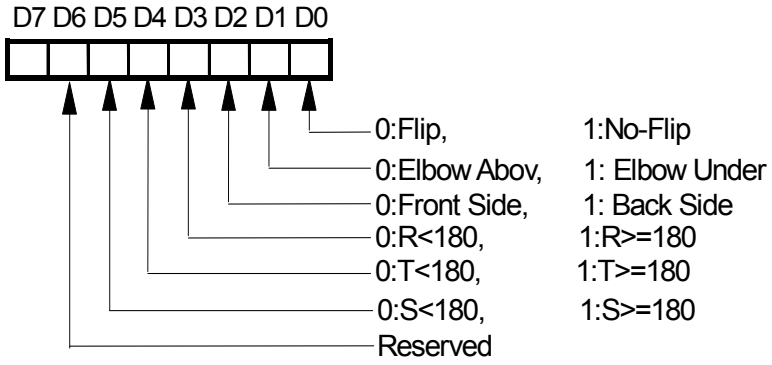
Variable type numberand and the storage area, number of values.

Variable Type Number	Storage area	Number of values
1～9	p	6

REMARKS

Form

The form data are represented by bit data in decimals.



* With the ERC or ERC2, the data from D3 to D7 are disregarded.
* With the MRC or MRC2, the data D5 and D7 are disregarded.

Content of the numeric variable storage area

Depending on the variable type, the numeric variable storage area contains the number of values indicated below.

Variable Type Number	Number of values	Content					
		P[0]	P[1]	P[2]	P[3]	P[4]	P[5]
1	1	Byte					
2	1	Integer	-	-	-	-	-
3	1	Double	-	-	-	-	-
4	1	Real	-	-	-	-	-
5	6	S-axis Pulses	L-axis Pulses	U-axis Pulses	R-axis Pulses	B-axis Pulses	T-axis Pulses
6	6	X-axis (mm)	Y-axis (mm)	Z-axis (mm)	Wrist angle Rx (deg)	Wrist angle Ry (deg)	Wrist angle Rz (deg)
7	6	Base Axis-1 Pulses	Base Axis-2 Pulses	Base Axis-3 Pulses	Base Axis-4 Pulses	Base Axis-5 Pulses	Base Axis-6 Pulses
8	6	Base Axis-1 (mm)	Base Axis-2 (mm)	Base Axis-3 (mm)	Base Axis-4 (mm)	Base Axis-5 (mm)	Base Axis-6 (mm)
9	6	Station Axis-1 Pulses	Station Axis-2 Pulses	Station Axis-3 Pulses	Station Axis-4 Pulses	Station Axis-5 Pulses	Station Axis-6 Pulses

CONTROLLER

YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet)
ERC (Serial Port)

REFERENCE

"BscDCILoadSave" "BscDCILoadSaveOnce" "BscDCIPutPos" "BscDCIPutPos2" "BscDCIPutVarData"

■ BscDCIGetPos2

FUNCTION	Gets a variable with DCI instruction. (robots with 7 axes or more)
FORMAT	_declspec(dllexport) short APIENTRY BscDCIGetPos2(short nCid,short *type,short *rconf,double *p,short *axisNum);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *type Variable type number (pointer) *rconf Form data (pointer) *p Head pointer to the numeric variable storage area *axisNum Number of axis (pointer)
	OUT (Return) *rconf Form data (pointer) *p Head pointer to the numeric variable storage area
	Return Value -1 : Failed to send Others : Variable type number
REMARKS	**Variable Type Number** The variable type number is represented as follows.

Variable Contents	YRC1000/YRC1000micro/DX200 DX100/FS100/NX100/XRC/MRC	ERC
1	Byte type	Byte type
2	Integer type	Integer type
3	Double-precision type	Double-precision type
4	Real type	Real type
5	Robot axis position type (pulse)	Robot axis position type (pulse)
6	Robot axis position type (XYZ)	Robot axis position type (XYZ)
7	Base axis position type (pulse)	External axis position type (pulse)
8	Base axis position type (XYZ)	External axis position type (XYZ)
9	Station axis position type (pulse)	-

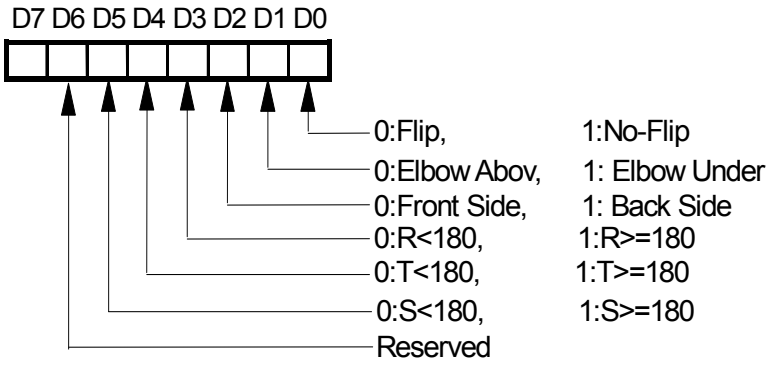
Variable type numberand and the storage area, number of values.

Variable Type Number	Storage area	Number of values
1～9	p	8

REMARKS

Form

The form data are represented by bit data in decimals.



* With the ERC or ERC2, the data from D3 to D7 are disregarded.
* With the MRC or MRC2, the data D5 and D7 are disregarded.

Content of the numeric variable storage area

Depending on the variable type, the numeric variable storage area contains the number of values indicated below.

Variable Type Number	Number of values	Content			
		P[0]	P[1]	P[2]	P[3]
1	1	Byte	-	-	-
2	1	Integer	-	-	-
3	1	Double	-	-	-
4	1	Real	-	-	-
5	8	S-axis Pulses	L-axis Pulses	U-axis Pulses	R-axis Pulses
6	6	X-axis (mm)	Y-axis (mm)	Z-axis (mm)	Wrist angle Rx (deg)
7	6	Base Axis-1 Pulses	Base Axis-2 Pulses	Base Axis-3 Pulses	Base Axis-4 Pulses
8	6	Base Axis-1 (mm)	Base Axis-2 (mm)	Base Axis-3 (mm)	Base Axis-4 (mm)
9	6	Station Axis-1 Pulses	Station Axis-2 Pulses	Station Axis-3 Pulses	Station Axis-4 Pulses

Variable Type Number	Number of values	Content			
		P[4]	P[5]	P[6]	P[7]
1	1	-	-	-	-
2	1	-	-	-	-
3	1	-	-	-	-
4	1	-	-	-	-
5	8	B-axis Pulses	T-axis Pulses	7^{th} axis Pulses	8^{th} axis Pulses
6	6	Wrist angle Ry (deg)	Wrist angle Rz (deg)	-	-
7	6	Base Axis-5 Pulses	Base Axis-6 Pulses	-	-
8	6	Base Axis-5 (mm)	Base Axis-6 (mm)	-	-
9	6	Station Axis-5 Pulses	Station Axis-6 Pulses	-	-

CONTROLLER

YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet)
ERC (Serial Port)

REFERENCE	"BscDCILoadSave" "BscDCILoadSaveOnce" "BscDCIPutPos" "BscDCIPutPos2" "BscDCIPutVarData"

■ BscDCIGetVarData

FUNCTION	Gets a variable with DCI instruction.
FORMAT	_declspec(dllexport) short APIENTRY BscDCIGetVarData(short nCid,short *type,short *rconf,double *p, char *str);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *type Variable type number (pointer) *rconf Form data (pointer) *p Head pointer to the numeric variable storage area *str Head pointer to the character variable storage area
	OUT (Return) *rconf Form data (pointer) *p Head pointer to the numeric variable storage area *str Head pointer to the character variable storage area
	Return Value -1 : Failed to send Others : Variable type number
REMARKS	**Restrictions** String variables can only be used with the YRC1000/YRC1000micro/DX200/DX100/FS100 or NX100 ver3.0 or later.
	Variable Type Number The variable type number is represented as follows.

Variable Contents	YRC1000/YRC1000micro DX200/DX100/FS100 NX100(v3.0 and after)	NX100 (before v3.0) XRC/MRC	ERC
1	Byte type		Byte type
2	Integer type		Integer type
3	Double-precision type		Double-precision type
4	Real type		Real type
5	Robot axis position type (pulse)		Robot axis position type (pulse)
6	Robot axis position type (XYZ)		Robot axis position type (XYZ)
7	Base axis position type (pulse)		External axis position type (pulse)
8	Base axis position type (XYZ)		External axis position type (XYZ)
9	Station axis position type (pulse)		-

NOTE String type is valid only for the NX100 V3.0 and later, or for the YRC1000/YRC1000micro/DX200/DX100/FS100.

REMARKS

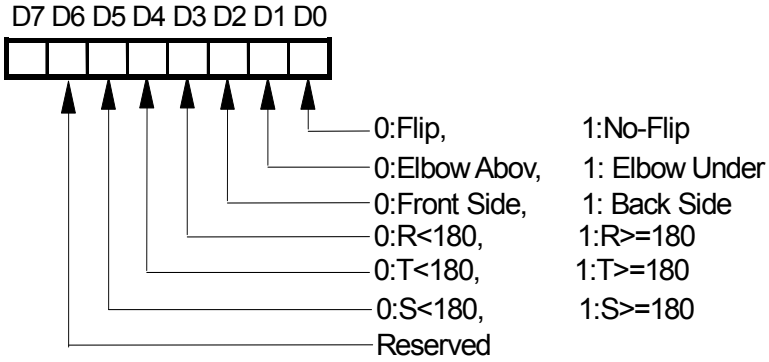
Variable type numberand and the storage area, number of values.

Variable Type Number	Storage area	Number of values
1～9	p	6
10	str	16

NOTE When this function is used to receive a string type variable make sure that the character variable storage area is allocated for 17 characters.
Declaration in Visual Basic: Dim S_Variable As String *17
Declaration in C++: char S_Variable[17]

Form

The form data are represented by bit data in decimals.



* With the ERC or ERC2, the data from D3 to D7 are disregarded.
* With the MRC or MRC2, the data D5 and D7 are disregarded.

Content of the numeric variable storage area

Depending on the variable type, the numeric variable storage area contains the number of values indicated below.

Variable Type Number	Number of values	Content					
		P[0]	P[1]	P[2]	P[3]	P[4]	P[5]
1	1	Byte					
2	1	Integer	-	-	-	-	-
3	1	Double	-	-	-	-	-
4	1	Real	-	-	-	-	-
5	6	S-axis Pulses	L-axis Pulses	U-axis Pulses	R-axis Pulses	B-axis Pulses	T-axis Pulses
6	6	X-axis (mm)	Y-axis (mm)	Z-axis (mm)	Wrist angle Rx (deg)	Wrist angle Ry (deg)	Wrist angle Rz (deg)
7	6	Base Axis-1 Pulses	Base Axis-2 Pulses	Base Axis-3 Pulses	Base Axis-4 Pulses	Base Axis-5 Pulses	Base Axis-6 Pulses
8	6	Base Axis-1 (mm)	Base Axis-2 (mm)	Base Axis-3 (mm)	Base Axis-4 (mm)	Base Axis-5 (mm)	Base Axis-6 (mm)
9	6	Station Axis-1 Pulses	Station Axis-2 Pulses	Station Axis-3 Pulses	Station Axis-4 Pulses	Station Axis-5 Pulses	Station Axis-6 Pulses

Content of the character variable storage area

Variable Type Number	Number of values	Content
		str
10	16	String

CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	"BscDCILoadSave" "BscDCILoadSaveOnce" "BscDCIPutPos" "BscDCIPutPos2" "BscDCIPutVarData"

■ BscDCIGetVarDataEx

FUNCTION	Gets a variable with DCI instruction.(robots with 7 axes or more)
FORMAT	_declspec(dllexport) short APIENTRY BscDCIGetVarDataEx(short nCid,short *type,long *rconf,double *p, char *str,short *axisNum);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *type Variable type number (pointer) *rconf Form data (pointer) *p Head pointer to the numeric variable storage area *str Head pointer to the character variable storage area *axisNum Number of axis (pointer)
	OUT (Return) *rconf Form data (pointer) *p Head pointer to the numeric variable storage area *str Head pointer to the character variable storage area *axisNum Number of axis (pointer)
	Return Value -1 : Failed to send Others : Variable type number
REMARKS	**Restrictions** String variables can only be used with the YRC1000/YRC1000micro/DX200/DX100/FS100 or NX100 ver3.0 or later.
	Variable Type Number The variable type number is represented as follows.

Variable Contents	YRC1000/YRC1000micro DX200/DX100/FS100 NX100(v3.0 and after)	NX100 (before v3.0) XRC/MRC	ERC
1	Byte type		Byte type
2	Integer type		Integer type
3	Double-precision type		Double-precision type
4	Real type		Real type
5	Robot axis position type (pulse)		Robot axis position type (pulse)
6	Robot axis position type (XYZ)		Robot axis position type (XYZ)
7	Base axis position type (pulse)		External axis position type (pulse)
8	Base axis position type (XYZ)		External axis position type (XYZ)
9	Station axis position type (pulse)		-

NOTE

- If the robot controller incluedes 7-axis robot, P-variable specification is for 7-axis robot.
- String type is valid only for the NX100 V3.0 and later, or for the YRC1000/YRC1000micro/DX200/DX100/FS100.

REMARKS

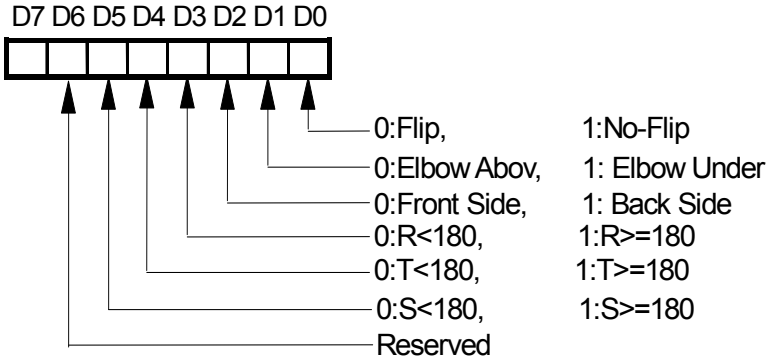
Variable type numberand and the storage area, number of values.

Variable Type Number	Storage area	Number of values
1～9	p	7
10	str	16

NOTE When this function is used to receive a string type variable make sure that the character variable storage area is allocated for 17 characters.
Declaration in Visual Basic: Dim S_Variable As String *17
Declaration in C++: char S_Variable[17]

Form

The form data are represented by bit data in decimals.



* With the ERC or ERC2, the data from D3 to D7 are disregarded.
* With the MRC or MRC2, the data D5 and D7 are disregarded.

Content of the numeric variable storage area

Depending on the variable type, the numeric variable storage area contains the number of values indicated below.

Variable Type Number	Number of values	Content						
		P[0]	P[1]	P[2]	P[3]	P[4]	P[5]	P[6]
1	1	Byte	-	-	-	-	-	-
2	1	Integer	-	-	-	-	-	-
3	1	Double	-	-	-	-	-	-
4	1	Real	-	-	-	-	-	-
5	7	S-axis Pulses	L-axis Pulses	U-axis Pulses	R-axis Pulses	B-axis Pulses	T-axis Pulses	E-axis Pulses
6	7	X-axis (mm)	Y-axis (mm)	Z-axis (mm)	Wrist angle Rx (deg)	Wrist angle Ry (deg)	Wrist angle Rz (deg)	Re (deg)
7	6	Base Axis-1 Pulses	Base Axis-2 Pulses	Base Axis-3 Pulses	Base Axis-4 Pulses	Base Axis-5 Pulses	Base Axis-6 Pulses	-
8	6	Base Axis-1 (mm)	Base Axis-2 (mm)	Base Axis-3 (mm)	Base Axis-4 (mm)	Base Axis-5 (mm)	Base Axis-6 (mm)	-
9	6	Station Axis-1 Pulses	Station Axis-2 Pulses	Station Axis-3 Pulses	Station Axis-4 Pulses	Station Axis-5 Pulses	Station Axis-6 Pulses	-

Content of the character variable storage area

Variable Type Number	Number of values	Content
		str
10	16	String

CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	"BscDCILoadSave" "BscDCILoadSaveOnce" "BscDCIPutPos" "BscDCIPutPos2" "BscDCIPutVarData" "BscDCIPutVarDataEx"

■ BscDCIPutPos

FUNCTION	Sets a variable with DCI instruction.
FORMAT	_declspec(dllexport) short APIENTRY BscDCIPutPos(short nCid,short type,short rconf,double *p);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number type Variable type number rconf Form data *p Head pointer to the numeric variable storage area
	OUT (Return) None
	Return Value -1 : Failed to send Others : Normal completion
REMARKS	**Variable Type Number** The variable type number is represented as follows.

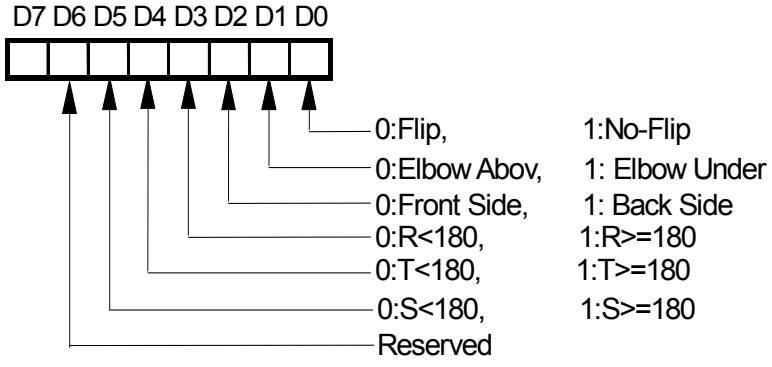
Variable Contents	YRC1000/YRC1000micro/DX200 DX100/FS100/NX100/XRC/MRC	ERC
1	Byte type	Byte type
2	Integer type	Integer type
3	Double-precision type	Double-precision type
4	Real type	Real type
5	Robot axis position type (pulse)	Robot axis position type (pulse)
6	Robot axis position type (XYZ)	Robot axis position type (XYZ)
7	Base axis position type (pulse)	External axis position type (pulse)
8	Base axis position type (XYZ)	External axis position type (XYZ)
9	Station axis position type (pulse)	-

Variable type numberand and the storage area, number of values.

Variable Type Number	Storage area	Number of values
1～9	p	6

Form

The form data are represented by bit data in decimals.



* With the ERC or ERC2, the data from D3 to D7 are disregarded.
* With the MRC or MRC2, the data D5 and D7 are disregarded.

REMARKS

Content of the numeric variable storage area

Depending on the variable type, the numeric variable storage area contains the number of values indicated below.

Variable Type Number	Number of values	Content					
		P[0]	P[1]	P[2]	P[3]	P[4]	P[5]
1	1	Byte					
2	1	Integer	-	-	-	-	-
3	1	Double	-	-	-	-	-
4	1	Real	-	-	-	-	-
5	6	S-axis Pulses	L-axis Pulses	U-axis Pulses	R-axis Pulses	B-axis Pulses	T-axis Pulses
6	6	X-axis (mm)	Y-axis (mm)	Z-axis (mm)	Wrist angle Rx (deg)	Wrist angle Ry (deg)	Wrist angle Rz (deg)
7	6	Base Axis-1 Pulses	Base Axis-2 Pulses	Base Axis-3 Pulses	Base Axis-4 Pulses	Base Axis-5 Pulses	Base Axis-6 Pulses
8	6	Base Axis-1 (mm)	Base Axis-2 (mm)	Base Axis-3 (mm)	Base Axis-4 (mm)	Base Axis-5 (mm)	Base Axis-6 (mm)
9	6	Station Axis-1 Pulses	Station Axis-2 Pulses	Station Axis-3 Pulses	Station Axis-4 Pulses	Station Axis-5 Pulses	Station Axis-6 Pulses

CONTROLLER

YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet)
ERC (Serial Port)

REFERENCE

"BscDCILoadSave" "BscDCILoadSaveOnce" "BscDCIGetPos" "BscDCIGetPos2" "BscDCIGetVarData"

■ BscDCIPutPos2

FUNCTION	Sets a variable with DCI instruction. (robots with 7 axes or more)
FORMAT	_declspec(dllexport) short APIENTRY BscDCIPutPos2(short nCid,short type,short rconf,double *p,short axisNum);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number type Variable type number rconf Form data *p Head pointer to the numeric variable storage area axisNum Number of axis
	OUT (Return) None
	Return Value -1 : Failed to send Others : Variable type number
REMARKS	**Variable Type Number** The variable type number is represented as follows.

Variable Contents	YRC1000/YRC1000micro/DX200 DX100/FS100/NX100/XRC/MRC	ERC
1	Byte type	Byte type
2	Integer type	Integer type
3	Double-precision type	Double-precision type
4	Real type	Real type
5	Robot axis position type (pulse)	Robot axis position type (pulse)
6	Robot axis position type (XYZ)	Robot axis position type (XYZ)
7	Base axis position type (pulse)	External axis position type (pulse)
8	Base axis position type (XYZ)	External axis position type (XYZ)
9	Station axis position type (pulse)	-

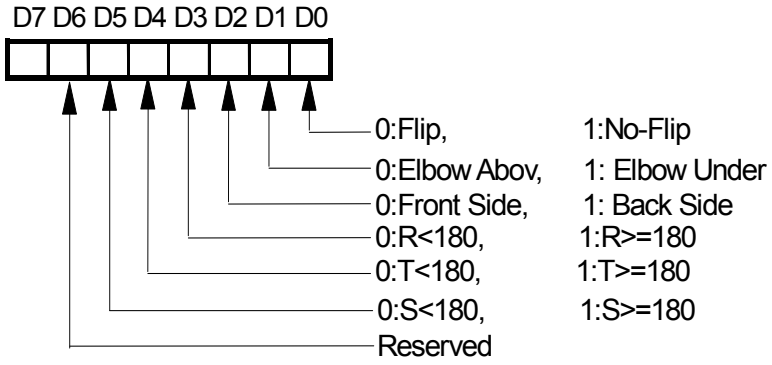
Variable type numberand and the storage area, number of values.

Variable Type Number	Storage area	Number of values
1～9	p	8

REMARKS

Form

The form data are represented by bit data in decimals.



* With the ERC or ERC2, the data from D3 to D7 are disregarded.
* With the MRC or MRC2, the data D5 and D7 are disregarded.

Content of the numeric variable storage area

Depending on the variable type, the numeric variable storage area contains the number of values indicated below.

Variable Type Number	Number of values	Content			
		P[0]	P[1]	P[2]	P[3]
1	1	Byte	-	-	-
2	1	Integer	-	-	-
3	1	Double	-	-	-
4	1	Real	-	-	-
5	8	S-axis Pulses	L-axis Pulses	U-axis Pulses	R-axis Pulses
6	6	X-axis (mm)	Y-axis (mm)	Z-axis (mm)	Wrist angle Rx (deg)
7	6	Base Axis-1 Pulses	Base Axis-2 Pulses	Base Axis-3 Pulses	Base Axis-4 Pulses
8	6	Base Axis-1 (mm)	Base Axis-2 (mm)	Base Axis-3 (mm)	Base Axis-4 (mm)
9	6	Station Axis-1 Pulses	Station Axis-2 Pulses	Station Axis-3 Pulses	Station Axis-4 Pulses

Variable Type Number	Number of values	Content			
		P[4]	P[5]	P[6]	P[7]
1	1	-	-	-	-
2	1	-	-	-	-
3	1	-	-	-	-
4	1	-	-	-	-
5	8	B-axis Pulses	T-axis Pulses	7^{th} axis Pulses	8^{th} axis Pulses
6	6	Wrist angle Ry (deg)	Wrist angle Rz (deg)	-	-
7	6	Base Axis-5 Pulses	Base Axis-6 Pulses	-	-
8	6	Base Axis-5 (mm)	Base Axis-6 (mm)	-	-
9	6	Station Axis-5 Pulses	Station Axis-6 Pulses	-	-

CONTROLLER

YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC (Serial Port, Ethernet)

REFERENCE	"BscDCILoadSave" "BscDCILoadSaveOnce" "BscDCIGetPos" "BscDCIGetPos2" "BscDCIGetVarData"

■ BscDCIPutVarData

FUNCTION	Sets a variable with DCI instruction.
FORMAT	_declspec(dllexport) short APIENTRY BscDCIPutVarData(short nCid,short type,short rconf,double *p, char *str);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number type Variable type number rconf Form data *p Head pointer to the numeric variable storage area *str Head pointer to the character variable storage area
	OUT (Return) None
	Return Value -1 : Failed to send Others : Variable type number
REMARKS	**Restrictions** String variables can only be used with the YRC1000/YRC1000micro/DX200/DX100/FS100 or NX100 ver3.0 or later.
	Variable Type Number The variable type number is represented as follows.

Variable Contents	YRC1000/YRC1000micro DX200/DX100/FS100 NX100(v3.0 and after)	NX100 (before v3.0) XRC/MRC	ERC
1	Byte type		Byte type
2	Integer type		Integer type
3	Double-precision type		Double-precision type
4	Real type		Real type
5	Robot axis position type (pulse)		Robot axis position type (pulse)
6	Robot axis position type (XYZ)		Robot axis position type (XYZ)
7	Base axis position type (pulse)		External axis position type (pulse)
8	Base axis position type (XYZ)		External axis position type (XYZ)
9	Station axis position type (pulse)		-

NOTE String type is valid only for the NX100 V3.0 and later, or for the YRC1000/YRC1000micro/DX200/DX100/FS100.

REMARKS

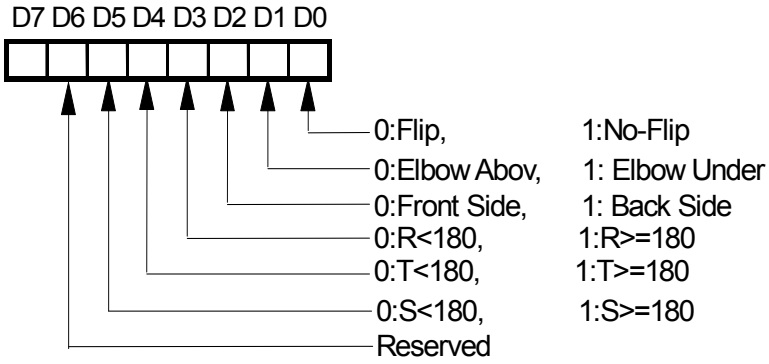
Variable type numberand and the storage area, number of values.

Variable Type Number	Storage area	Number of values
1～9	p	6
10	str	16

NOTE When this function is used to receive a string type variable make sure that the character variable storage area is allocated for 17 characters.
Declaration in Visual Basic: Dim S_Variable As String *17
Declaration in C++: char S_Variable[17]

Form

The form data are represented by bit data in decimals.



* With the ERC or ERC2, the data from D3 to D7 are disregarded.
* With the MRC or MRC2, the data D5 and D7 are disregarded.

Content of the numeric variable storage area

Depending on the variable type, the numeric variable storage area contains the number of values indicated below.

Variable Type Number	Number of values	Content					
		P[0]	P[1]	P[2]	P[3]	P[4]	P[5]
1	1	Byte					
2	1	Integer	-	-	-	-	-
3	1	Double	-	-	-	-	-
4	1	Real	-	-	-	-	-
5	6	S-axis Pulses	L-axis Pulses	U-axis Pulses	R-axis Pulses	B-axis Pulses	T-axis Pulses
6	6	X-axis (mm)	Y-axis (mm)	Z-axis (mm)	Wrist angle Rx (deg)	Wrist angle Ry (deg)	Wrist angle Rz (deg)
7	6	Base Axis-1 Pulses	Base Axis-2 Pulses	Base Axis-3 Pulses	Base Axis-4 Pulses	Base Axis-5 Pulses	Base Axis-6 Pulses
8	6	Base Axis-1 (mm)	Base Axis-2 (mm)	Base Axis-3 (mm)	Base Axis-4 (mm)	Base Axis-5 (mm)	Base Axis-6 (mm)
9	6	Station Axis-1 Pulses	Station Axis-2 Pulses	Station Axis-3 Pulses	Station Axis-4 Pulses	Station Axis-5 Pulses	Station Axis-6 Pulses

Content of the character variable storage area

Variable Type Number	Number of values	Content
		str
10	16	String

CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	"BscDCILoadSave" "BscDCILoadSaveOnce" "BscDCIGetPos" "BscDCIGetPos2" "BscDCIGetVarData"

■ BscDCIPutVarDataEx

FUNCTION	Sets a variable with DCI instruction.(robots with 7 axes or more)
FORMAT	_declspec(dllexport) short APIENTRY BscDCIPutVarDataEx(short nCid,short type,long rconf,double *p, char *str,short axisNum);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number type Variable type number rconf Form data *p Head pointer to the numeric variable storage area *str Head pointer to the character variable storage area axisNum Number of axis
	OUT (Return) None
	Return Value -1 : Failed to send Others : Variable type number
REMARKS	**Restrictions** String variables can only be used with the YRC1000/YRC1000micro/DX200/DX100/FS100 or NX100 ver3.0 or later.
	Variable Type Number The variable type number is represented as follows.

Variable Contents	YRC1000/YRC1000micro DX200/DX100/FS100 NX100(v3.0 and after)	NX100 (before v3.0) XRC/MRC	ERC
1	Byte type		Byte type
2	Integer type		Integer type
3	Double-precision type		Double-precision type
4	Real type		Real type
5	Robot axis position type (pulse)		Robot axis position type (pulse)
6	Robot axis position type (XYZ)		Robot axis position type (XYZ)
7	Base axis position type (pulse)		External axis position type (pulse)
8	Base axis position type (XYZ)		External axis position type (XYZ)
9	Station axis position type (pulse)		-

NOTE

- If the robot controller incluedes 7-axis robot, P-variable specification is for 7-axis robot.
- String type is valid only for the NX100 V3.0 and later, or for the YRC1000/YRC1000micro/DX200/DX100/FS100.

REMARKS

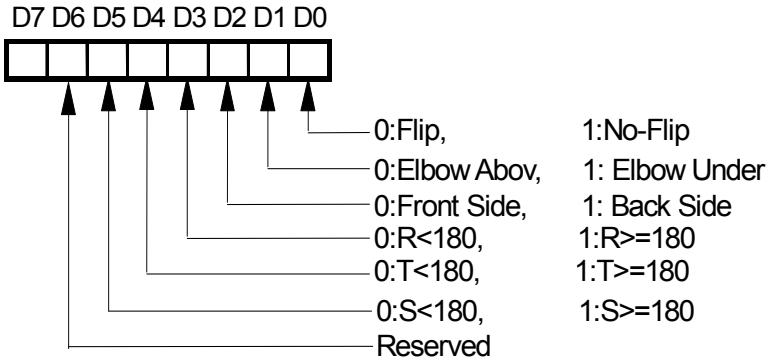
Variable type numberand and the storage area, number of values.

Variable Type Number	Storage area	Number of values
1～9	p	7
10	str	16

NOTE When this function is used to receive a string type variable make sure that the character variable storage area is allocated for 17 characters.
Declaration in Visual Basic: Dim S_Variable As String *17
Declaration in C++: char S_Variable[17]

Form

The form data are represented by bit data in decimals.



* With the ERC or ERC2, the data from D3 to D7 are disregarded.
* With the MRC or MRC2, the data D5 and D7 are disregarded.

Content of the numeric variable storage area

Depending on the variable type, the numeric variable storage area contains the number of values indicated below.

Variable Type Number	Number of values	Content						
		P[0]	P[1]	P[2]	P[3]	P[4]	P[5]	P[6]
1	1	Byte	-	-	-	-	-	-
2	1	Integer	-	-	-	-	-	-
3	1	Double	-	-	-	-	-	-
4	1	Real	-	-	-	-	-	-
5	7	S-axis Pulses	L-axis Pulses	U-axis Pulses	R-axis Pulses	B-axis Pulses	T-axis Pulses	E-axis Pulses
6	7	X-axis (mm)	Y-axis (mm)	Z-axis (mm)	Wrist angle Rx (deg)	Wrist angle Ry (deg)	Wrist angle Rz (deg)	Re (deg)
7	6	Base Axis-1 Pulses	Base Axis-2 Pulses	Base Axis-3 Pulses	Base Axis-4 Pulses	Base Axis-5 Pulses	Base Axis-6 Pulses	-
8	6	Base Axis-1 (mm)	Base Axis-2 (mm)	Base Axis-3 (mm)	Base Axis-4 (mm)	Base Axis-5 (mm)	Base Axis-6 (mm)	-
9	6	Station Axis-1 Pulses	Station Axis-2 Pulses	Station Axis-3 Pulses	Station Axis-4 Pulses	Station Axis-5 Pulses	Station Axis-6 Pulses	-

Content of the character variable storage area

Variable Type Number	Number of values	Content
		str
10	16	String

CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	"BscDCILoadSave" "BscDCILoadSaveOnce" "BscDCIGetPos" "BscDCIGetPos2" "BscDCIGetVarData" "BscDCIGetVarDataEx"

7.5 I/O Signal Read/Write Function

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

BscReadIO
BscReadIO2
BscWriteIO
BscWriteIO2

■ BscReadIO

FUNCTION	Reads specified count of coil status. Up to 256 coil numbers can be specified.
FORMAT	_declspec(dllexport) short APIENTRY BscReadIO(short nCid,short add,short num,short *stat);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number add Read starting address number num Number of read signals (up to 256) *stat Coil status
	OUT (Return) *stat Coil status
	Return Value -1 : Header number error 0 : Normal completion Others : Error codes
REMARKS	**Restrictions** This function is effective only for transmission with the XRC/MRC. Refer to the BscReadIO2 for transmission with the YRC1000/ YRC1000micro/DX200/DX100/FS100/NX100.
	Unnecessary Signals All unnecessary signals are set to 0 unless the number of the read data items is a multiple of 8.

REMARKS

List of I/O Signals

MRC

Signal	Signal Range	Name	Read	Write
0xxx	0010 to 0167 (128)	Robot universal input	○	×
1xxx	1010 to 1167 (128)	Robot universal output	○	×
2xxx	2010 to 2187 (144)	External input	○	×
3xxx	3010 to 3187 (144)	External output	○	×
4xxx	4010 to 4167 (128)	Robot specific input	○	×
5xxx	5010 to 5247 (192)	Robot specific output	○	×
6xxx	6010 to 6047 (32)	Timer/counter	×	×
7xxx	7010 to 7327 (256)	Auxiliary relay	○	×
8xxx	8010 to 8087 (64)	Control status signal	○	×
82xx	8210 to 8247 (32)	Pseudo input signal	○	×
9xxx	9010 to 9167 (128)	DL input	○	○

XRC

Signal	Signal Range	Name	Read	Write
0xxx	0010 to 0247 (192)	Robot universal input	○	×
1xxx	1010 to 1247 (192)	Robot universal output	○	×
2xxx	2010 to 2327 (256)	External input	○	×
3xxx	3010 to 3327 (256)	External output	○	×
4xxx	4010 to 4287 (224)	Robot specific input	○	×
5xxx	5010 to 5387 (304)	Robot specific output	○	×
6xxx	-	-	×	×
7xxx	7010 to 7887 (704)	Auxiliary relay	○	×
8xxx	8010 to 8127 (96)	Control status signal	○	×
82xx	8210 to 8247 (32)	Pseudo input signal	○	×
9xxx	9010 to 9167 (128)	Network input	○	○

CONTROLLER: XRC, MRC (Serial Port, Ethernet)

REFERENCE: "BscWriteIO" "BscReadIO2" "BscWriteIO2"

■ BscReadIO2

FUNCTION	Reads specified count of coil status. Up to 256 coil numbers can be specified.
FORMAT	_declspec(dllexport) short APIENTRY BscReadIO2(short nCid,DWORD add,short num,short *stat);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number add Read starting address number num Number of read signals (up to 256) *stat Coil status
	OUT (Return) *stat Coil status
	Return Value -1 : Header number error 0 : Normal completion Others : Error codes
REMARKS	**Restrictions** This function is effective only for transmission with the YRC1000/ YRC1000micro/DX200/DX100/FS100/NX100. Refer to the BscReadIO for transmission with the XRC/ MRC.
	Unnecessary Signals All unnecessary signals are set to 0 unless the number of the read data items is a multiple of 8.
	List of I/O Signals YRC1000

Signal	Signal Range	Name	Read	Writ
0xxxx	00010 to 05127 (4096)	Robot universal input	○	×
1xxxx	10010 to 15127 (4096)	Robot universal output	○	×
2xxxx	20010 to 25127 (4096)	External input	○	×
27xxx	27010 to 29567 (2048)	Network input	○	○
3xxxx	30010 to 35127 (4096)	External output	○	×
37xxx	37010 to 39567 (2048)	Network output	○	×
4xxxx	40010 to 42567 (2048)	Robot specific input	○	×
5xxxx	50010 to 55127 (4096)	Robot specific output	○	×
6xxxx	-	-	×	×
7xxxx	70010 to 79997 (7992)	Auxiliary relay	○	×
8xxxx	80010 to 85127 (4096)	Control status signal	○	×
87xxx	87010 to 87207 (160)	Pseudo input signal	○	×

REMARKS

YRC1000micro

Signal	Signal Range	Name	Read	Write
0xxxx	00010 to 05127 (4096)	Robot universal input	○	×
1xxxx	10010 to 15127 (4096)	Robot universal output	○	×
2xxxx	20010 to 21287 (1024)	External input	○	×
27xxx	27010 to 29567 (2048)	Network input	○	○
3xxxx	30010 to 31287 (1024)	External output	○	×
37xxx	37010 to 39567 (2048)	Network output	○	×
4xxxx	40010 to 42567 (2048)	Robot specific input	○	×
5xxxx	50010 to 55127 (4096)	Robot specific output	○	×
6xxxx	-	-	×	×
7xxxx	70010 to 79997 (7992)	Auxiliary relay	○	×
8xxxx	80010 to 85127 (4096)	Control status signal	○	×
87xxx	87010 to 87207 (160)	Pseudo input signal	○	×

DX200

Signal	Signal Range	Name	Read	Write
0xxxx	00010 to 05127 (4096)	Robot universal input	○	×
1xxxx	10010 to 15127 (4096)	Robot universal output	○	×
2xxxx	20010 to 25127 (4096)	External input	○	×
27xxx	27010 to 29567 (2048)	Network input	○	○
3xxxx	30010 to 35127 (4096)	External output	○	×
37xxx	37010 to 39567 (2048)	Network output	○	×
4xxxx	40010 to 41607 (1280)	Robot specific input	○	×
5xxxx	50010 to 53007 (2400)	Robot specific output	○	×
6xxxx	-	-	×	×
7xxxx	70010 to 79997 (7992)	Auxiliary relay	○	×
8xxxx	80010 to 80647 (512)	Control status signal	○	×
82xxx	82010 to 82207 (160)	Pseudo input signal	○	×

DX100

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

FS100

Signal	Signal Range	Name	Read	Write
0xxxx	00010 to 01287 (1024)	Robot universal input	○	×
1xxxx	10010 to 11287 (1024)	Robot universal output	○	×
2xxxx	20010 to 21287 (1024)	External input	○	×
25xxx	25010 to 26287 (1024)	Network input	○	○
3xxxx	30010 to 31287 (1024)	External output	○	×
35xxx	35010 to 36287 (1024)	Network output	○	×
4xxxx	40010 to 41607 (1280)	Robot specific input	○	×
5xxxx	50010 to 52007 (1600)	Robot specific output	○	×
6xxxx	-	-	×	×
7xxxx	70010 to 79997 (7992)	Auxiliary relay	○	×
8xxxx	80010 to 80647 (512)	Control status signal	○	×
82xxx	82010 to 82207 (160)	Pseudo input signal	○	×

REMARKS	NX100

Signal	Signal Range	Name	Read	Write
0xxxx	00010 to 01287 (1024)	Robot universal input	○	×
1xxxx	10010 to 11287 (1024)	Robot universal output	○	×
2xxxx	20010 to 21287 (1024)	External input	○	×
22xxx	22010 to 23287 (1024)	Network input	○	○
3xxxx	30010 to 31287 (1024)	External output	○	×
32xxx	32010 to 33287 (1024)	Network output	○	×
4xxxx	40010 to 40807 (640)	Robot specific input	○	×
5xxxx	50010 to 51007 (800)	Robot specific output	○	×
6xxxx	-	-	×	×
7xxxx	70010 to 79997 (7992)	Auxiliary relay	○	×
8xxxx	80010 to 80647 (512)	Control status signal	○	×
82xxx	82010 to 82127 (96)	Pseudo input signal	○	×

CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100 (Serial Port, Ethernet)
REFERENCE	"BscWriteIO2" "BscReadIO" "BscWriteIO"

■ BscWriteIO

FUNCTION	Writes specified count of coil status. Up to 256 coil numbers can be specified. Address numbers to be written are only of Nos. 9000's.
FORMAT	_declspec(dllexport) short APIENTRY BscWriteIO(short nCid,short add,short num,short *stat);
ARGUMENTS	I**N (Transfer)** nCid Communication handler ID number add Read starting address number num Number of read signals (up to 256) *stat Coil status
	OUT (Return) *stat Coil status
	Return Value -1 : Header number error 0 : Normal completion Others : Error codes
REMARKS	**Restrictions** This function is effective only for transmission with the XRC/MRC. Refer to the BscWriteIO2 for transmission with the YRC1000/ YRC1000micro/DX200/DX100/FS100/NX100.
	Unnecessary Signals All unnecessary data exist in the last data unless the number of the written data items is a multiple of 8.

REMARKS

List of I/O Signals

MRC

Signal	Signal Range	Name	Read	Write
0xxx	0010 to 0167 (128)	Robot universal input	○	×
1xxx	1010 to 1167 (128)	Robot universal output	○	×
2xxx	2010 to 2187 (144)	External input	○	×
3xxx	3010 to 3187 (144)	External output	○	×
4xxx	4010 to 4167 (128)	Robot specific input	○	×
5xxx	5010 to 5247 (192)	Robot specific output	○	×
6xxx	6010 to 6047 (32)	Timer/counter	×	×
7xxx	7010 to 7327 (256)	Auxiliary relay	○	×
8xxx	8010 to 8087 (64)	Control status signal	○	×
82xx	8210 to 8247 (32)	Pseudo input signal	○	×
9xxx	9010 to 9167 (128)	DL input	○	○

XRC

Signal	Signal Range	Name	Read	Write
0xxx	0010 to 0247 (192)	Robot universal input	○	×
1xxx	1010 to 1247 (192)	Robot universal output	○	×
2xxx	2010 to 2327 (256)	External input	○	×
3xxx	3010 to 3327 (256)	External output	○	×
4xxx	4010 to 4287 (224)	Robot specific input	○	×
5xxx	5010 to 5387 (304)	Robot specific output	○	×
6xxx	-	-	×	×
7xxx	7010 to 7887 (704)	Auxiliary relay	○	×
8xxx	8010 to 8127 (96)	Control status signal	○	×
82xx	8210 to 8247 (32)	Pseudo input signal	○	×
9xxx	9010 to 9167 (128)	Network input	○	○

CONTROLLER	XRC, MRC (Serial Port, Ethernet)
REFERENCE	"BscReadIO" "BscReadIO2" "BscWriteIO2"

■ BscWriteIO2

FUNCTION	Writes specified count of coil status. Up to 256 coil numbers can be specified. Address numbers to be written are only of or Nos.27000's (YRC1000/YRC1000micro/DX200) Nos.25000's (FS100/DX100) or Nos. 22000's (NX100).
FORMAT	_declspec(dllexport) short APIENTRY BscWriteIO2(short nCid,DWORD add,short num,short *stat);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number add Read starting address number num Number of read signals (up to 256) *stat Coil status
	OUT (Return) *stat Coil status
	Return Value -1 : Header number error 0 : Normal completion Others : Error codes
REMARKS	**Restrictions** This function is effective only for transmission with the YRC1000/ YRC1000micro/DX200/DX100/FS100/NX100. Refer to the BscWriteIO for transmission with the XRC/ MRC.
	Unnecessary Signals All unnecessary data exist in the last data unless the number of the written data items is a multiple of 8.
	List of I/O Signals YRC1000

Signal	Signal Range	Name	Read	Writ
0xxxx	00010 to 05127 (4096)	Robot universal input	○	×
1xxxx	10010 to 15127 (4096)	Robot universal output	○	×
2xxxx	20010 to 25127 (4096)	External input	○	×
27xxx	27010 to 29567 (2048)	Network input	○	○
3xxxx	30010 to 35127 (4096)	External output	○	×
37xxx	37010 to 39567 (2048)	Network output	○	×
4xxxx	40010 to 42567 (2048)	Robot specific input	○	×
5xxxx	50010 to 55127 (4096)	Robot specific output	○	×
6xxxx	-	-	×	×
7xxxx	70010 to 79997 (7992)	Auxiliary relay	○	×
8xxxx	80010 to 85127 (4096)	Control status signal	○	×
87xxx	87010 to 87207 (160)	Pseudo input signal	○	×

REMARKS

YRC1000micro

Signal	Signal Range	Name	Read	Write
0xxxx	00010 to 05127 (4096)	Robot universal input	○	×
1xxxx	10010 to 15127 (4096)	Robot universal output	○	×
2xxxx	20010 to 21287 (1024)	External input	○	×
27xxx	27010 to 29567 (2048)	Network input	○	○
3xxxx	30010 to 31287 (1024)	External output	○	×
37xxx	37010 to 39567 (2048)	Network output	○	×
4xxxx	40010 to 42567 (2048)	Robot specific input	○	×
5xxxx	50010 to 55127 (4096)	Robot specific output	○	×
6xxxx	-	-	×	×
7xxxx	70010 to 79997 (7992)	Auxiliary relay	○	×
8xxxx	80010 to 85127 (4096)	Control status signal	○	×
87xxx	87010 to 87207 (160)	Pseudo input signal	○	×

DX200

Signal	Signal Range	Name	Read	Write
0xxxx	00010 to 05127 (4096)	Robot universal input	○	×
1xxxx	10010 to 15127 (4096)	Robot universal output	○	×
2xxxx	20010 to 25127 (4096)	External input	○	×
27xxx	27010 to 29567 (2048)	Network input	○	○
3xxxx	30010 to 35127 (4096)	External output	○	×
37xxx	37010 to 39567 (2048)	Network output	○	×
4xxxx	40010 to 41607 (1280)	Robot specific input	○	×
5xxxx	50010 to 53007 (2400)	Robot specific output	○	×
6xxxx	-	-	×	×
7xxxx	70010 to 79997 (7992)	Auxiliary relay	○	×
8xxxx	80010 to 80647 (512)	Control status signal	○	×
82xxx	82010 to 82207 (160)	Pseudo input signal	○	×

DX100

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

FS100

Signal	Signal Range	Name	Read	Write
0xxxx	00010 to 01287 (1024)	Robot universal input	○	×
1xxxx	10010 to 11287 (1024)	Robot universal output	○	×
2xxxx	20010 to 21287 (1024)	External input	○	×
25xxx	25010 to 26287 (1024)	Network input	○	○
3xxxx	30010 to 31287 (1024)	External output	○	×
35xxx	35010 to 36287 (1024)	Network output	○	×
4xxxx	40010 to 41607 (1280)	Robot specific input	○	×
5xxxx	50010 to 52007 (1600)	Robot specific output	○	×
6xxxx	-	-	×	×
7xxxx	70010 to 79997 (7992)	Auxiliary relay	○	×
8xxxx	80010 to 80647 (512)	Control status signal	○	×
82xxx	82010 to 82207 (160)	Pseudo input signal	○	×

REMARKS	NX100

Signal	Signal Range	Name	Read	Write
0xxxx	00010 to 01287 (1024)	Robot universal input	○	×
1xxxx	10010 to 11287 (1024)	Robot universal output	○	×
2xxxx	20010 to 21287 (1024)	External input	○	×
22xxx	22010 to 23287 (1024)	Network input	○	○
3xxxx	30010 to 31287 (1024)	External output	○	×
32xxx	32010 to 33287 (1024)	Network output	○	×
4xxxx	40010 to 40807 (640)	Robot specific input	○	×
5xxxx	50010 to 51007 (800)	Robot specific output	○	×
6xxxx	-	-	×	×
7xxxx	70010 to 79997 (7992)	Auxiliary relay	○	×
8xxxx	80010 to 80647 (512)	Control status signal	○	×
82xxx	82010 to 82127 (96)	Pseudo input signal	○	×

CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100 (Serial Port, Ethernet)
REFERENCE	"BscReadIO2" "BscReadIO" "BscWriteIO"

7.6 Other Functions

The following functions are also available.

BscClose
BscCommand
BscConnect
BscDisConnect
BscDiskFreeSizeGet
BscEnforcedClose
BscGets
BscInBytes
BscOpen
BscOutBytes
BscPuts
BscReConnect
BscReStartJob
BscSetBreak
BscSetCom
BscSetCondBSC
BscSetEServerMode
BscSetEther
BscStatus

■ BscClose

FUNCTION	Releases a communication handler.
FORMAT	_declspec(dllexport) short APIENTRY BscClose(short nCid);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number
	OUT (Return) None
	Return Value 0 : Normal completion Others : Failed to release
REMARKS	**Call Condition** It is necessary to disconnect the communications lines BscDisConnect function before calling this function.
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	"BscOpen" "BscDisConnect"

■ BscCommand

FUNCTION	Sends a transmission command.
FORMAT	_declspec(dllexport) short APIENTRY BscCommand(short nCid,char *h_buf,char *d_buf,short fforever);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *h_buf Header character string pointer *d_buf Data character string pointer fforever Robot response; 0 : No wait; 1 : Wait
	OUT (Return) None
	Return Value -1 : Failed to send Others : Normal completion
REMARKS	**Header Character String** The header character string is represented by the header number and sub code number, in that order.
	Data Character String The data character string is represented by the data queue plus \r (carriage return) at the end. <Example> When sending the "SERVO ON" command If Header number 01 Sub code number 000, then Header character string 01,000 Data character string SVON 1\r
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)

■ BscConnect

FUNCTION	Connects communications lines.
FORMAT	_declspec(dllexport) short APIENTRY BscConnect(short nCid);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number
	OUT (Return) None
	Return Value 0 : Error 1 : Normal completion
REMARKS	**Call Condition** Before calling this function, it is necessary to set the communications lines with BscOpen function followed by BscSetCom function (serial port) or BscSetEther function (Ethernet) or BscSetEServerMode function (Ethernet Server).
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	"BscOpen" "BscSetCom"(Serial Port) "BscSetEther"(Ethernet) "BscDisConnect"

■ BscDisConnect

FUNCTION	Disconnects communications lines.
FORMAT	_declspec(dllexport) short APIENTRY BscDisConnect(short nCid);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number
	OUT (Return) None
	Return Value 0 : Error 1 : Normal completion
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	"BscClose" "BscConnect"

■ BscDiskFreeSizeGet

FUNCTION	Gets free capacity of the specified drive.
FORMAT	_declspec(dllexport) short APIENTRY BscDiskFreeSizeGet(short nCid,short dno,long *dsize);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number dno Drive number 1:A to 26:Z dsize Free capacity pointer
	OUT (Return) dsize Free capacity pointer
	Return Value 0 : Error 1 : Normal completion
REMARKS	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)

■ BscEnforcedClose

FUNCTION	Closes compulsorily.
FORMAT	_declspec(dllexport) short APIENTRY BscEnforcedClose(short nCid);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number
	OUT (Return) None
	Return Value 0 : Normal completion Others : Failed to release
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	"BscOpen" "BscDisConnect"

■ BscGets

FUNCTION	Sends a character string by transmission in TTY level.
FORMAT	_declspec(dllexport) short APIENTRY BscGets(short nCid,char *bufptr,WORD bsize,WORD *plengets);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *bufptr Received character string pointer bsize Maximum character count plengets Received character count
	OUT (Return) *bufptr Received character string pointer
	Return Value Received character count
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	"BscPuts" "BscInBytes" "BscOutBytes"

■ BscInBytes

FUNCTION	Returns the number of characters which are received by transmission in TTY level.
FORMAT	_declspec(dllexport) short APIENTRY BscInBytes(short nCid);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number
	OUT (Return) None
	Return Value Received character count
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	"BscGets" "BscPuts" "BscOutBytes"

■ BscOpen

FUNCTION	Gets a communication handler.
FORMAT	_declspec(dllexport) short APIENTRY BscOpen(char *path,short mode);
ARGUMENTS	**IN (Transfer)** *path Communication current directory storage pointer mode Communication type : 1 (=0x01) : serial port, 16 (=0x10) : Ethernet 256 (=0x100) : Ethernet Server
	OUT (Return) None
	Return Value -1 : Acquisition Failure -8 : License Error Others : Communication handler ID number
REMARKS	**Call Condition** By calling the BscSetCom function (serial port) or BscSetEther (Ethernet) BscSetEServerMode(Ethernet Server) BscConnect function after calling this function, communications can be started.
	Type of Communications Only 1 (=1x01): serial port or 16 (=0x10): Ethernet or 256 (=0x100): Ethernet Server can be used. For any other values, an error occurs.
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	"BscClose" "BscSetCom"(Serial Port) "BscSetEther"(Ethernet) "BscSetEServerMode"(Ethernet Server) "BscConnect"

■ BscOutBytes

FUNCTION	Returns the remaining number of characters which are sent by transmission in TTY level.
FORMAT	_declspec(dllexport) short APIENTRY BscOutBytes(short nCid);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number
	OUT (Return) None
	Return Value Sending character count
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	"BscGets" "BscPuts" "BscInBytes"

■ BscPuts

FUNCTION	Sends a character string by transmission in TTY level.
FORMAT	_declspec(dllexport) short APIENTRY BscPuts(short nCid,char *bufptr,WORD length);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *bufptr Sending character string pointer length Sending character count
	OUT (Return) None
	Return Value Sending character count
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	"BscGets" "BscInBytes" "BscOutBytes"

■ BscReConnect

FUNCTION	Connects communications lines again.
FORMAT	_declspec(dllexport) short APIENTRY BscReConnect(short nCid);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number
	OUT (Return) None
	Return Value 0 : Error 1 : Normal completion
REMARKS	**Call Condition** Before calling this function, it is necessary to set the communications lines with the BscOpen function and BscSetCom function (serial port), or BscSetEther function (Ethernet) or BscSetEServerMode function (Ethernet Server).
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	"BscOpen" "BscSetCom"(Serial Port) "BscSetEther"(Ethernet) "BscDisConnect"

■ BscReStartJob

FUNCTION	Starts job again.
FORMAT	_declspec(dllexport) short APIENTRY BscReStartJob(short nCid);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number
	OUT (Return) None
	Return Value 0 : Normal completion 1 : Current job name not specified Others : Error codes
REMARKS	**Call Condition** The BscSelectJob function must be called up and the current job name must be specified before executing this function. To restart a job during startup that has been held by the BscHoldOn function, release the hold by BscHoldOff function to call up the BscContinueJob function.
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	"BscContinueJob" "BscSelectJob"

■ BscSetBreak

FUNCTION	Cancels transmission.
FORMAT	_declspec(dllexport) short APIENTRY BscSetBreak(short nCid,short flg);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number flg Forced completion flag ; 0 : No forced completion, 1 : Forced completion
	OUT (Return) None
	Return Value -1 : Communication handler error 0 : Normal completion
REMARKS	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)

■ BscSetCom

FUNCTION	Sets communications parameters of the serial port.
FORMAT	_declspec(dllexport) short APIENTRY BscSetCom(short nCid, short port,DWORD baud,short parity,short clen,short stp);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number port Communication port number 1 : COM1, 2 : COM2, 3 : COM3, 4 : COM4, ・・・, 255 : COM255 baud Baud rate 150, 300, 600, 1200, 2400, 4800, 9600, 19200 parity Parity 0: None, 1: Odd, 2: Even clen Data length 7: 7 bits, 8: 8 bits stp Stop bit 0 : 1 bit, 1 : 1.5 bits, 2 : 2 bits
	OUT (Return) None
	Return Value 0 : Error 1 : Normal completion
REMARKS	**Call Condition** Before calling this function, it is necessary to get the communication handler of the serial port with BscOpen function. After calling this function, communications can be done using the BscConnect function.
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC, ERC (Serial Port)
REFERENCE	"BscOpen" "BscConnect"

■ BscSetCondBSC

FUNCTION	Sets a communication control timer or retry counter.
FORMAT	_declspec(dllexport) short APIENTRY BscSetCondBSC(short nCid,short timerA,short timerB,short rtyR,short rtyW);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number timerA Timer A (control timer, unit: msec) timerB Timer B (text timer, unit: msec) rtyR Sequence retry counter rtyW Text retry counter
	OUT (Return) None
	Return Value -1 : Communication handler error 0 : Normal completion
REMARKS	**Initial Value** timerA 10000(msec) timerB 30000(msec) rtyR 3 rtyW 3
	Note: This function is used to change the parameters of MOTOCOM32 on the personal computer. To change the robot controller transmission parameters (control timers, retry counter), use the programming pendant of the robot controller.
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)

■ BscSetEServerMode

FUNCTION	Sets the communication parameters for Ethernet Server function.
FORMAT	_declspec(dllexport) short APIENTRY BscSetEServerMode(short nCid, char *IPaddr, short Mode);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *IPaddr IP address of receiver Mode Server communication mode 1: Server mode, -1: Exclusive mode
	OUT (Return) None
	Return Value 0 : Error 1 : Normal completion
REMARKS	**Call Condition** Before calling this function, it is necessary to get the communication handler of the serial port with BscOpen function. After calling this function, communications can be done using the BscConnect function.
	Mode Specification This function specifies the Ethernet Server mode. The following modes are available: Server Mode: The network connection between the controller and application is ended after each command. Because of this, multiple applications can communicate simultaneously to the same controller via the network connection. Exclusive Mode: The network connection with the controller is exclusive to a single application. After the BscConnect function is called, the network connection is maintained until the BscDisConnect function is called. Because the differences between these network connection modes are processed inside MOTOCOM32, it is not necessary to make any change to the connection method on application side, other than to appoint the mode with this function.
	Restrictions DCI function is not supported with Ethernet Server function communication. The function is only available for communication with the YRC1000/ YRC1000micro/DX200/DX100/FS100/NX100.
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100 (Ethernet)
REFERENCE	"BscOpen" "BscConnect"

■ BscSetEther

<table>
<tr><td>FUNCTION</td><td>Sets parameters for Ethernet communications.</td></tr>
<tr><td>FORMAT</td><td>_declspec(dllexport) short APIENTRY BscSetEther(short nCid, char *IPaddr, short mode, HWND hWnd);</td></tr>
<tr><td>ARGUMENTS</td><td>IN (Transfer)
nCid Communication handler ID number
IPaddr IP address of receiver
mode Execution mode 0: Client, 1: Stand alone
hWnd Window handle</td></tr>
<tr><td></td><td>OUT (Return)
None</td></tr>
<tr><td></td><td>Return Value
0 : Error
1 : Normal completion</td></tr>
<tr><td>REMARKS</td><td>Call Condition
Before calling this function, it is necessary to get the communication handler of the serial port with BscOpen function. After calling this function, communications can be done using the BscConnect function.</td></tr>
<tr><td></td><td>Execution Mode and IP Address of Receiver
Select the corresponding "mode" argument to the communications function to be used. That "mode" argument determines whether the application to be operated by personal computer is to be client or server.
<table><tr><th>Function</th><th>Mode
(Personal Computer)</th><th>IPaddr</th></tr><tr><td>Host Control</td><td>0 (Client)</td><td>Must be always set.</td></tr><tr><td>DCI
Stand Alone</td><td>1 (Server)</td><td>Can be omitted.</td></tr></table>When the personal computer is set to server (mode = 1), setting YRC1000/YRC1000micro/DX200/DX100/FS100/NX100/XRC/MRC IP address to the IP address (IPaddr) determines that the server is a specified client server.

<Example>
Client:
BscSetEther(nCid, "192.168.10.10", 0, hWnd);
Specified client server (IP address must be always written.):
BscSetEther(nCid, "192.168.10.10", 1, hWnd);
Some client servers (IP address is not written.):
BscSetEther(nCid, "", 1, hWnd);</td></tr>
<tr><td>CONTROLLER</td><td>YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Ethernet)</td></tr>
<tr><td>REFERENCE</td><td>"BscOpen" "BscConnect"</td></tr>
</table>

■ BscStatus

FUNCTION	Reads the status.
FORMAT	_declspec(dllexport) short APIENTRY BscStatus(short nCid,char *hpt,char *dpt,unsigned short sz,char *rbuf);
ARGUMENTS	**IN (Transfer)** nCid Communication handler ID number *hpt Header character string pointer *dpt Sending data character string pointer sz Sending data character string size *rbuf Received data character string pointer
	OUT (Return) *rbuf Received data character string pointer
	Return Value 0 : Normal completion Others : Error codes
CONTROLLER	YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, XRC, MRC (Serial Port, Ethernet) ERC (Serial Port)
REFERENCE	"BscGetStatus"

7.7 DLL Functions Corresponding to Transmission-related Key words

7.7.1 DLL Functions Related to Transmission Commands

■ Read/Monitoring System

RALARM	BscGetError BscGetError2 BscGetFirstAlarm BscGetNextAlarm BscIsErrorCode
RPOS	BscIsLoc
RPOSJ	BscIsLoc BscGetPulsePos
RSTATS	BscGetStatus BscIsCycle BscIsServo BscIsTeachMode BscIsPlayMode BscIsRemoteMode BscIsHold BscIsAlarm BscIsError
RJSEQ	BscIsJobName BscIsJobLine BscIsJobStep
RPOSC	BscIsRobotPos BscGetCartPos
JWAIT	BscJobWait
RGROUP	BscGetCtrlGroupDX BscIsCtrlGroupDX BscGetCtrlGroupXrc BscIsCtrlGroupXrc BscIsTaskInfXrc BscGetCtrlGroup BscIsCtrlGroup BscIsTaskInf
RTRQ	BscGetTorque
RMAXTRQ	BscGetMaxTorque
RENCTMP	BscGetEncTmp

RSYSTM	BscGetSysTime

■ Read/Data Access System

RJDIR	BscFindFirst BscFindFirstMaster BscFindNext BscFindNextMaster
RUFRAME	BscGetUFrame
UPLOAD	BscUpload BscUploadEx
SAVEV	BscGetVarData BscGetVarData2 BscHostGetVarData BscHostGetVarDataM BscGetVarDataEx

■ Operation System

HOLD	BscHoldOn BscHoldOff
RESET	BscReset
CANCEL	BscCancel
MODE	BscSelectMode
CYCLE	BscSelStepCycle BscSelOneCycle BscSelLoopCycle
HLOCK	BscOPLock BscOPUnLock
MDSP	BscMDSP
SVON	BscServoOn BscServoOff
CGROUP	BscSetCtrlGroupDX BscSetCtrlGroupXrc BscSetCtrlGroup
CTASK	BscChangeTask

■ Editing System

DELETE	BscDeleteJob
WUFRAME	BscPutUFrame BscPutUFrameEx2
CVTRJ	BscConvertJobP2R
DOWNLOAD	BscDownload BscDownloadEx
CVTSJ	BscConvertJobR2P
LOADV	BscPutVarData BscPutVarData2 BscHostPutVarData BscHostPutVarDataM BscPutVarDataEx

■ Job Selection System

SETMJ	BscSetMasterJob
JSEQ	BscSetLineNumber

■ Startup System

START	BscStartJob BscContinueJob
MOVJ	BscMovj BscMovjEx BscMov BscMovEx BscMovEx2
MOVL	BscMovl BscMovlEx BscMov BscMovEx BscMovEx2
IMOV	BscImov BscImovEx BscImovEx2 BscMov BscMovEx BscMovEx2
PMOVJ	BscPMovj BscPMovjEx BscPMov BscPMovEx

PMOVL	BscPMovl BscPMovlEx BscPMov BscPMovEx

■ Other DLL Functions

BscCommand
BscStatus

7.7.2 DLL Functions Related to DCI Function

LOADJ	BscDCILoadSave BscDCILoadSaveOnce
SAVEJ	BscDCILoadSave BscDCILoadSaveOnce
LOADV	BscDCIGetPos BscDCIGetPos2 BscDCIGetVarData BscDCIGetVarDataEx
SAVEV	BscDCIPutPos BscDCIPutPos2 BscDCIPutVarData BscDCIPutVarDataEx

7.7.3 DLL Functions Related to I/O Read/Write

I/O Read	BscReadIO BscReadIO2
I/O Write	BscWriteIO BscWriteIO2

7.7.4 DLL Functions Related to Personal Computer Communications Port

Port Connection	BscOpen BscSetCom BscSetEServerMode BscSetEther BscConnect BscReConnect
Port Disconnection	BscClose BscDisConnect BscEnforcedClose
Transmission Parameter Setting	BscSetCondBSC

7.7.5 Other DLL Functions

BscDiskFreeSizeGet
BscGets
BscInBytes
BscOutBytes
BscPuts
BscSetBreak

8 Appendix

8.1 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 " 1.4 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 " 1.4 Hardware Lock Key " in the following section.)

MOTOCOM32
OPERATOR'S MANUAL

For inquiries or after-sales service on this product, contact your local YASKAWA representative as shown below.

YASKAWA ELECTRIC CORPORATION
2-1 Kurosakishiroishi, Yahatanishi-ku, Kitakyushu, 806-0004, Japan
Phone: +81-93-645-7703 Fax: +81-93-645-7802
www.yaskawa.co.jp

YASKAWA AMERICA, INC. (MOTOMAN ROBOTICS DIVISION)
100 Automation Way, Miamisburg, OH 45342, U.S.A.
Phone: +1-937-847-6200 Fax: +1-937-847-6277
www.motoman.com

YASKAWA EUROPE GmbH (ROBOTICS DIVISION)
Yaskawastrasse 1, 85391, Allershausen, Germany
Phone: +49-8166-90-100 Fax: +49-8166-90-103
www.yaskawa.eu.com

YASKAWA NORDIC AB
Verkstadsgatan 2, Box 504, SE-385 25 Torsas, Sweden
Phone: +46-480-417-800 Fax: +46-486-414-10
www.yaskawa.se

YASKAWA ELECTRIC (CHINA) CO., LTD.
22F, One Corporate Avenue, No.222 Hubin Road, Huangpu District, Shanghai 200021, China
Phone: +86-21-5385-2200 Fax: +86-21-5385-3299
www.yaskawa.com.cn

YASKAWA SHOUGANG ROBOT CO., LTD.
No.7 Yongchang North Road, Beijing E&T Development Area, Beijing 100076, China
Phone: +86-10-6788-2858 Fax: +86-10-6788-2878
www.ysr-motoman.cn

YASKAWA ELECTRIC KOREA CORPORATION
35F, Three IFC, 10 Gukjegeumyung-ro, Yeongdeungpo-gu, Seoul, 07326, Korea
Phone: +82-2-784-7844 Fax: +82-2-784-8495
www.yaskawa.co.kr

YASKAWA ELECTRIC TAIWAN CORPORATION
12F, No.207, Sec. 3, Beishin Rd., Shindian District, New Taipei City 23143, Taiwan
Phone: +886-2-8913-1333 Fax: +886-2-8913-1513
www.yaskawa.com.tw

YASKAWA ASIA PACIFIC PTE. LTD.
30A Kallang Place, #06-01, 339213, Singapore
Phone: +65-6282-3003 Fax: +65-6289-3003
www.yaskawa.com.sg

YASKAWA ELECTRIC (THAILAND) CO., LTD.
59, 1st-5th Floor, Flourish Building, Soi Ratchadapisek 18, Ratchadapisek Road, Huaykwang, Bangkok 10310, Thailand
Phone: +66-2-017-0099 Fax: +66-2-017-0199
www.yaskawa.co.th

PT. YASKAWA ELECTRIC INDONESIA
Secure Building-Gedung B Lantai Dasar & Lantai 1 Jl. Raya Protokol Halim Perdanakusuma, Jakarta 13610, Indonesia
Phone: +62-21-2982-6470 Fax: +62-21-2982-6471
www.yaskawa.co.id

YASKAWA INDIA PRIVATE LIMITED (ROBOTICS DIVISION)
#426, Udyog Vihar Phase-IV, Gurugram, Haryana 122016, India
Phone: +91-124-475-8500 Fax: +91-124-475-8542
www.yaskawaindia.in

YASKAWA

YASKAWA ELECTRIC CORPORATION

MANUAL NO.
HW9482689