MITSUBISHI ELECTRIC

Mitsubishi Electric Industrial Robots

CR750/CRn-700/ CRn-500 Series
Communication Middleware (MelfaRXM.ocx)
Instruction Manual

MELFA

BFP-A8253-E

⚠ Safety Precautions

Always read the following precautions and the "Safety Manual" of Robot Controller before starting use of the robot to learn the required measures to be taken.

A. These show precautions based on Labor Health and Safety Regulations (Articles 36, 104, 150, 151).

⚠Caution	For the sake of safety, teaching work should only be performed by workers who have received special education. (The same is true for any maintenance work done with the power source not cut off.) →Implementation of safety education
⚠Caution	For teaching work, prepare work regulations concerning robot operation methods and procedures, measures for when there is an abnormality and when restarting, etc. Perform teaching work according to these regulations. (The same is true for any maintenance work done with the power source not cut off.) →Prepare work regulations.
⚠Warning	For teaching work, set up a device that can stop operation immediately. (The same is true for any maintenance work done with the power source not cut off.) →Emergency stop switch setting
⚠Caution	During teaching work, label the start switch etc. to indicate that teaching work is underway. (The same is true for any maintenance work done with the power source not cut off.) →Display that teaching work is underway
⚠Danger	During operation, set up a fence or barrier to prevent contact between workers and the robot. → Setting up a safety fence
⚠Caution	Determine a uniform signal to relevant staff for the start of operation and use that signal. → Signal for the start of operation
⚠Caution	For maintenance work, in principle, cut off the power and label the start switch etc. to indicate that maintenance work is underway. →Display that maintenance work is underway
⚠Caution	Before starting work, check the robot, emergency stop switches, related devices, etc. and make sure there are no abnormalities. → Check before the start of work

B. This shows precaution points given in the separate "Safety Manual".
For details, please read the text of the "Safety Manual".

⚠Caution	Use the robot in an environment that is within the range of its specifications. Failure to do this can cause a drop in reliability and breakdown. (Temperature, humidity, atmosphere, noise, etc.)
⚠Caution	When transporting the robot, put it into its specified transport posture. Failure to do this can cause a drop in reliability and breakdown.
⚠Caution	Install the robot on a solid platform. If the robot is in an unstable posture, this can cause positional deviation and vibration.
⚠Caution	As much as possible, wire cables away from noise sources. If cables are brought too close to noise sources, this can cause positional deviation and malfunction.
⚠Caution	Do not apply excess force to a connector or bend a cable excessively. Doing so can cause a contact defect or cut line.
⚠Caution	Set work masses, including hands, so that they do not excess rated load or permitted torque. Exceeding either of these can cause an alarm or breakdown.
⚠Warning	Install hands and tools and hold work securely. Failure to do this can cause objects to fly loose during operation and cause personnel injury or damage.
⚠Warning	Ground the robot and controller reliably. Failure to do this can cause malfunction due to noise or in an extreme case, electrical shock.
⚠Caution	Display the operating state while the robot is operating. Lack of such a display can result in someone coming too close to the robot by mistake or mistaken operation.
⚠Warning	Always secure the priority right for control of the robot before doing any teaching work within the robot's operating range. Failure to do this can allow the robot to start upon instruction from the outside and cause personnel injury or damage.
⚠Caution	Always secure the priority right for control of the robot before doing any teaching work within the robot's operating range. Failure to do this can allow the robot to start upon instruction from the outside and cause personnel injury or damage.
⚠Caution	Make the jog speed as slow as possible and do not take your eyes off the robot. Failure to do this may cause a collision between a work piece and peripheral devices.
⚠Caution	After completing program editing but before starting automatic operation, always check operations with step operation. Failure to do this may cause a collision with a peripheral device due to a programming mistake or the like.
⚠Caution	Set up the safety fence in such a way that, while the equipment is running on automatic, either the safety fence door is locked or if anyone tries to open the door, the robot is stopped. Failure to take these protective measures can cause an accident resulting in injury.
⚠Caution	Never on your own judgment make an alterations or use maintenance parts other than those designated. Doing so can cause breakdown and problems.

⚠Warning

When moving the robot arm from the outside, never stick a hand or finger into an opening. Depending on the posture, the hand or finger could get caught in the equipment.

⚠Caution

Do not switch the robot Off or make an emergency stop of the robot by switching Off the robot controller's main power supply.
If the robot controller's main power supply is switched Off during automatic operation, this can reduce the robot's precision. It could also cause the arm to fall or allow inertia to result in collisions with peripheral device or the like.

⚠Caution

When rewriting a program, parameters, or other internal information within the robot's controller, do not switch Off the robot controller's main power supply.
If the robot controller's main power supply is switched Off during automatic operation or while a program or parameter is being rewritten, there is a danger of the internal information in the robot controller being destroyed.

⚠Warning

For using RH-5AH/10AH/15AH series or RH-6SH/12SH/18SH series.
While pressing the brake releasing switch on the robot arm, beware of the arm which may drop with its own weight.
Dropping of the hand could lead to a collision with the peripheral equipment or catch the hands or fingers.

Revision history

Date	Specifications No.	Details of revisions
2002-12-02	BFP-A8253-*	First print
2004-12-20	BFP-A8253-A	Changed the composition in CD-ROM Changed the installed contents.
2009-03-12	BFP-A8253-B	Correspondence to CRn-700 series controller.
2009-10-1	BFP-A8253-C	Added the screen of setting GOT communication. (However, to use the GOT communication, it is necessary to install the “RT ToolBox2” Ver.1.5 or later in the computer.)
2014-01-17	BFP-A8253-D	Correspondence to Windows 7/8/8.1 and 64bit application.
2015-11-10	BFP-A8248-E	Supports Windows10. Added Visual C# (Visual Studio 2010)　sample program.

Preface

Thank you for purchasing MITSUBISHI ELECTRIC Industrial Robot MELFA.
This instruction manual describes the operating procedure of MelfaRXM.ocx, an ActiveX control that communicates with the robot controller.
To ensure correct operation of the MITSUBISHI ELECTRIC Industrial Robot MELFA series, please be sure to read this manual thoroughly before operating MelfaRXM.ocx. Also, keep this manual in a safe place where it will be readily available to you as a reference.

The Product ID is needed when this "MelfaRXM.ocx" is installed. Prepare the License Certification which ismentioned the product ID in. Product ID is different from standard version and mini version.

This document uses the following general terms and abbreviations

General Term/Abbreviation	Contents
RT ToolBox2	General name for the RT ToolBox2 and RT ToolBox2 mini To distinguish them in explanations, these two are called the "standard edition" and "mini edition".
Universal model QCPU	General term for Mitsubishi PLC CPU modules of Q02U, Q03UD, Q03UDE, Q04UDH, Q04UDEH, Q06UDH, Q06UDEH, Q13UDH, Q13UDEH, Q26UDH and Q26UDEH.
Built-in Ethernet port QCPU	General term for Mitsubishi PLC CPU modules of Q03UDE, Q04UDEH, Q06UDEH, Q13UDEH and Q26UDEH.

The Microsoft® Windows® operating system is a registered trademark of the Microsoft Corp. of the United States in the United States and other countries.
Adobe® and Acrobat® are registered trademarks of Adobe Systems Incorporated.
Borland C++ Builder is product of Borland Software Corporation in the United States.

The system names, product names, etc. in this manual are generally trademarks and registered trademarks of their respective companies. In the main text, the ©, ®, and ™ marks are omitted.

Transfer of all or part of the contents of this document without permission is prohibited.
The contents of this document are subject to change without notice.
Every effort has been made to ensure the accuracy of the contents of this document, but if you should notice any unclear point, mistake, or omission, please notify Mitsubishi Electric.

Copyright (C) 2002-2014 MITSUBISHI ELECTRIC CORPORATION. All rights reserved.

Contents

1 BEFORE USE 1

1.1 Symbols Used in This Manual 1

2 OVERVIEW OF PRODUCT 2

2.1 Functions 2
2.2 Description of Product 2
2.3 Operating Environment 3
2.3.1 Connectable robot controllers 3
2.3.2 Computer system 4
2.3.3 Computer cable 5
2.4 Compatibility with Conventional MelfaRX.ocx 6
2.5 Development Environment 7
2.6 Software Configuration 8

3 SETUP 9

3.1 Installation 9
3.1.1 Installation on Windows 8 / 8.1 /10 13
3.2 USB driver (CR750-D/CRnD-700 series robot controller) installation 16
3.2.1 When Using Windows XP 16
3.2.2 When using Windows Vista/7/8/8.1/10 16
3.3 CRnQ communications USB driver installation 17
3.3.1 When using Windows XP 17
3.3.2 When using Windows Vista 19
3.3.3 When using Windows 7/8/8.1/10 20
3.4 CRnQ Communications USB driver for GOT transparent function / GOT communitation installation 23
3.4.1 When using Windows XP 23
3.4.2 When using Windows Vista/7/8/8.1/10 24
3.5 Making MelfaRXM.ocx Coexist with RT ToolBox/RT ToolBox2 25
3.5.1 Version Update of Communication Server 2 25
3.6 Registering and Deleting MelfaRxM.ocx Using Visual Basic 26
3.6.1 Registration Procedure 26
3.6.2 Deleting Procedure 27
3.7 Precautions for Creating Applications 28

4 PROCEDURE FOR SETTING UP COMMUNICATION SERVER 2 29

4.1 About Communication Server 2 29
4.2 Communication Settings 31
4.2.1 Detailed Settings (for RS-232) 32
4.2.2 Detailed Settings (for TCP/IP) 33
4.2.3 Detailed Setting (for USB) 35
4.2.4 GOT Communications Settings 36
4.2.5 CRnQ Communications Settings 37
4.2.6 Robot Controller Connection Setting List 44
4.3 Information of Connected Robot Controllers 45

5 METHODS 46

5.1 Method List 46
5.2 Detailed Description of Methods 48

5.2.1 Methods Related to Communication Server 2 48
5.2.2 Request for Service/Request to Cancel....51
5.2.3 Data Reception....56
5.2.4 Getting the Status of Robot Controllers 62
5.3 Request IDs Specified by Request for Service Methods 76
5.3.1 Requests Related to Editing 80
5.3.2 Requests Related to Monitoring 96
5.3.3 Requests Related to Maintenance....120
5.3.4 Requests Related to Operation 129
6 EVENTS 136
7 SAMPLE PROGRAM....137
7.1 Overview 137
7.2 Detailed Specifications....138
7.2.1 Request ID Test Window (Object Name: FormMain)....138
7.2.2 Operation Panel Window (Object Name: FormOP)....139
8 APPENDIX....142
8.1 About Task Slots....142
8.2 Precautions for Distribution....144

1 Before Use

This chapter provides explanations of the basic terms in this manual.

1.1 Symbols Used in This Manual

The symbols and expressions as shown in Table 1-1 below are used in this manual.
Please be sure to understand the meaning of each symbol before reading beyond this section.

Table 1-1 : Symbols in This Manual

Symbol	Explanation
Danger	Indicates that incorrect handling is most likely to cause hazardous conditions, resulting in death or severe injury to the operator. Please be sure to observe this to ensure safe handling of the robot.
Warning	Indicates that incorrect handling may cause hazardous conditions, resulting in death or severe injury to the operator. Please be sure to observe this to ensure safe handling of the robot.
Caution	Indicates that incorrect handling may cause hazardous conditions, resulting in injury to the operator, or physical damage alone. Please be sure to observe this to ensure safe handling of the robot.
POINT	Provides important information for handling the robot.

2 Overview of Product

2.1 Functions

MelfaRXM.ocx is an ActiveX control that communicates wit the robot controller. When MelfaRXM.ocx is registered on a Windows system, it enables communications with the robot controller from an application created with Visual Basic or other program in order to collect the status information and to perform operations.

The Product ID is needed when this "MelfaRXM.ocx" is installed. Input the Product ID same as the Product ID of **"RT ToolBox2"**. "MelfaRXM.ocx" can be installed with the Product ID of standard version, but can not be installed with mini version's.

2.2 Description of Product

The configuration of the MelfaRXM.ocx product is as follows:

One CD-ROM (RT ToolBox2)

* The CD-ROM contains the following items:

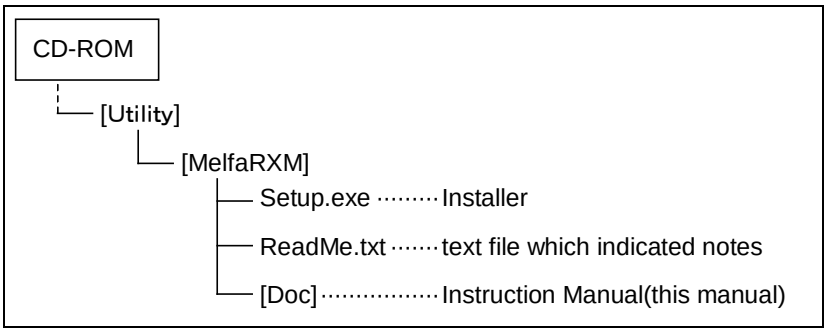


Figure 2-1: Contents of the CD-ROM

2.3 Operating Environment

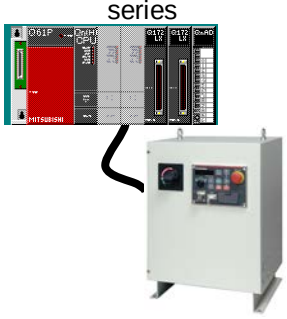
This explains the operating environment.

2.3.1 Connectable robot controllers

This software can be connected with the robot controllers shown below.

> **CAUTION**
> ***RXM.ocx in the CD-ROM of RT ToolBox2 version 1.4 or later can be connected with the CR750-D/CRnD-700 or CR750-Q/CRnQ-700 series controller.***

Table 2-1 : Target robot controllers

Robot controller	Communications (*1)		Remark
CR750-D/CRnD-700 series 	USB (*2)		The commercial item such as computer and cable might be unsuitable to compatibility with our equipment or the FA environments of the temperature and the noise, etc. When it is used, please confirm the operation enough, because you might have to take measures against noise such as EMI measure or addition of ferrite core.
	Ethernet (TCP/IP)		
	RS-232		Only CRnD-700 series
	GOT communications		Communicate with CRnD-700 series controller connected with the GOT via Ethernet by USB / RS-232. GOT 1000 series and the Ethernet communication unit (GT15-J1E71-100) are needed. (*4)
CR750-Q/CRnQ-700 series 	CRnQ communications (*4)	USB (*2)	
		Ethernet (TCP/IP)	The PLC Ethernet interface module or Built-in Ethernet port QCPU (*3) is required.
		RS-232	
CRn-500 series 	Ethernet (TCP/IP)		The robot controller must have the "Ethernet interface" option.
	RS-232		

(*1) The computer must have each ports for communications.
(*2) When using USB connection, 1 computer can connect to only 1 robot controller.
And, it is necessary to install the USB driver when connecting the robot controller by using USB connection.
(*3) Built-in Ethernet port QCPU can be used with **"RT ToolBox2"** Ver. 1.2 or later.
(*4) It is necessary to install **"GX Developer2"** or **"RT ToolBox2"** in the computer when connectiong it with CR750-Q/CRnQ-700 series controller by using the CRnQ communications or GOT communications.
GOT communications can be used with **"RT ToolBox2"** Ver.1.5 or later.

This software can be connected to a maximum of 32 controllers at the same time. These controllers may be different models.

2.3.2 Computer system

Lists the operating environment.

Table 2-2 : Operating Environment

Item	Recommended environment
CPU	Pentium III 1 GHz or higher However, for using the simulation function, Pentinum IV 2 GHz or higher.
Main memory	512 MB min.
Hard disk	Available capacity 300 MB min.
Display	XGA (1024x768) or higher
Optical device	CD-ROM drive
Keyboard	PC/AT compatible keyboard
Pointing device	Must operate on Windows
Communications functions Communications port	- USB2.0 (Caution: This cannot be used for connection with the CRn-500 series controller.) - LAN: 100Base-TX/10Base-T - RS-232 communications port that operates on Windows (Minimum 9600bps: 1 port) Must have one of the above interfaces
OS	Windows XP Professional (32-bit version) Windows XP Home Edition (32-bit version) Windows Vista Ultimate (32-bit version) Windows Vista Business (32-bit version) Windows Vista Home Premium (32-bit version) Windows Vista Home Basic (32-bit version) Windows 7 Starter (32-bit version) Windows 7 Home Premium (32/64-bit version) (*1) Windows 7 Professional(32/64-bit version) (*1) Windows 7 Enterprise (32/64-bit version) (*1) Windows 7 Ultimate (32/64-bit version) (*1) Windows 8 (32/64-bit version) (*1) (*2) Windows 8 Pro (32/64-bit version) (*1) (*2) Windows 8 Enterprise (32/64-bit version) (*1) (*2) Windows 8.1 (32/64-bit version) (*1) (*2) Windows 8.1 Pro (32/64-bit version) (*1) (*2) Windows 8.1 Enterprise (32/64-bit version) (*1) (*2) Windows 10 Home (32/64-bit version) (*1) (*2) Windows 10 Pro (32/64-bit version) (*1) (*2) Windows 10 Education (32/64-bit version) (*1) (*2) Windows 10 Enterprise (32/64-bit version) (*1) (*2) English versions for each edition

*1 : This Windows version is corresponded with the communication middleware in theRT ToolBox2 version 3.01B or later.

*2 : When you install this software, [.NET Framework 3.5 (includes .NET 2.0 and 3.0)] feature should be turned on. Refer to Chapter 3.1.1 for the method of a present state confirmation and turning on.

2.3.3 Computer cable

Prepare the cable for connecting the controller and the computer. The cable required depends on the connection specifications and controller used, as shown below.

For the RS-232 cable, refer to the "Standard Specifications" for your robot.

Table 2-3 : CR750-D/CRnD-700 Series, CRn-500 Series Communication Cables

Method	Description		Model name	Manufacturer
USB	USB A type, USB mini B type		-	-
Ethernet	10BASE-T, 100BASE-TX		-	-
RS-232 (Only CRnD-700 series controller)	For controller front panel	CRnD-700 series	2D-232CBL03M	Mitsubishi Electric
		CRn-500 series	RS-MAXY-CBL RS-AT-RCBL (for expansion serial interface (option))	Mitsubishi Electric
	For expansion option box (CR1-EB3)		RS-AT-RCBL	Mitsubishi Electric

Table 2-4 : CR750-Q/CRnQ-700 Series Communication Cables

Method	Description	Cables confirmed by Mitsubishi Electric to operate properly	
		Model name	Manufacturer
USB	USB A type to mini B type	ZUM-430	Loas Co.
		USB-M53	Elecom Co.
		GT09-C20USB-5P	Mitsubishi Electric System Service
		MR-J3USBCBL3M	Mitsubishi Electric
Ethernet	10BASE-T, 100BASE-TX		
RS-232	For connecting by personal computer - PLC CPU (when Personal computer connector is D-sub, 9-pin)	QC30R2	Mitsubishi Electric

2.4 Compatibility with Conventional MelfaRX.ocx

If you are currently using conventional MelfaRX.ocx, delete MelfaRX.ocx and communication DLLs, and then install new MelfaRXM.ocx, communication DLLs and other required components. The procedure for deleting MelfaRX.ocx and other components are described later.

MelfaRXM.ocx supports the multi-controller function; therefore, a method compatible with the multi-controller function (method described as <Multi-Controller> in Chapter 5, "Methods") has newly been added.

Although the conventional MelfaRX.ocx method may also be used, the method for the robot controller has to be executed for robot controller identifier “1” in this case.

POINT

What is the multi-controller function?
What is the controller identifier?

The "multi-controller" function performs simultaneous communication by connecting multiple robot controllers with one personal computer. Up to 32 robot controllers may be connected.

The "controller identifier" signifies controller numbers (1 through 32) that are set by communication server in order to use the multi-controller function. The controllers are identified by these controller identifiers in methods that support the multi-controller function.

Clear and delete conventional MelfaRX.ocx and communication DLLs from the registry according to the following procedure:

(1) Delete the MelfaRX.ocx control from the project.
Delete the MelfaRX.ocx control from the application project that is using conventional MelfaRX.ocx. For how to delete the MelfaRX.ocx control, see **3.6.2**, **"Deleting Procedure"**.

(2) Clear conventional MelfaRX.ocx and communication DLLs from the registry.
Start the command prompt, move to the System folder (example: cd System), and enter the following to clear conventional MelfaRX.ocx and communication DLLs from the registry:

regsvr32 -u MelfaRX.ocx

The screen shown in Figure 2-2 appears on the screen to indicate that MelfaRX.ocx has been cleared. Do the same for EZSocketRC.dll.



Figure 2-2: Clear Complete Screen (MelfaRX.ocx)

(3) Delete files.

Delete the following three files from the System folder:

- MelfaRX.ocx
- EZSocketRC.dll
- NarcServerApi.dll



POINT

It is not necessary to clear NarcServerApi.dll from the registry.

2.5 Development Environment

The development environment is as follows:

- Microsoft Visual Basic 4.0
- Microsoft Visual Basic 5.0
- Microsoft Visual Basic 6.0
- Microsoft Visual C++ 4.0
- Microsoft Visual C++ 5.0
- Microsoft Visual C++ 6.0
- Microsoft Visual Studio.NET 2003 (VB.NET)
- Microsoft Visual Studio.NET 2005 (VB.NET)
- Microsoft Visual Studio.NET 2008 (VB.NET)
- Microsoft Visual Studio.NET 2010 (VB.NET/C#)
- Borland C++ Builder 5.0

When neither the Microsoft Remote Data Objects and Microsoft Remote Data Control are installed, it is necessary to install it separately. Please see the Web site of Microsoft for details.



CAUTION

Notes when developing by Visual C++

The BSTR type variable passed to MelfaRXM.ocx should open a memory using a SysFreeString( ) function.

2.6 Software Configuration

Figure 2-3 shows the software configuration when communications are performed between MelfaRXM.ocx and robot controllers.

The following describes the communication procedure:

(1) When a request for service method of MelfaRXM.ocx is called from an application, transmission processing is performed in communication server 2(RoboCom.exe) via communication DLLs.

(2) Communication server 2(RoboCom.exe) sends a request to the robot controllers via communication DLLs, and also sends a transmission complete message among processes to MelfaRXM.ocx. When MelfaRXM.ocx receives this message, it queues in that message, and issue a reception complete event to the application (container).

(3) Once this event occurs, the application must read the received content using a get received data method.

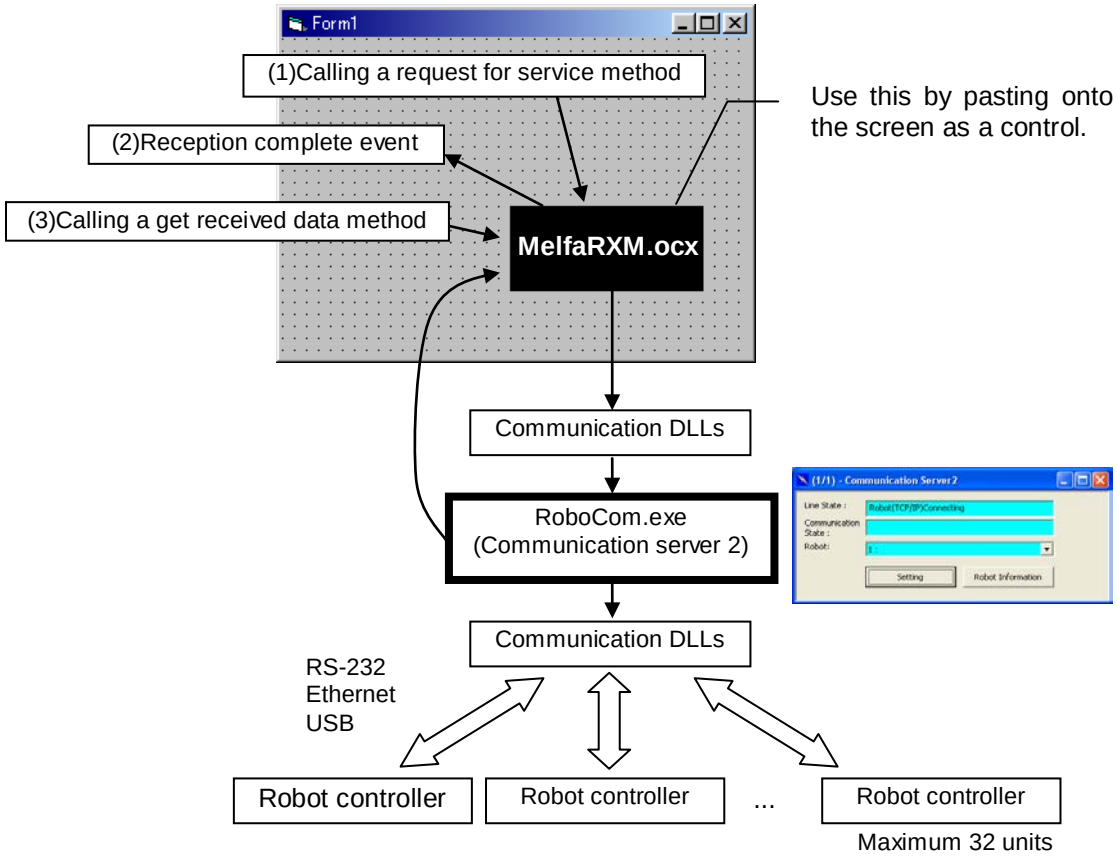


Figure 2-3 : Software Configuration Diagram

POINT

What is the communication server?

The communication server is a tool to operate robot controllers and transmission/reception status data externally.

3 Setup

3.1 Installation

Caution

When installing, log in as a user with administrator authority.

When installing, log in as a user with administrator authority. The system will not let you install if you log in as a user who does not have administrator authority.

Caution

Uninstall the older version before installing.

If an older version of the “Development setting” or the “Files for the distribution" is installed in the computer already, be sure to uninstall the older version before installing the newer version. Especially, please be sure to uninstall them when you use the “Development setting” or the “Files for the distribution" in the CD-ROM of RT ToolBox. Even if the upgrade message appears when you try to overwrite, actually the upgrade would be fail.

Caution

Don't install MELFA-Works (64bit) Version 4.1 or earlier to the same PC.

If MELFA-Works (64bit) Version 4.1 or earlier is installed in the computer already, don't install this product. If this product is installed in the computer already, don’t install MELFA-Works (64bit) Version 4.1 or earlier.

If you uninstall MELFA-Works after both products are installed, uninstall this product once and install this product again.

Perform installation according to the following procedure:

(1) Insert the program CD-ROM into the CD-ROM drive of your personal computer. If Setup of **"RT ToolBox2"** is automatically started, please cancel it.

(2) Select [Run] from the [Start] button.

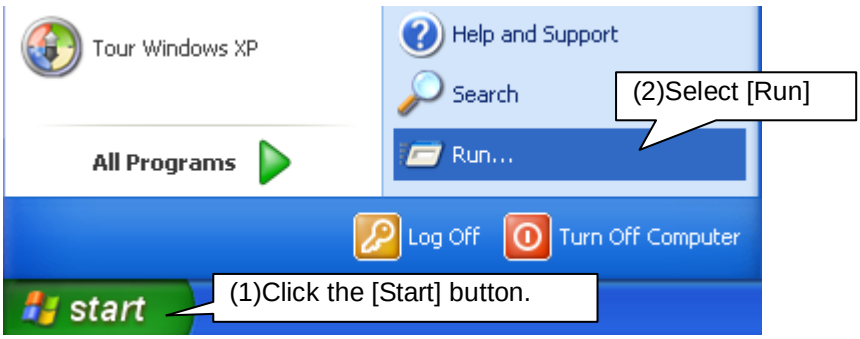


Figure 3-1 : Selecting [Run]

(3) Check the drive name of the CD-ROM drive. Enter the following:

"drive name":/Utility/MelfaRXM/Setup.exe

(If the CD-ROM drive is “D”, enter “D:/Utility/MelfaRXM/Setup.exe”.)

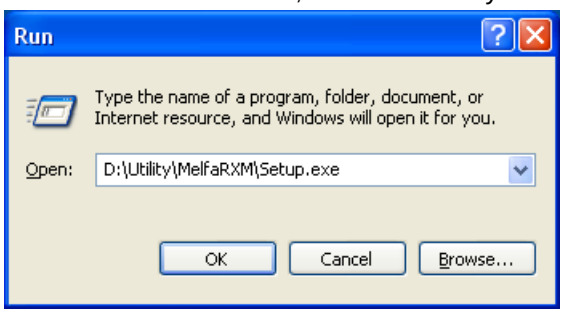


Figure 3-2 : [Run] Screen

Click the [OK] button.

(4) Installer starts and the Setup screen appears. Install according to the instructions that appear on the screen.

The Product ID is needed when this "MelfaRXM.ocx" is installed. Input the Product ID same as the Product ID of **"RT ToolBox2"**. "MelfaRXM.ocx" can be installed with the Product ID of standard version, but can not be installed with mini version's.

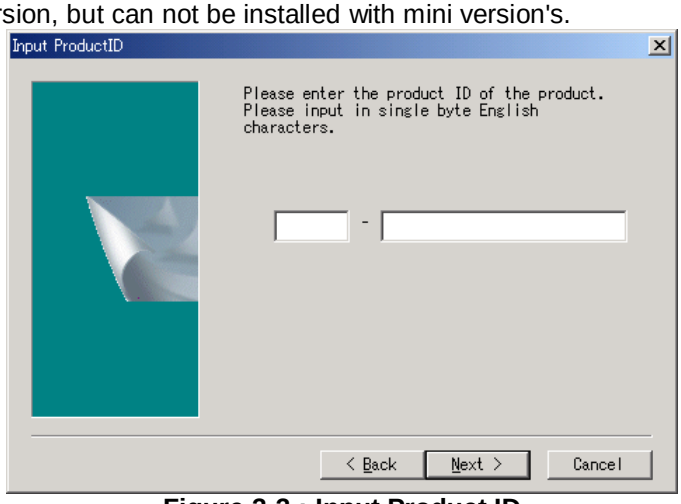


Figure 3-3 : Input Product ID

The files listed in Table 3-1 are installed at the destinations as indicated.

Table 3-1 : Files to be installed

No.	Description	Install destination
1	MelfaRXM.ocx	These files are installed under the system folder. For example: 32bitOS : ¥Windows¥System32 64bitOS : ¥Windows¥SysWOW64
2	communication DLLs	
3	RoboCom.exe (communication server2)	A folder specified during install operation (Normally, C:/ is used.) [MelfaRXM_Dev] ├ ReadMe.txt……text file which indicated notes ├ [RoboCom]……folder of communication server ├ [Doc]……folder of Instruction Manual ├ [Sample]……folder of sample programs │ ├[BCB]……Borland C++ Builder (5.0) │ ├[CS]……Visual C# (Visual Studio 2010) │ ├[VB]……Visual Basic (6.0) │ └[VC++]……Visual C++ (6.0) │ [VB.NET]……Visual Basic .NET 2003 │ (This has confirmed the operation only in a Japanese version.) ├ [Redist] │ └[Installer]……folder of system files Installer (for redistribution)
4	Instruction Manual (this document)	
5	Sample programs	
6	Redistribution files	
7	Information files for USB driver (for CR750-Q/CRnD-700)	

If you are using "Robot Total Engineering Support Software"(hereafter "Support Software"), there are precautions that must be observed in order to allow the coexistence of MelfaRXM.ocx and the "Support Software". Be sure to refer to **3.5**, **"Making MelfaRXM.ocx Coexist with "**.

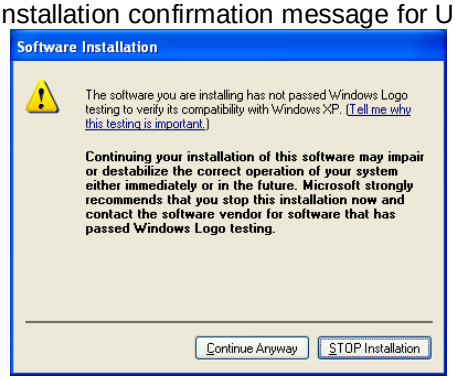
To use an application created with MelfaRXM.ocx on another personal computer, MelfaRXM.ocx, communication DLLs and communication server 2 are required. Please use the installer for distribution to install these files. For more information, see **8.2**, **"Precautions for Distribution"**.

⚠ Caution

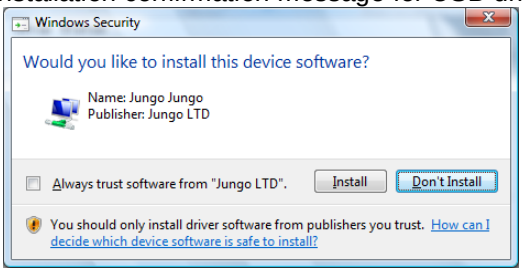
About the confirmation and warning message displayed during installation (The communication middleware in the RT ToolBox2 version 3.00A or earlier)

During installation on Windows XP or Windows Vista/7/8/8.1/10, the following confirmation and warning messages are displayed, but select to continue installation. If you select not to install, please execute the installation again.

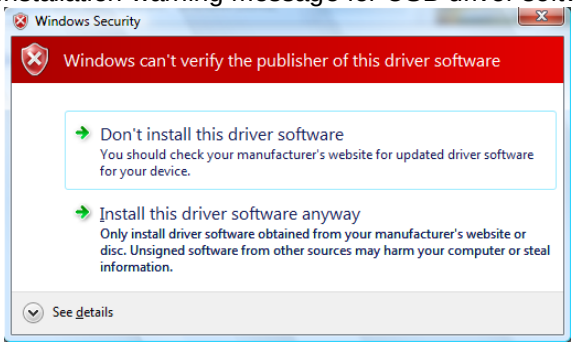
(1) Installation confirmation message for USB driver software (for Windows XP)



(2) Installation confirmation message for USB driver software (for Windows Vista/7/8/8.1/10)



(3) Installation warning message for USB driver software (for Windows Vista/7/8/8.1/10)



<Remark>

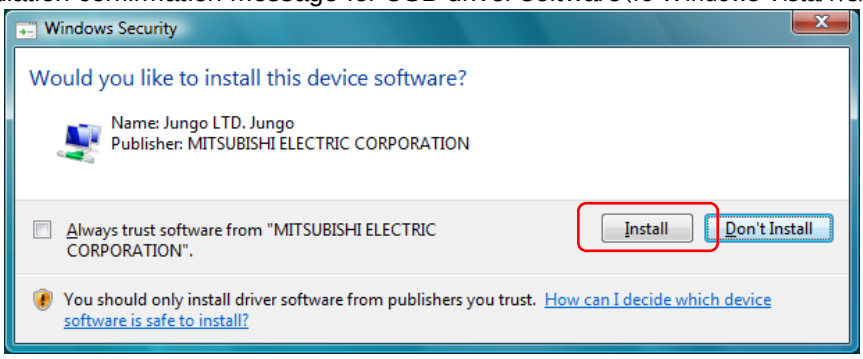
We have confirmed operation at our company. No problem occurs after installation.

⚠ Caution

About the confirmation and warning message displayed during installation (The communication middleware in the RT ToolBox2 version 3.01B or later)

During installation on Windows XP or Windows Vista/7/8/8.1/10, the following confirmation and warning messages are displayed, but select to continue installation. If you select not to install, please execute the installation again.

Installation confirmation message for USB driver software（ro Windows Vista/7/8/8.1/10）



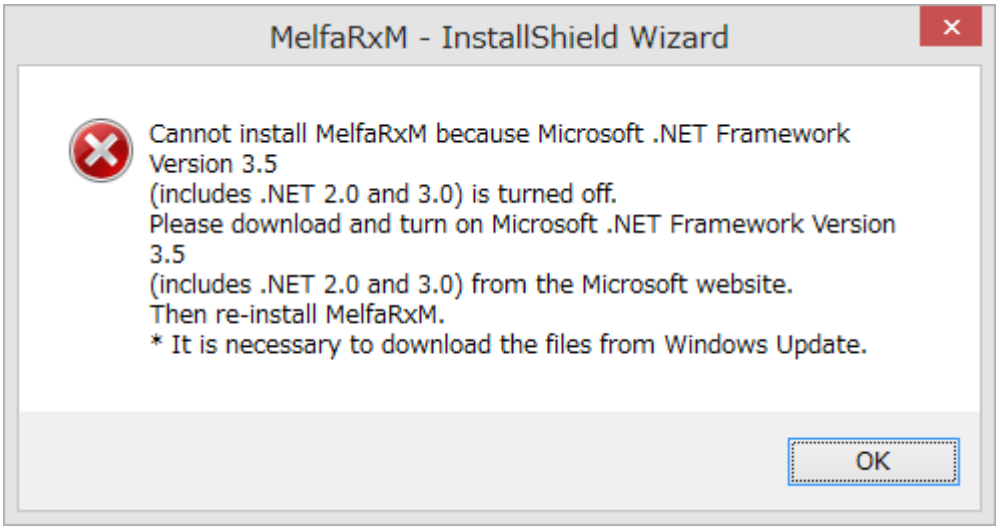
When “Always trust software from MITSUBISHI ELECTRIC CORPORATION” is checked, the alert message will not be displayed after next time.

<Remark>

We have confirmed operation at our company. No problem occurs after installation.

3.1.1 Installation on Windows 8 / 8.1 /10

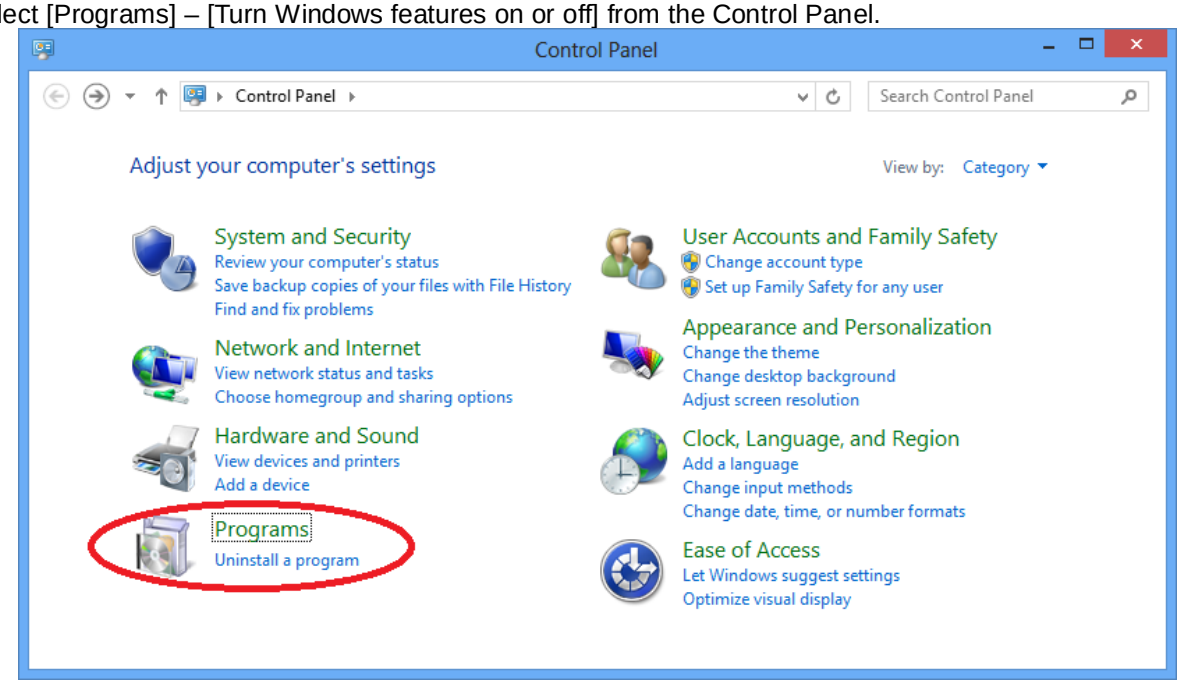
When you install this software on Windows 8/8.1/10, you will need to turn on ".NET Framework Version 3.5 (includes .NET 2.0 and 3.0)” feature of the PC you want to install. If you install this software during turn off the feature, the installation will stop and show the following dialog.

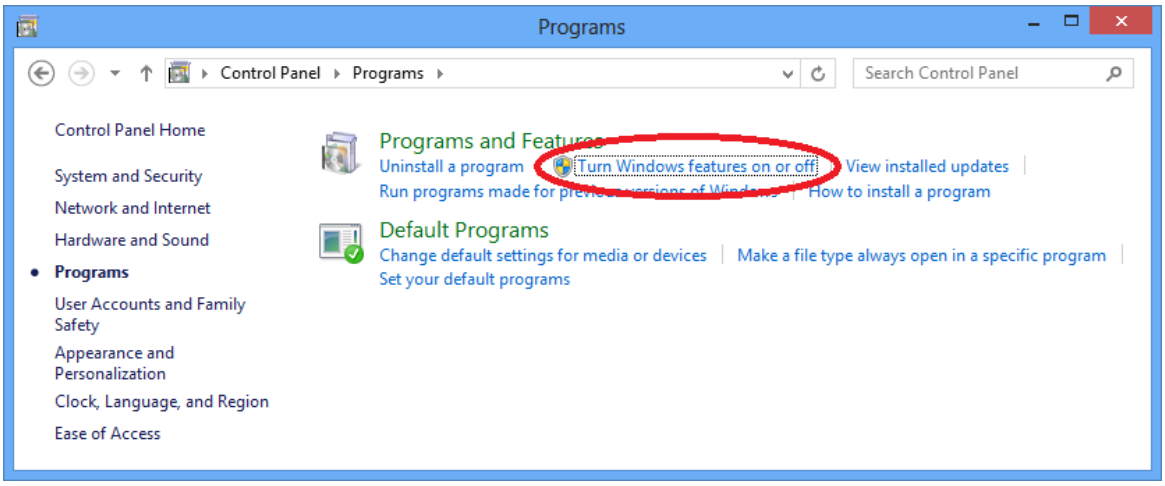


If the dialog is shown, install this software again after you turn on ".NET Framework Version 3.5 (includes .NET 2.0 and 3.0)" feature as following steps.

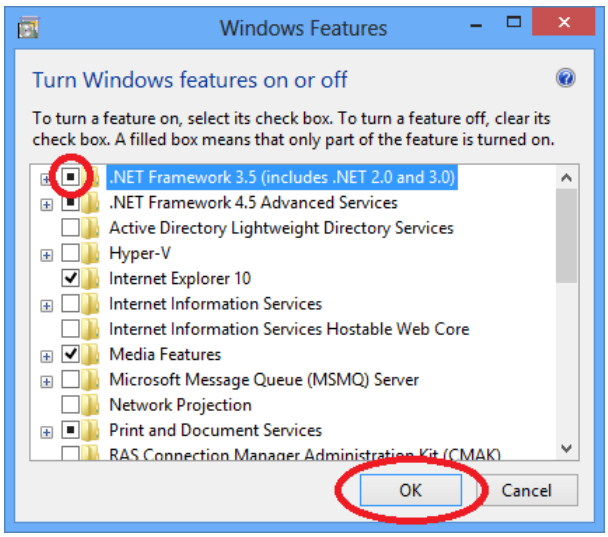
(1) Be ready to connect the PC to the Internet.

(2) Select [Programs] – [Turn Windows features on or off] from the Control Panel.

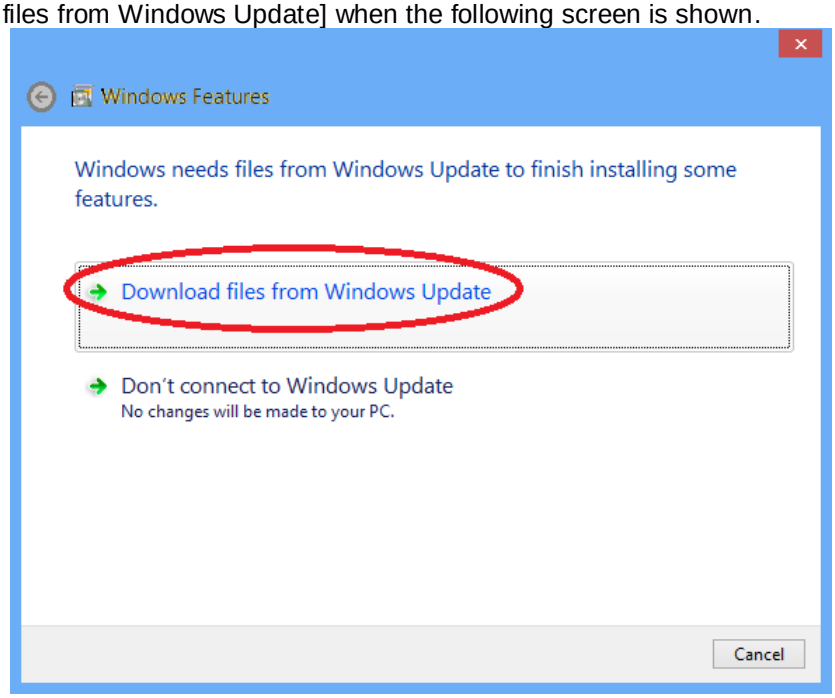




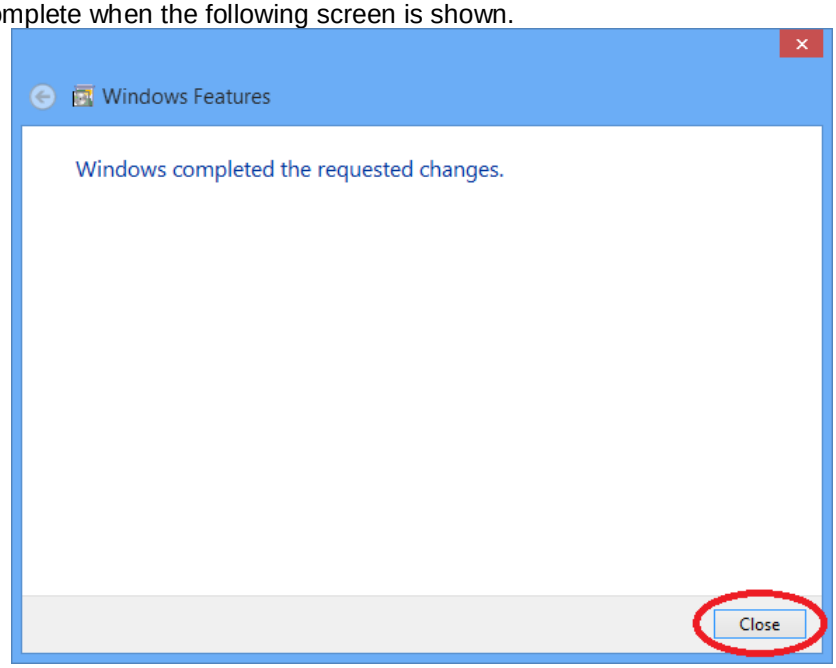
(3) Check [.NET Framework 3.5 (includes .NET 2.0 and 3.0)] check box on Turn Windows features on or off screen and click [OK].



(4) Select [Download files from Windows Update] when the following screen is shown.



(5) This process is complete when the following screen is shown.



3.2 USB driver (CR750-D/CRnD-700 series robot controller) installation

Connecting the CRnD-700 series robot controller with USB requires installation of the robot USB driver. Install with the following procedure.

Caution

If the USB driver cannot be installed, check the following setting.

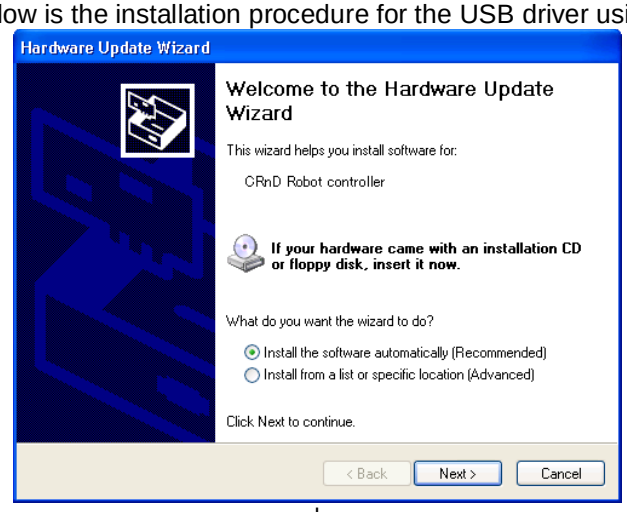
<When Windows XP is used>

If you have selected "Block-Never install unsigned driver software" after [Control Panel] - [System] - [Hardware] - [Driver Signing], the USB driver may not be installed.

Choose "Ignore-Install the software anyway and don't ask for my approval" or "Warn-Prompt me each time to choose an action" for [Driver Signing], and install the USB driver.

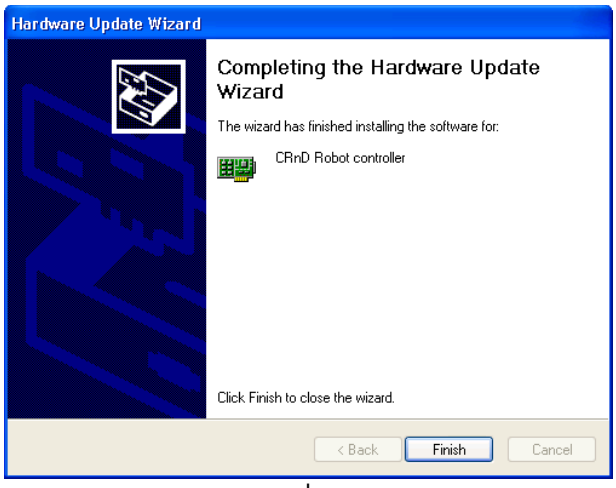
3.2.1 When Using Windows XP

Below is the installation procedure for the USB driver using Windows XP (Professional).



1) When you connect the computer and CR750-D/CRnD-700 series robot controller with a USB cable, the screen on the left is displayed. Select "Install the software automatically (Recommended)", then click the [Next] button. Installation of the USB driver starts.

↓



2) When the screen on the left is displayed, the installation is complete. Click the [Finish] button to end the installation.

↓

(Completed)

3.2.2 When using Windows Vista/7/8/8.1/10

When you connect the CR750-D/CRnD-700 robot controller and the computer with a USB cable, installation starts and completes automatically.

3.3CRnQ communications USB driver installation

Connecting the CR750-Q/CRnQ-700 series robot controller with USB requires installation of the robot USB driver. Install with the following procedure.

Caution

Please install RT ToolBox2 when you use CRnQ communications.

When the software is connected to CR750-Q/CRnQ-700 series controller with the USB cable, it is necessary to install the CRnQ communicaions USB driver. When **RT ToolBox2** is installed, the USB driver for the CRnQ communication is installed.

Caution

If the USB driver cannot be installed,
check the following setting.

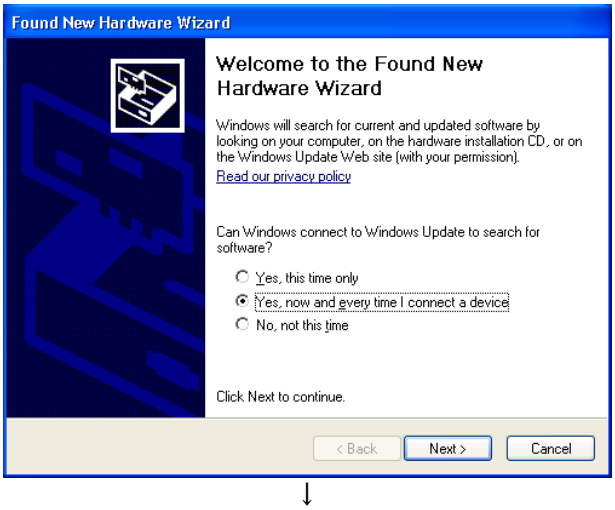
<When Windows XP is used>

If you have selected "Block-Never install unsigned driver software" after [Control Panel] - [System] - [Hardware] - [Driver Signing], the USB driver may not be installed.

Choose "Ignore-Install the software anyway and don't ask for my approval" or "Warn-Prompt me each time to choose an action" for [Driver Signing], and install the USB driver.

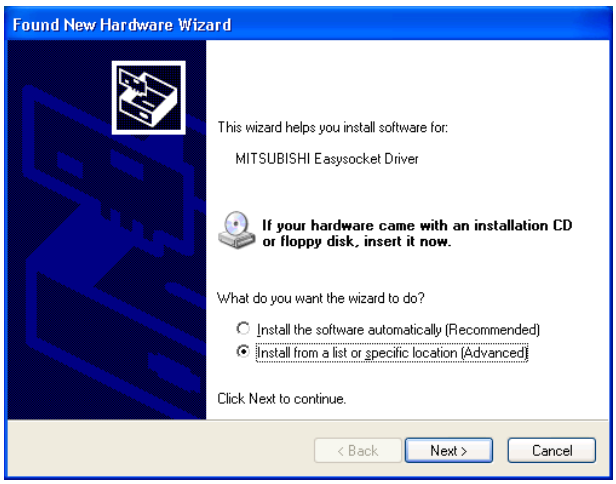
3.3.1 When using Windows XP

The following indicates the procedure for installing the USB driver when using Windows XP (Professional).



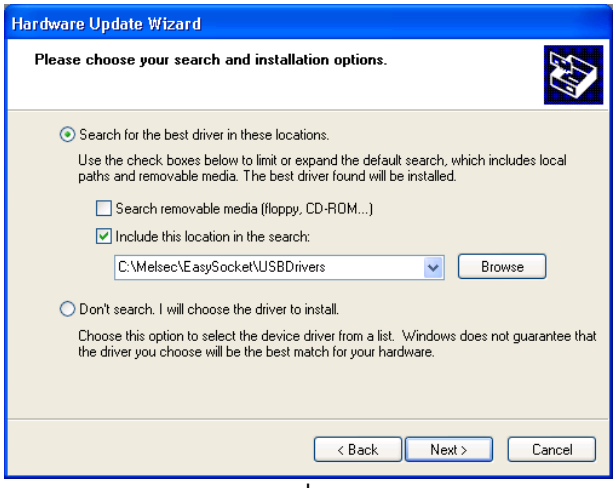
↓

1) The screen shown on the left appears when you connect the personal computer and Universal model QCPU by the USB cable.
Choose "Yes, now and every time I connect a device" and click the [Next] button.



2) As the screen on the left appears, choose "Install from a list or specific location [Advanced]" and click the [Next] button.

↓



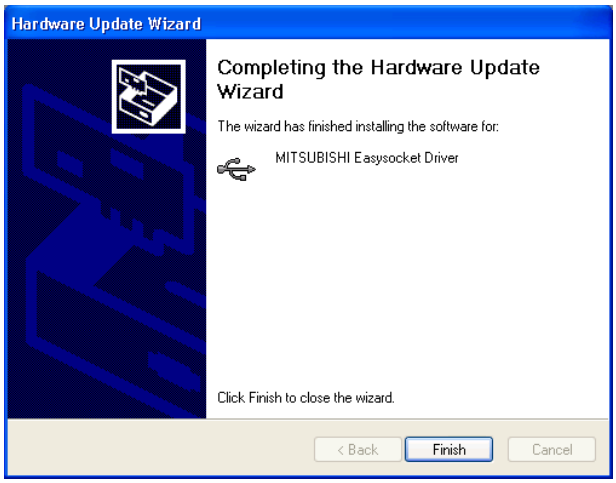
3) As the screen on the left appears, choose "Search for the best driver in these locations".
Check "Include this location in the search" and set the “C:¥Melsec¥EasySocket¥USBDrivers”. After setting, click the [Next] button.
If volume MELSOFT products have been installed, browse the installation destination "EasySocket¥USBDrivers" of the first product.

↓



4) As the screen on the left appears, click the [Continue Anyway] button to continue the installation of the USB driver.
(No problem will occur after installation of the USB driver.)

↓



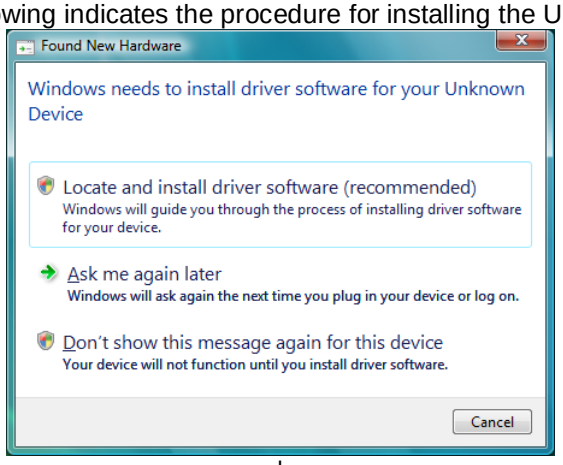
5) The screen on the left appears to indicate completion of installation.
Click the [Finish] button to terminate installation.

↓

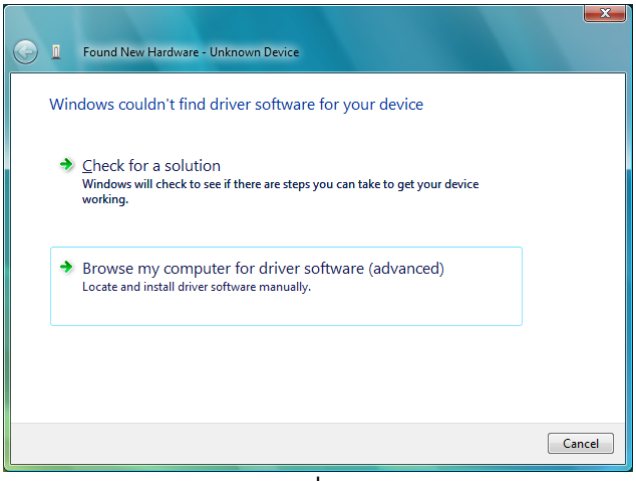
(Completed)

3.3.2 When using Windows Vista

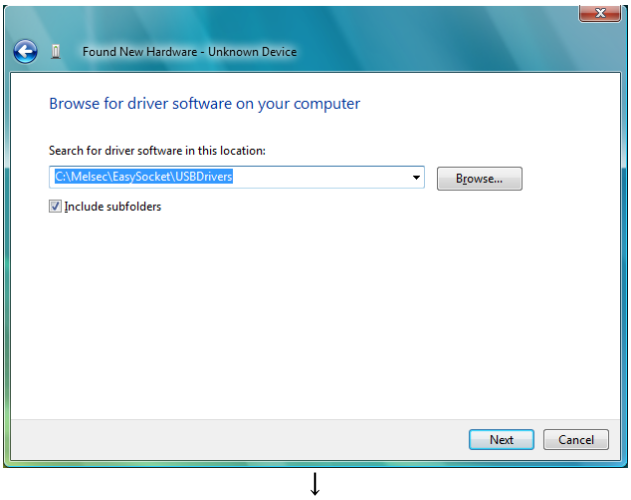
The following indicates the procedure for installing the USB driver when using Windows Vista (Business).



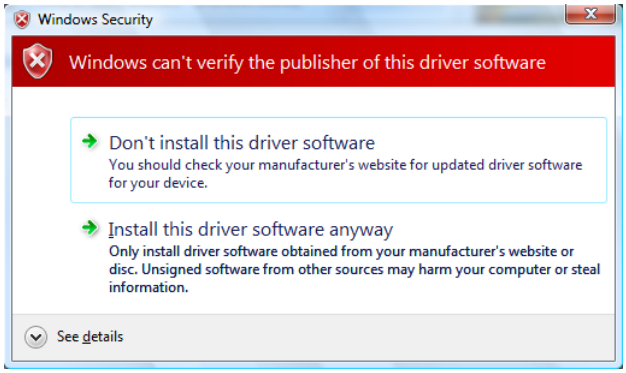
1) The screen shown on the left appears when you connect the personal computer and Universal model QCPU by the USB cable. Select "Locate and install driver software (recommended)" and wait for the search to end.



2) The screen on the left is displayed, so select "Browse my computer for driver software (advanced)".

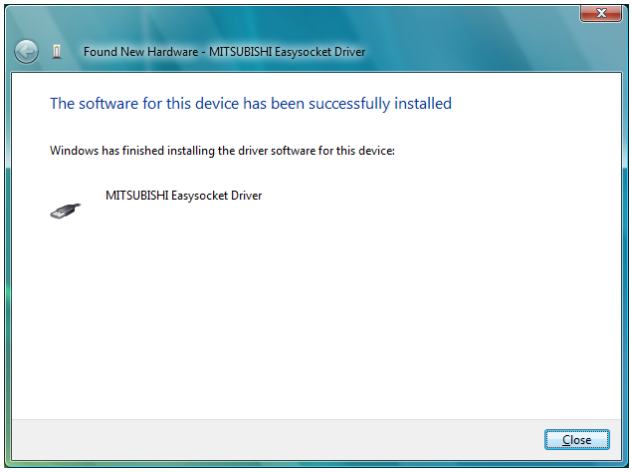


3) The screen on the left is displayed, so select "C:¥Melsec¥EasySocket¥USBDrivers". After making the setting, click the [Next] button. When multiple MELSOFT products are installed, set the default product installation folder "EasySocket¥USBdrivers".



4) The screen on the left is displayed, so select "Install this driver software anyway".

↓



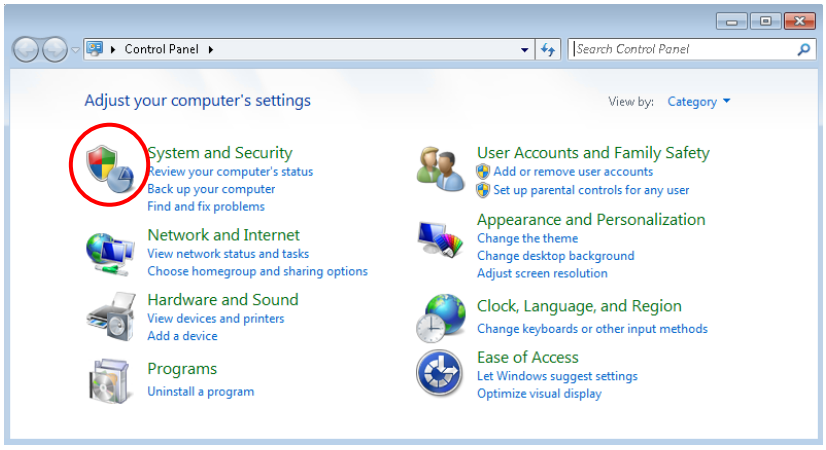
5) The screen on the left is displayed. Click the "Close" button. This completes the installation.

↓

(Completed)

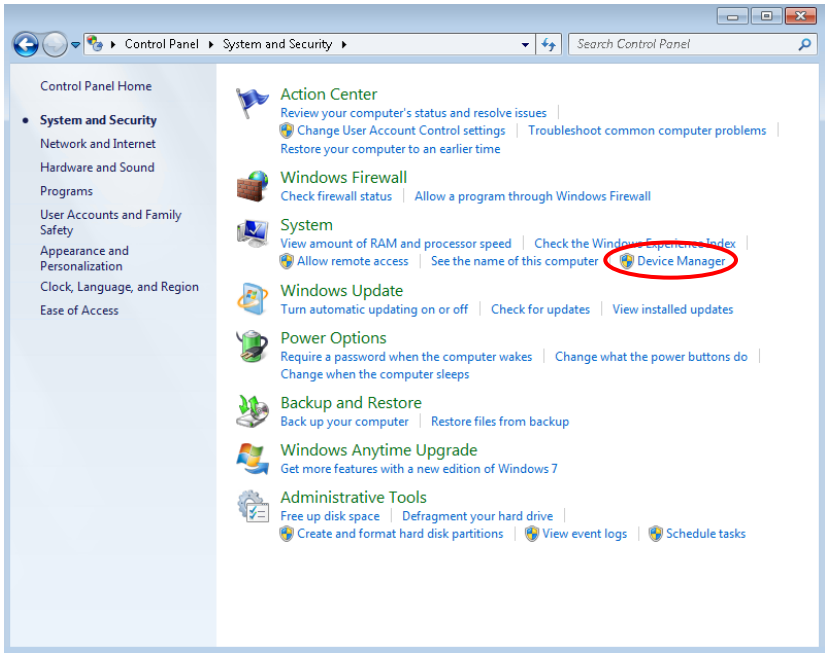
3.3.3 When using Windows 7/8/8.1/10

The following indicates the procedure for installing the USB driver when using Windows 7/8/8.1/10.

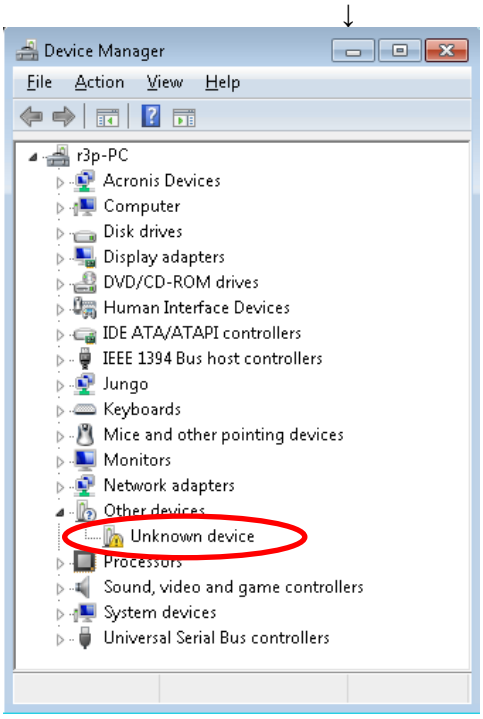


1) When "Control panel" is selected from the start menu with the USB cable connected, the following screens are displayed.
The screen on the left is displayed, clicking the red circle part in this screen.

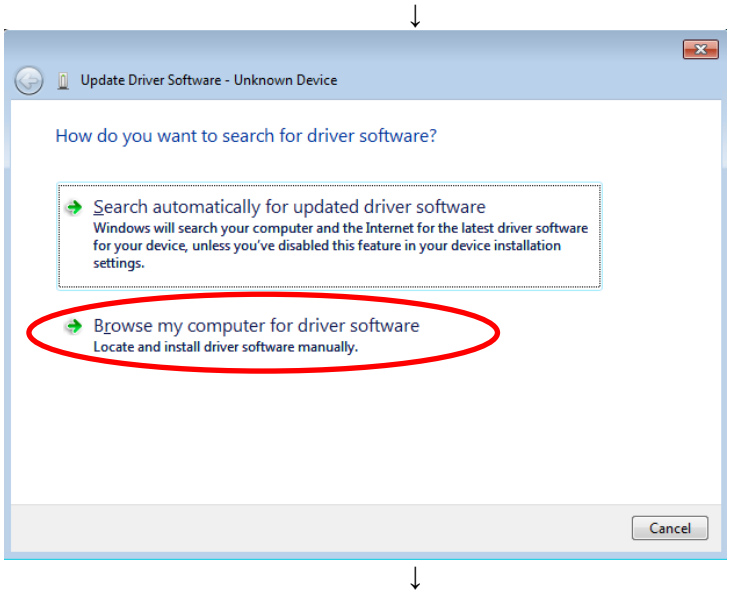
↓



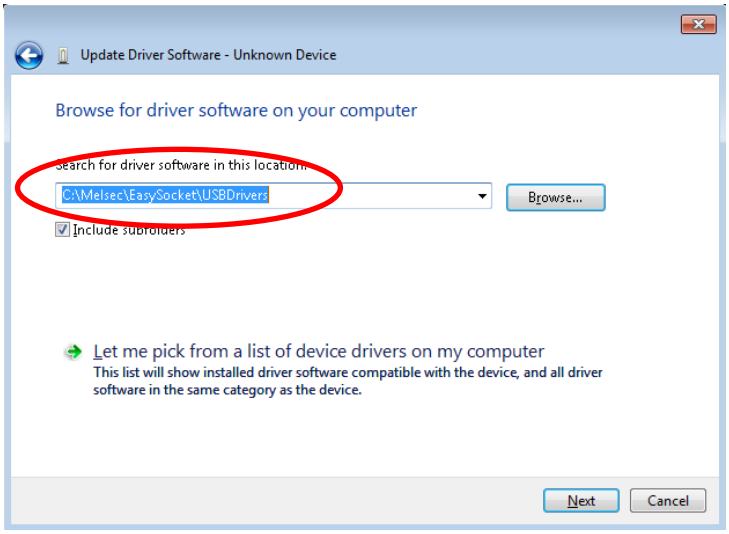
2) The screen on the left is displayed, clicking the red circle part in this screen.



3) The screen on the left is displayed, right-clicks on “Unknown device” is displayed on this screen.
Select "Update Driver Software" of the menu that right-clicks on this item and is displayed.

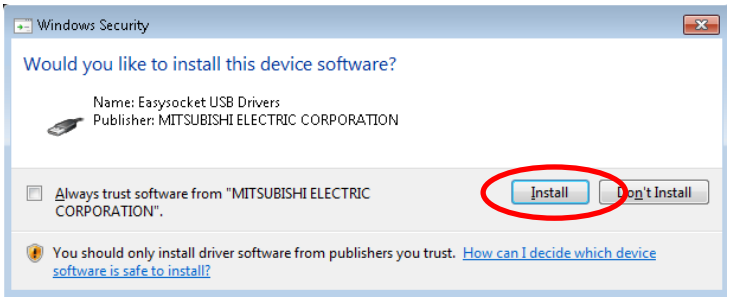


4) The screen on the left is displayed, click lower "Browse my computer for driver software".



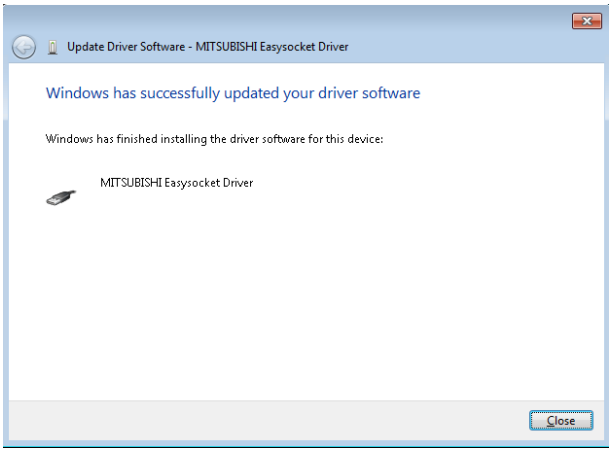
↓

5) The screen on the left is displayed, set "C:¥Melsec¥Easysocket ¥USBdrivers".
If volume MELSOFT products have been installed, browse the installation destination "EasySocket\USBDrivers" of the first installed product.
After setting, click the "Next" button.



↓

6) The screen on the left is displayed, click "install".



↓

7) The screen on the left appears to indicate completion of installation.
Click the [Finish] button.

↓

(Completed)

3.4 CRnQ Communications USB driver for GOT transparent function / GOT communitation installation

Connecting the CR750-Q/CRnQ-700 series robot controller with USB via GOT transparent mode and connecting the CR750-D/CRnD-700 series robot controller with USB via GOT communication requires installation of the USB driver for the GOT transparent function. Install with the following procedure.
GOT communication is available from **"RT ToolBox2"** Ver.1.5 or later.

Caution

Please install RT ToolBox2 when you use CRnQ communications.

When the software is connected to CR750-Q/CRnQ-700 series controller with the USB cable, it is necessary to install the CRnQ communicaions USB driver. When **RT ToolBox2** is installed, the USB driver for the CRnQ communication is installed.

Caution

If the USB driver cannot be installed, check the following setting.

<When Windows XP is used>
If you have selected "Block-Never install unsigned driver software" after [Control Panel] - [System] - [Hardware] - [Driver Signing], the USB driver may not be installed.
Choose "Ignore-Install the software anyway and don't ask for my approval" or "Warn-Prompt me each time to choose an action" for [Driver Signing], and install the USB driver.

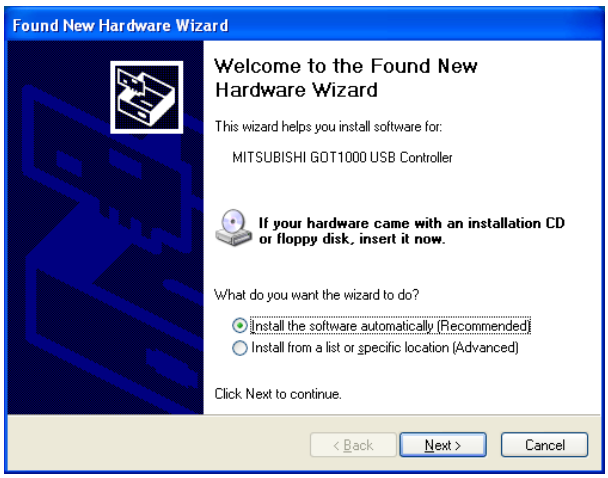
3.4.1 When using Windows XP

The following indicates the procedure for installing the USB driver when using Windows (Professional)



↓

1) The screen shown on the left appears when you connect the personal computer and GOT by the USB cable.
Choose "Yes, now and every time I connect a device" and click the [Next] button.



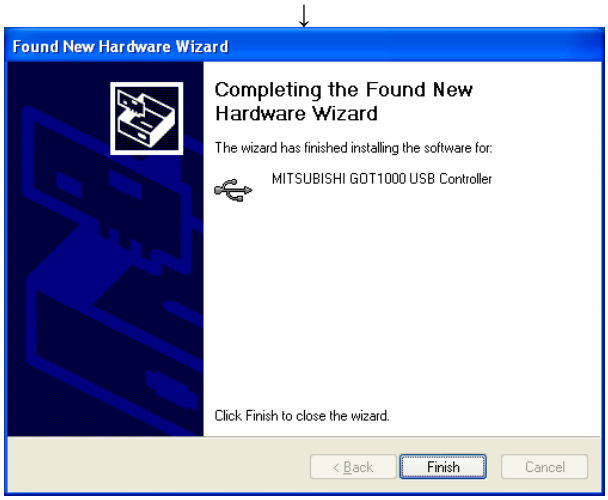
2) As the screen on the left appears, select "Install the software automatically (Recommended)", then click the [Next] button.

↓



3) As the screen on the left appears, click the [Continue Anyway] button to continue the installation of the USB driver.
(No problem will occur after installation of the USB driver.)

↓



4) The screen on the left appears to indicate completion of installation. Click the [Finish] button to terminate installation.

↓

(Completed)

3.4.2 When using Windows Vista/7/8/8.1/10

When you connect the GOT and the computer by the USB cable, installation starts and completes automatically.

3.5 Making MelfaRXM.ocx Coexist with RT ToolBox/RT ToolBox2

MelfaRXM.ocx uses the same communication server 2 and communication DLLs as those of **"RT ToolBox"** or **"RT ToolBox2"**.

To operate an application that uses MelfaRXM.ocx and **"RT ToolBox"** or **"RT ToolBox2"** together on a single personal computer, be sure to fully understand the precautions below and perform necessary operations.

3.5.1 Version Update of Communication Server 2

If the version of installed communication server 2 is different from the version of communication server 2 of **"RT ToolBox"** or **"RT ToolBox2"**, it is necessary to update communication server 2 to the latest version. Copy the executable file (RoboCom.exe) of communication server 2 in the new install folder to the old install folder by overwriting.

Table 3-2 lists the folders that normally need to be specified during installation.

Table 3-2 : Locations of Install Folders

Application	Version	Location of install folder
Personal computer support software	version E1 or earlier	C:/Program Files/Melfa
Robot Total Engineering Support Software RT ToolBox	version F1 or later	C:/Program Files/MELSOFT/RT ToolBox E
RT ToolBox2	Version 1.8 or ealier	C:/Program Files/MELSOFT/RT ToolBox2 E
	Version 2.00A or later	C:/Program Files/MELSOFT/RT ToolBox2
MelfaRXM		C:/MelfaRXM_Dev/RoboCom

3.6 Registering and Deleting MelfaRxM.ocx Using Visual Basic

3.6.1 Registration Procedure

The following describes the procedure for registering the MelfaRxM.ocx control using Visual Basic 6.0.

(1) Load the MelfaRxM.ocx control into a Visual Basic project.
Select [Components] from the [Project] menu of Visual Basic. In the Controls tab, check [MelfaRxM ActiveX Control module], and then press the [OK] button.

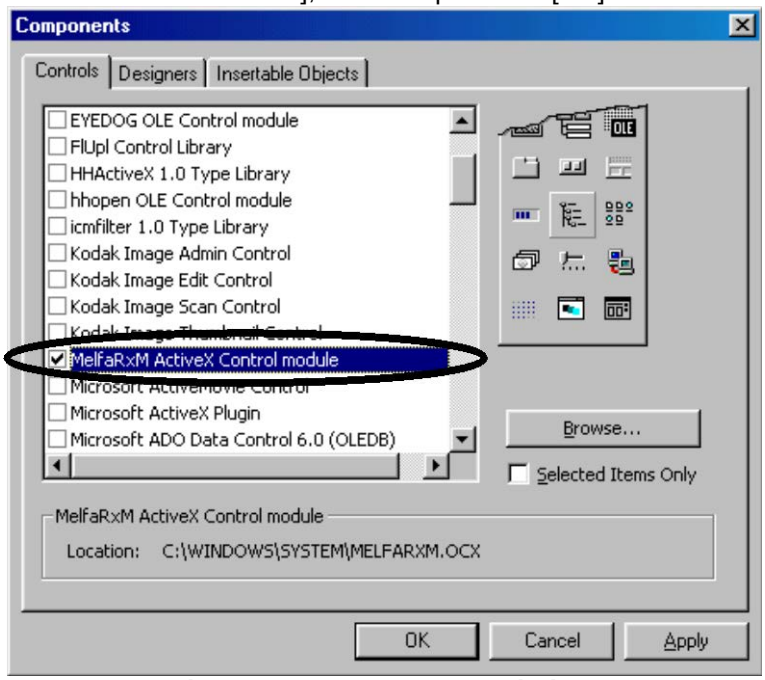


Figure 3-4 : Components Window

(2) Paste MelfaRxM.ocx control onto a form.
Paste MelfaRxM.osx control onto a form from the tool box.
It can then be used just like other controls.

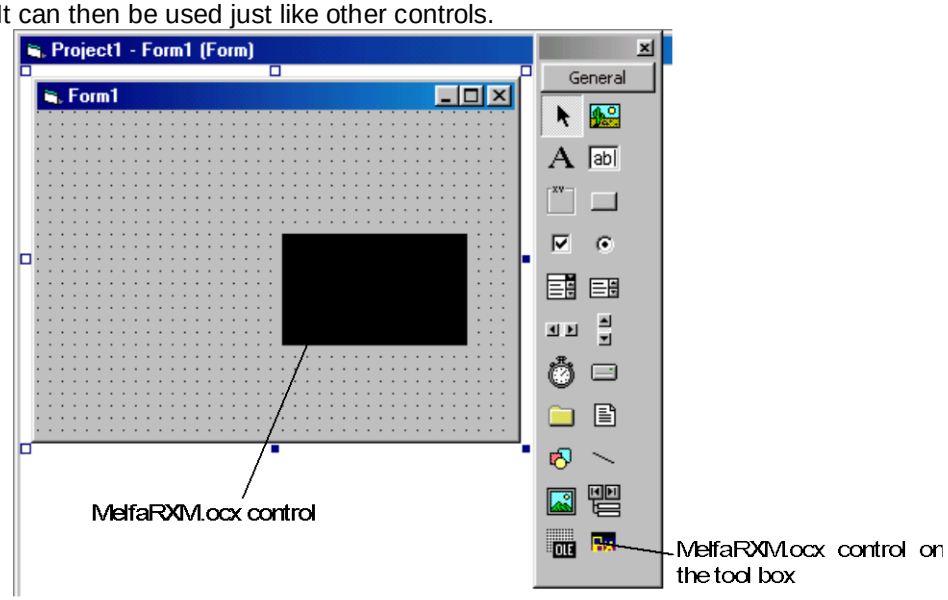


Figure 3-5 : Example of Pasting MelfaRxM.ocx Control onto a Form

If communication server 2 has been started, the MelfaRXM.ocx control displays the connection status (ONLINE/OFFLINE) between communication server 2 and robot controllers.



Figure 3-6 : Display of Connection Status

For more information about the general use of OCX, refer to its reference documents.

3.6.2 Deleting Procedure

The following describes the procedure for deleting the MelfaRxM.ocx control using Visual Basic 6.0.

(1) Delete the MelfaRxM.ocx control from a form.
Delete MelfaRxM.ocx control that has been pasted onto a form.

(2) Delete the MelfaRxM.ocx control from a Visual Basic project.
Select [Components] from the [Project] menu of Visual Basic. In the Controls tab, uncheck [MelfaRxM ActiveX Control module], and then press the [OK] button.

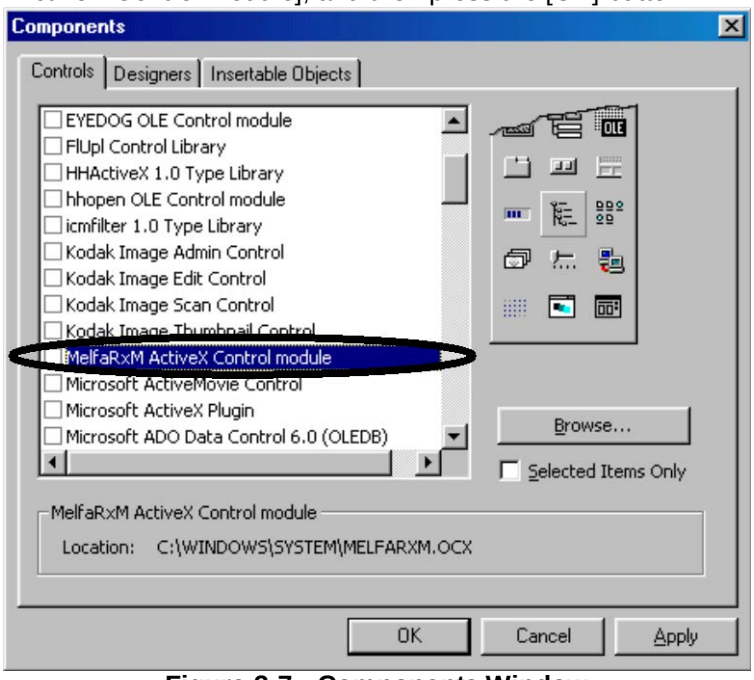


Figure 3-7 : Components Window

3.7 Precautions for Creating Applications

To start communication server 2 from an application you created, RoboCom.exe (communication server 2) must reside in a work folder of the application. Copy RoboCom.exe (communication server 2) in the folder installed (normally, C:/MelfaRXM_Dev/RoboCom) to a work folder of the application you created before using the application.

Once one communication server 2 has been started, communications can be performed from any application on your personal computer. Note that two or more communication servers cannot be started simultaneously. The settings of communication server 2 are saved in a folder of communication server 2 that has been started.

4 Procedure for Setting Up Communication Server 2

Communication server 2 is a tool to operate robot controllers and transmission/reception status data externally. When communication server 2 receives a transmission request from MelfaRXM.ocx, it expands the request, and communicates with the robot controllers via a communication DLL. Therefore, it is necessary to set up the information of the robot controllers to be communicated.

POINT

It is necessary to start communication server in order to communicate with robot controllers.

To communicate with robot controllers, it is necessary to start communication server. Start communication server using the start communication server (ServerStart) method.

4.1 About Communication Server 2

POINT

If the communication settings are changed, communication may no longer be performed.

If the communication settings are changed, communication server may not be able to communicate with robot controllers. Therefore, be careful when changing the communication settings.

When communication server 2 is started for the first time, it is set to the RS-232 communication setting.
If the communication settings on the robot controller side have been changed or to use an Ethernet(TCP/IP) and USB, it is therefore necessary to change the communication settings of communication server 2 according to the following procedure.

Because communication server 2 is represented by an icon at startup, in order to perform communication settings, restore the window to the original size by clicking [Communication Server 2] on the taskbar.

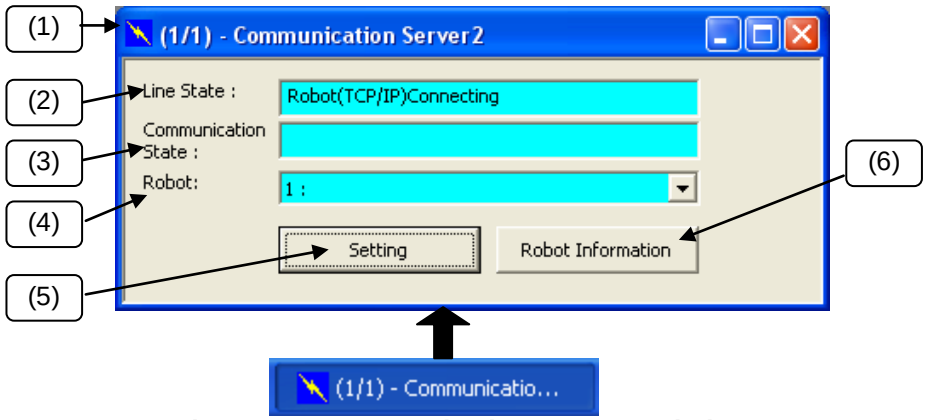


Figure 4-1 : Communication Server 2 Window

1) [Title bar] "(1/1)" indicates (the number of robot controllers connected / the number of robot controllers set up for communication).

2) [Line State] Shows the connection status of the communication line with the robot. The status of the robot controller currently selected is indicated by color and display.

Table 4-1 : Line States

Display	Meaning	Color
Connecting	Indicates that connection with the robot is being established.	Light blue or Blue
Connection wait	Indicates that the robot controller is communicating to verify connection when connecting via RS-232. Indicates that the robot controller is waiting for the connection of the communication port when connecting via TCP/IP and USB.	Green
Connection error	Displayed when the data reception enable signal cannot be detected due to cable disconnection or because the robot has not been started for connection via RS-232. Displayed when the communication port cannot be opened for connection via TCP/IP and USB. In the case of USB connection, if the USB driver has not been installed, it is also displayed in red.	Red
Communication Setting error	Displayed when the COM port cannot be opened for connection via RS-232. This is not displayed for connection via TCP/IP.	Red

3) [Communication State] Displays the status of communication with a robot controller.

4) [Robot] Specify a robot controller of which you want to display the line state and the communication state.
Only the robot controllers set up for communication are displayed as selections.

5) [Setting] Changes the communication settings.

6) [Robot Information] You can reference the information about the robots currently being connected.

This communication server in the **"RT ToolBox2"** Ver.1.5 or later can communicate with robot controller by higher reliability communication. In this case, line state color (connecting) of communication server 2 is "Blue".

Table 4-2 : Combination of higher reliability communication

		Robot controller software				
		CRn-500	CR750-D/ CRnD-700		CR750-Q/ CRnQ-700	
		All versions	Ver.P7a or former	Ver.P8 or later	Ver.N7a or former	Ver.N8 or later
RT ToolBox2	Ver.1.4.1 or former	○	○	○	○	○
	Ver.1.5 or later	○	○	◎	○	◎

○ : Conventional communication (line state color (connecting) : light bule)

◎ : Higher reliability communication (line state color (connecting) : blue)

4.2 Communication Settings

When [Setting] is clicked, windows as shown in **Figure 4-2** appear.

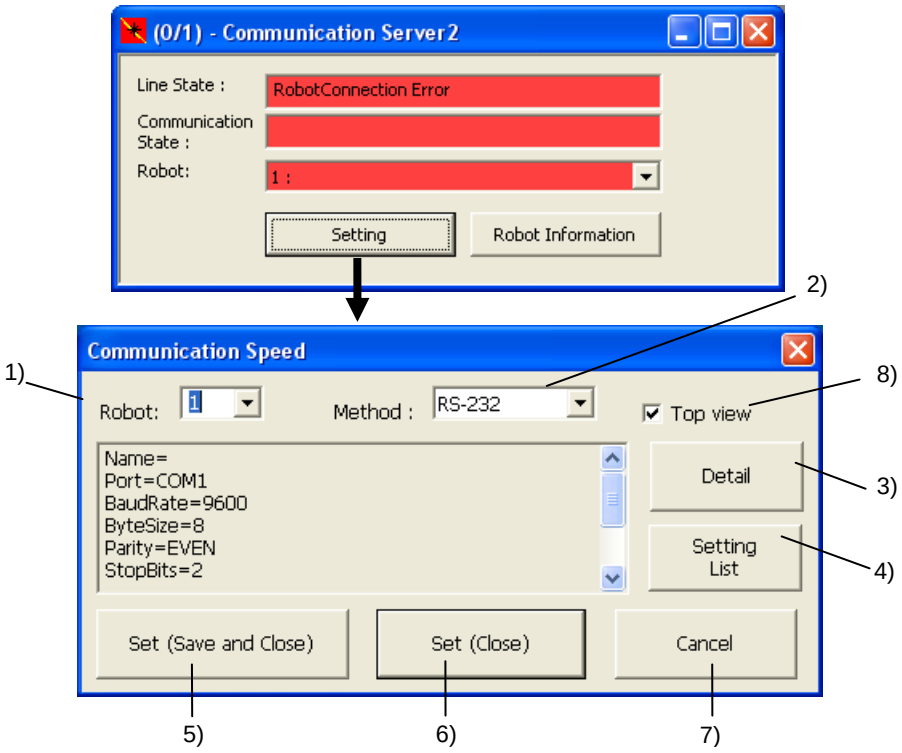


Figure 4-2 : Communication Settings Windows

1) [Robot] Specify robot controllers (1 to 32) to set up communication for.

2) [Method] Select the method to use for communication.
You can select from [RS-232], [TCP/IP], [USB], [CRnQ], [GOT] and [Not used].
Set [Not used] for robot controllers that are not connected. Otherwise, the performance of your personal computer may degrade. Note that robot controller 1 cannot be set as [Not used].

3) [Detail] Displays the window used to change the communication settings. The window to be displayed varies depending on whether RS-232, TCP/IP, USB, CRnQ or GOT is selected in [Method].

4) [Setting List] This makes it possible to perform communication settings in a table format. This is useful when performing communication setting for multiple robot controllers at once.

5) [Set (Save and Close)] Changes the communication settings and saves the changes. When communication is started next time, these settings are used for communication.

6) [Set (Close)] Changes the communication settings. Because the changes are not saved, the previous settings are used for communication when communication is started next time.

7) [Cancel] Ends communication settings without changing the communication settings.

8) [Top view] If this check box is checked, a window indicating the communication status is displayed in the foreground. If unchecked, that window is hidden.

⚠ CAUTION

Do not shut down Communication Server 2 manually.

When the setting of "Communication Server 2" is completed, click the button to change Communication Server 2 to an icon.

Communication Server 2 automatically shuts down when the functions requiring the communication of this tool are complete, so do not click the button to shut down communication server.

Once Communication Server 2 is shut down, communicate with robot controllers cannot be continued.

4.2.1 Detailed Settings (for RS-232)

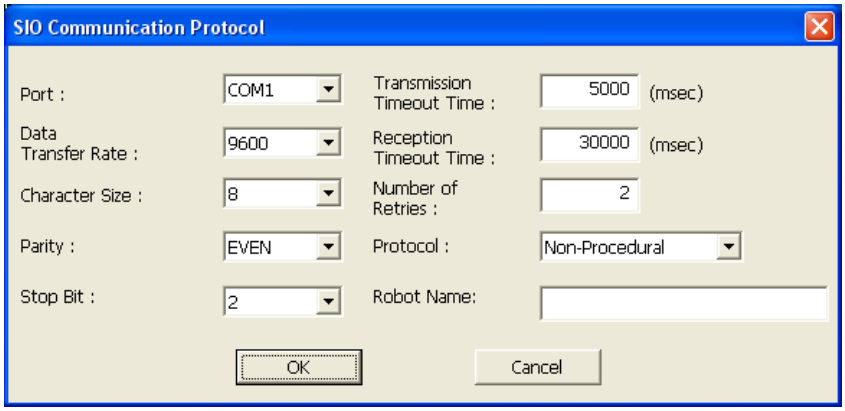


Figure 4-3 : SIO Communication Protocol Window (RS-232)

Table 4-3 : RS-232 Communication Setting Items

Item	Explanation
Port used	COM1 – COM10 can be selected.
Communications speed	Select from 4800, 9600, and 19200. * It is possible to select Baud rate 38400 only when connecting this software with CRnD-700 controller.
Character size	7 or 8 can be selected, but select 8.
Parity	None, Odd, or Even can be selected.
Stop bit	Select from 1, 1.5, and 2.
Send timeout	Timeout time for sending Range that can be set (*1) Minimum value : 1000 msec Maximum value : 30000 msec
Receive timeout	Timeout time for receiving Range that can be set (*1) Minimum value : 5000 msec Maximum value : 120000 msec
Retries	Number of communication retries Range that can be set (*1) Minimum value : 0 Maximum value : 10
Usage protocol	Non-Procedural or Procedural can be selected.
Robot Name	Enter up to 32 single-byte characters as a nickname of a robot controller. It is not a required item, but because the name entered here is displayed during robot name selection, it is useful if multiple robot controllers are to be connected.

(*1) The setting range of Send timeout, Receive timeout and Retries are limited function of communication server in the **"RT ToolBox2"** Ver.1.2 or later.

Set the following to perform stable communication at high speed.

[Data Transfer Rate] : 19200 bps
[Protocol] : Procedural

It is also necessary to change the communication settings of the robot controller.
Table 4-4 lists the parameters that need to be changed. For more information, refer to the instruction manual of the robot controller.

Table 4-4 : Communication Setting Parameters of the Robot Controller

No.	Parameter name	Setting
1	COMDEV	Method, Port
2	CBAU232	Baud rate
3	CPRTY232	Parity
4	CSTOP232	Stop Bit
5	CPRC232	Protocol

Caution

Use of USB to RS-232 does not guarantee normal operation.

When you use USB to RS-232 cable, normal operation is not guaranteed.
If you want to use RS-232 as a communication port, use of computer with serial ports is recommended.

4.2.2 Detailed Settings (for TCP/IP)

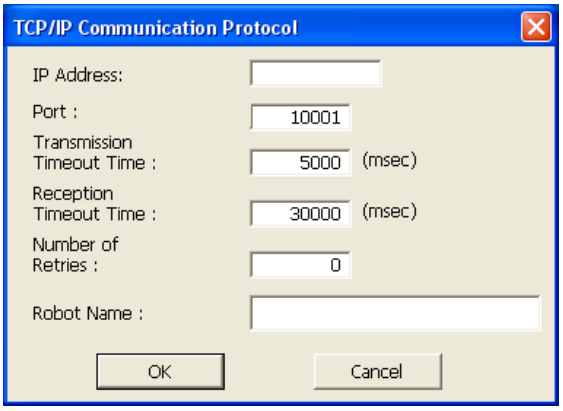


Figure 4-4 : TCP/IP Communication Protocol Window

Table 4-5 : TCP/IP Communication Setting Items

Item	Explanation
IP address	Sets the IP address of the robot controller connected to.
Port used	Number of the port used for communications
Send timeout	Timeout time for sending Range that can be set (*1) Minimum value : 1000 msec Maximum value : 30000 msec
Receive timeout	Timeout time for receiving Range that can be set (*1) Minimum value : 5000 msec Maximum value : 120000 msec
Retries	Number of communication retries Range that can be set (*1) Minimum value : 0

	Maximum value : 10
Robot Name	Enter up to 32 single-byte characters as a nickname of a robot controller. It is not a required item, but because the name entered here is displayed during robot name selection, it is useful if multiple robot controllers are to be connected.

(*1) The setting range of Send timeout, Receive timeout and Retries are limited function of communication server in the **"RT ToolBox2"** Ver.1.2 or later.

For help with the computer side network settings (IP address, subnet mask, default gateway, etc.), ask the network administrator.

Moreover, for details on the communication settings on the robot controller side, please refer as follow.

CR750/CRn-700 series controllers : See **"Operations and Detailed Explanation"** and **"Standard Specifications"** in the user's manual for the robot controller.

CRn-500 series controllers : See the **"Ethernet Interface Option User's Manual"**.

⚠ Caution

When connecting to 10 or more robot controllers at the same time.

When connecting to 10 or more robot controllers at the same time, set the reception timeout time to at least 10000 msec, because it takes time to connected processing.

⚠ Caution

Caution for connecting with robot controller by Ethernet.

In the program edit function, please change "Receive timeout" and "Retries" of TCP/IP communication setting when it corresponds to all of the following use conditions, and the program is written to robot controller.

<< Use conditions >>

1. It connects with CRn-500/CR750-Q/CRnQ-700/CR750-D/CRnD-700 controller by using Ethernet. (* In CRn-500 controller, Ethernet is an option.)
2. When you write only command line in the controller.
   (Only "Command lines" is selected when the program is written to robot controller.)
3. The number of lines of the command line of the robot program is 1600 or more.
4. In TCP/IP Communication Settings, the both value of "Receive timeout" and "Retries" are numbers except 0.

⇩

item	Value
Receive timeout (msec)	30000 or more
Retries	0

When using it on the condition of corresponding without changing the communication setting, the robot program might not be correctly written or the error might occur while writing the robot program. Therefore, it is necessary to check all content of the robot program. So please change the communication setting.

4.2.3 Detailed Setting (for USB)

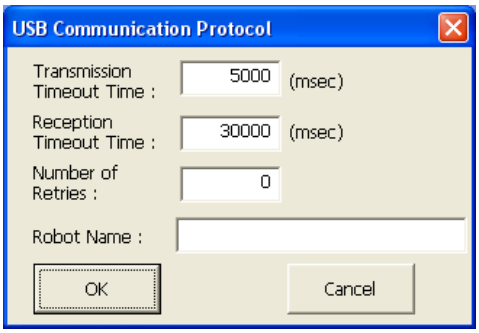


Figure 4-5 USB Communication Settings

Table 4-6 : USB Communication Setting Items

Item	Explanation
Send timeout	Timeout time for sending Range that can be set (*1) Minimum value : 1000 msec Maximum value : 30000 msec
Receive timeout	Timeout time for receiving Range that can be set (*1) Minimum value : 5000 msec Maximum value : 120000 msec
Retries	Number of communication retries Range that can be set (*1) Minimum value : 0 Maximum value : 10
Robot Name	Enter up to 32 single-byte characters as a nickname of a robot controller. It is not a required item, but because the name entered here is displayed during robot name selection, it is useful if multiple robot controllers are to be connected.

(*1) The setting range of Send timeout, Receive timeout and Retries are limited function of communication server in the **“RT ToolBox2”** Ver.1.2 or later.

When the USB cable is connected to the computer, the screen for installing the USB driver may be displayed. For details on USB driver installation, see **"3.2 USB driver (CR750-D/CRnD-700 series robot controller)** installation**"**.

4.2.4 GOT Communications Settings

The GOT Communications is the method to connect with the CR750-D/CRnD-700 series controller by using the RS-232 / USB on the GOT which is connected with the controller via Ethernet.
GOT Communication can be used with communication server in the **"RT ToolBox2"** Ver.1.5 or later.

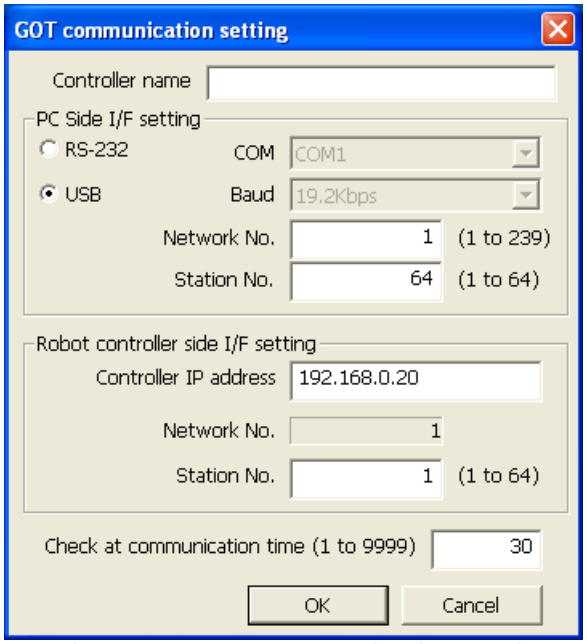


Figure 4-6 GOT Communication Settings

Table 4-7 : GOT Communication Setting Items

Item	Explanation
COM (*1)	COM1 - COM10 can be selected.
Baud (*1)	Select from 9.6Kbps、19.2Kbps、38.4Kbps、57.6Kbps、115.2Kbps.
Network No.	1 - 239 can be set. Be matched to Network No. of the GOT.
Station No. (PC side I/F setting)	1 - 64 can be set. Be matched to Station No. of the GOT. Set a value different from the Station No. of Robot controller side I/F setting.
Controller IP address	Set the IP address of the controller connected with GOT.
Station No. (Robot controller side I/F setting)	1 - 64 can be set. Be matched to Station No. set by GOT.
Check at communication time	1 - 9999 can be set. Set the timeout after transmits. Note: If too short time is set, the timeout error occurs.
Controller name	Enter up to 32 single-byte characters as a nickname of a robot controller. It is not a required item, but because the name entered here is displayed during robot name selection, it is useful if multiple robot controllers are to be connected.

(*1) COM and Baud are available only that RS-232 is selected.

4.2.5 CRnQ Communications Settings

The CRnQ communication is the method to connect with the CR750-Q/CRnQ-700 series controller by using the PLC Universal model QCPU module or the PLC Ethernet Interface module.

When using RS-232 or USB, please connect to connector of the PLC Universal model QCPU module.

When using Ethernet, please connect to connector of the PLC Ethernet Interface module or Built-in Ethernet port QCPU. (*Built-in Ethernet port QCPU can be used with communication server in the **"RT ToolBox2"** version 1.2 or later.)

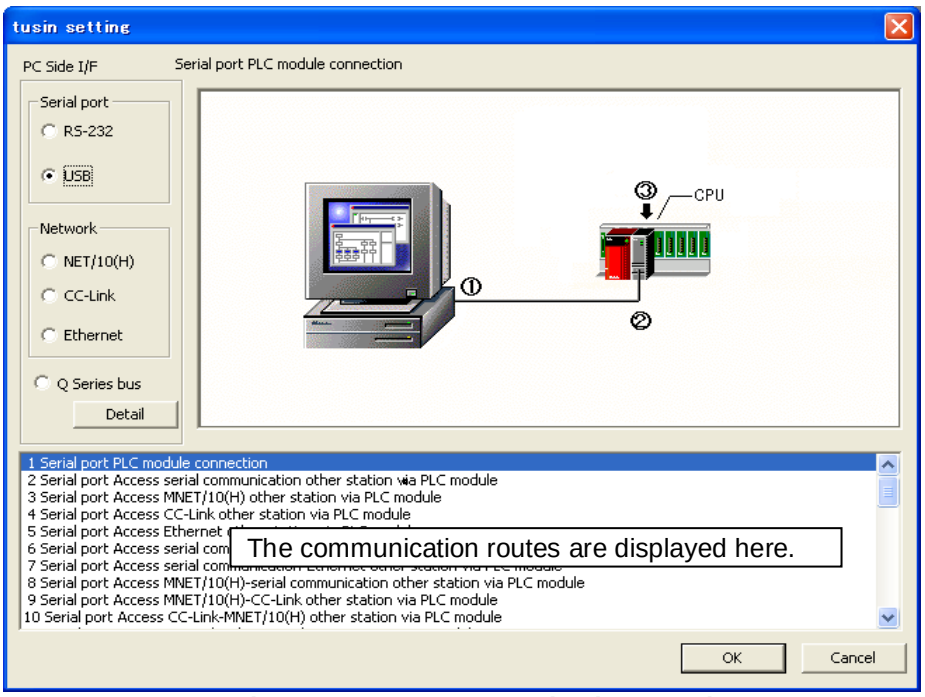


Figure 4-7 RnQ Communications Settings

On the CRnQ communications setting screen, when you select the computer interface, the communications routes using the selected interface are displayed at the bottom of the screen. Select the communications route, then press the [Detailed Settings] button. The detailed settings screen corresponding to the selected communications route is displayed. These detailed settings will be explained from the next item.

4.2.5.1 When RS-232 is selected

When "RS-232" is selected as the computer interface, the communications routes that can be selected are as follows.

Table 4-8 Communications Routes That Can Be Selected for "RS-232"

No.	Communications routes
1	Serial communications CPU unit connection
2	Serial communications Serial communications other station access through CPU unit
3	Serial communication NET/-0(H) other station access through CPU unit
4	Serial communication CC-Link other station access through CPU unit
5	Serial communication Ethernet other station access through CPU unit
6	Serial communication Serial communication through CPU unit - NET/10(H) other station access
7	Serial communication Serial communication through CPU unit-Ethernet other station access
8	Serial communication NET/10(H) through CPU unit - serial communication other station access
9	Serial communication NET/10(H) through CPU unit - CCLink other station access
10	Serial communication Other station access CCLink - NET/10(H) through CPU unit
11	Serial communication CCLink through CPU unit - Ethernet other station access
12	Serial communication Ethernet through CPU unit - serial communication other station access
13	Serial communication Ethernet through CPU unit - CCLink other station access

No.	Communications routes
14	Serial communication Serial communication connection
15	Serial communication Other station access serial communication through serial communication
16	Serial communication Other station access NET/10(H) through serial communication
17	Serial communication Other station access CC-Link through serial communication
18	Serial communication Other station access Ethernet communication through serial communication
19	Serial communication Serial communication through serial communication - NET/10(H) other station access
20	Serial communication Serial communication through serial communication - Ethernet other station access
21	Serial communication NET/10(H) through serial communication - serial communication other station access
22	Serial communication NET/10(H) through serial communication - CCLink other station access
23	Serial communication CCLink through serial communication - NET/10(H) other station access
24	Serial communication CCLink through serial communication - Ethernet other station access
25	Serial communication Ethernet through serial communication - serial communication other station access
26	Serial communication Ethernet through serial communication - CCLink other station access
27	Serial communication CC-Link other station access through G4
28	Serial communication CCLink through G4 - NET/10(H) other station access
29	Serial communication CC-Link through G4 – Ethernet other station access
30	Serial communication Serial communication other station access through NET/10(H) remote
31	Serial communication NET/10(H) other station access through NET/10(H) remote
32	Serial communication CC-Link other station access through NET/10(H) remote
33	Serial communication Ethernet other station access through NET/10(H) remote
34	Serial communication Serial communication other station access through NET/10(H) remote
35	Serial communication Serial communication through NET/10(H) remote - Ethernet other station access
36	Serial communication NET/10(H) through NET/10(H) remote - Serial communication other station access
37	Serial communication NET/10(H) through NET/10(H) remote - CC-Link other station access
38	Serial communication CC-Link through NET/10(H) remote - NET/10(H) other station access
39	Serial communication CC-Link through NET/10(H) remote - Ethernet other station access
40	Serial communication Ethernet through NET/10(H) remote - Serial communication other station access
41	Serial communication Ethernet through NET/10(H) remote - CC-Link other station access

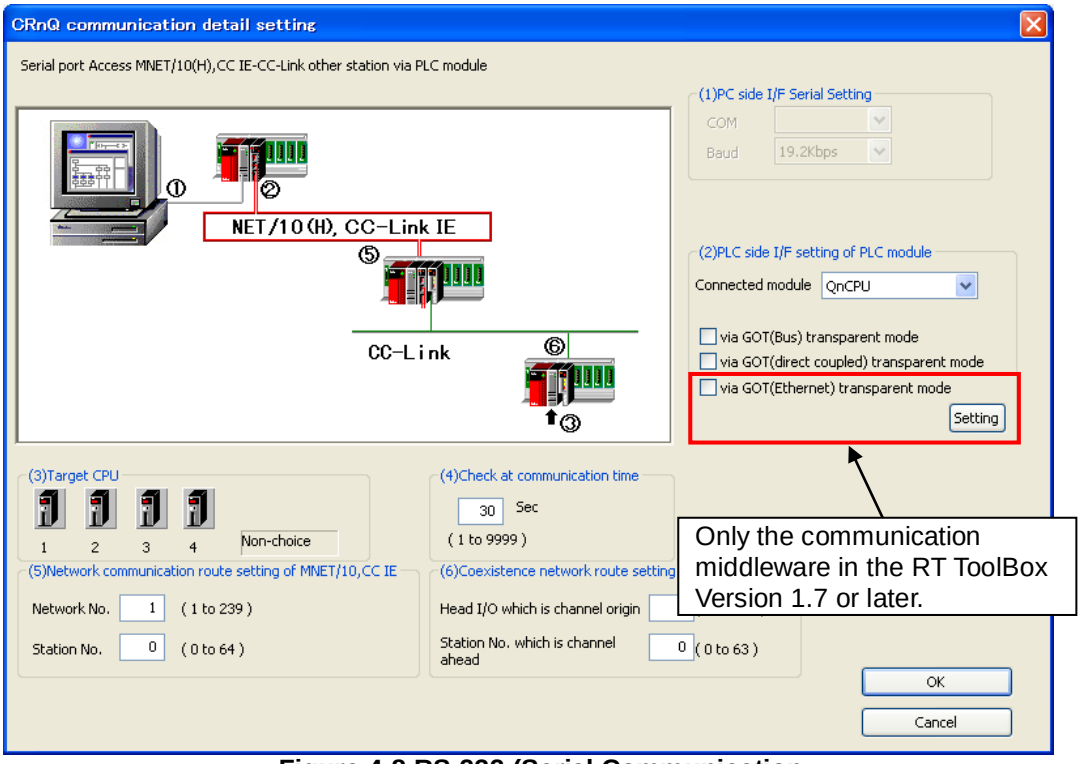


Figure 4-8 RS-232 (Serial Communication NET/10(H) through CPU Unit – CCLink Other Station Access) Detailed Settings

For some communications routes that you can select, "(5)Network communication route setting of MNET/10" and "(6)Coexistence network route setting of C24" can not be set.

GOT(Ethernet) transparent mode can be used with the communication middleware in the RT ToolBox Version 1.7 or later of software. When you use GOT(Ethernet) transparent mode, please display a set screen clicking the “Setting” button, and set the communication item.

4.2.5.2 When USB is selected

When "USB" is selected as the computer interface, the communications routes that you can select are the same as for RS-232. See **"**Table 4-8 Communications Routes That Can Be Selected for "RS-232"**".**

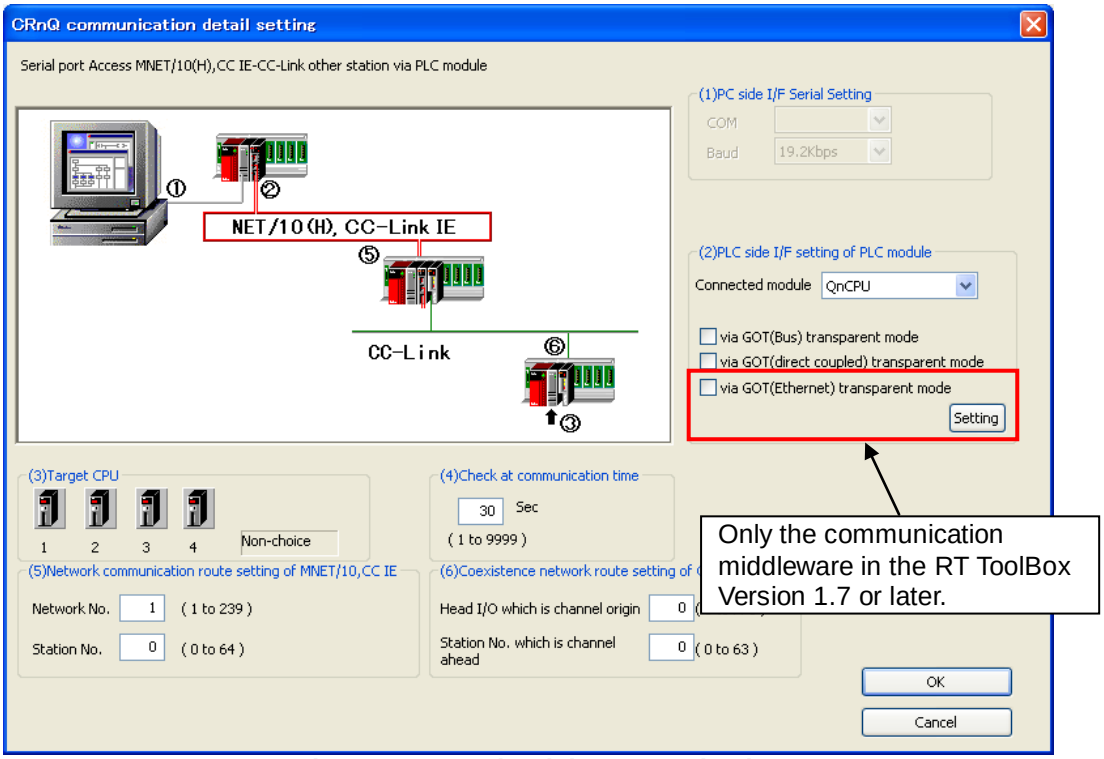


Figure 4-9 USB (Serial Communication NET/10(H) through CPU Unit – Serial Communication Other Station Access) Detailed Settings

For some communications routes that you can select, "(5)Network communication route setting of MNET/10" and "(6)Coexistence network route setting of C24" can not be set.

GOT(Ethernet) transparent mode can be used with the communication middleware in the RT ToolBox2 Version 1.7 or later. When you use GOT(Ethernet) transparent mode, please display a set screen clicking the “Setting” button, and set the communication item.

4.2.5.3 When NET/10(H) is selected

When "NET/10" is selected as the computer interface, the communications routes that you can select are as follows.

Table 4-9 Communications Routes That Can Be Selected for "NET/10(H)"

No.	Communications Route
1	NET/10 board communication Other station access through NET/10(H) unit
2	NET/10 board communication NET/10(H) through NET/10(H) unit – CC-Link other station access
3	NET/10(H) board communication NET/10(H) through NET/10(H) unit - Serial communication other station access

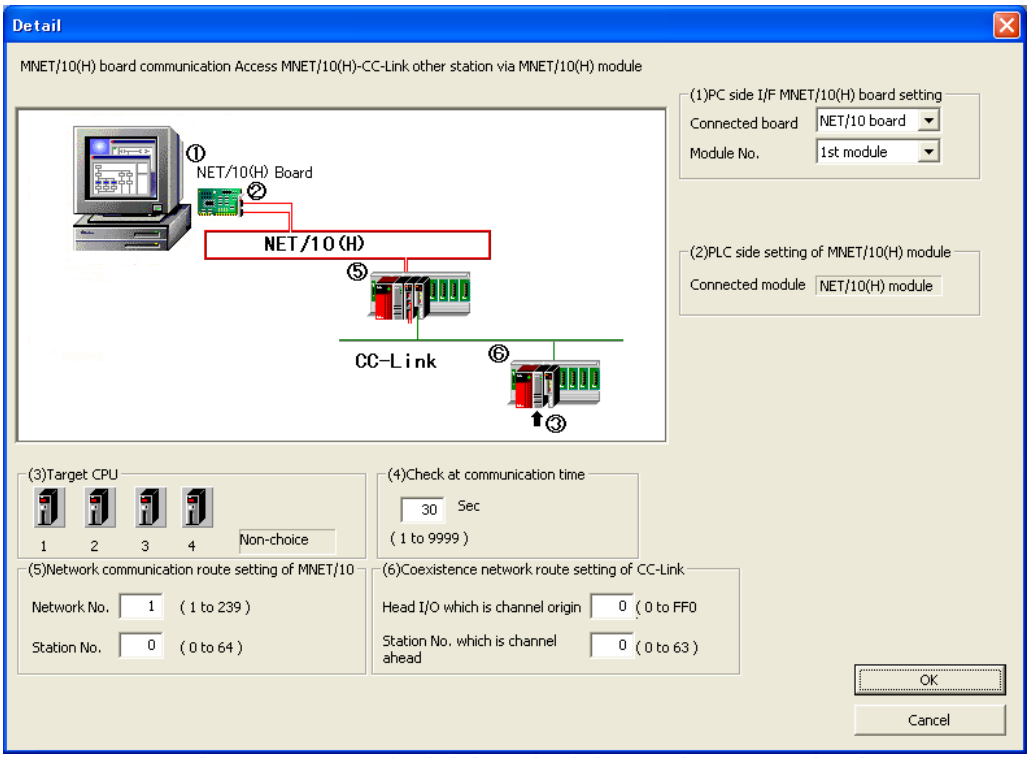


Figure 4-10 NET/10(H) (NET/10(H) Board Communication NET/10(H) through NET/10(H) Unit – CC-Link Other Station Access) Detailed Settings

For some communications routes that you can select, "(5)Network communication route setting of MNET/10" and "(6)Coexistence network route setting of C24" can not be set.

4.2.5.4 When CC-Link is selected

When "CC-Link" is selected as the computer interface, the communications routes that you can select are as follows.

Table 4-10 Communications Routes That Can Be Selected for "CC-Link"

No.	Communications Route
1	CC-Link board communication Other station access through CC-link unit
2	CC-Link board communication CC-Link through CC-link unit – NET/10(H) other station access
3	CC-Link board communication CC-Link through CC-link unit – Ethernet other station access

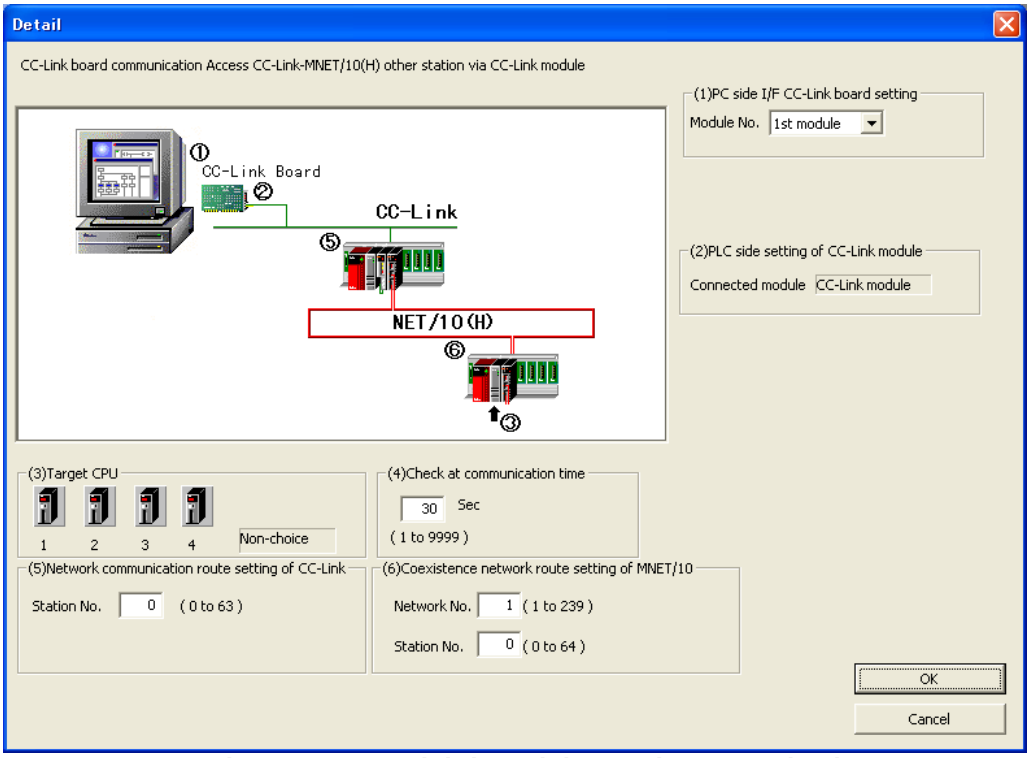


Figure 4-11 CC-Link (CC-Link Board Communication CC-Link through CC-Link Unit – NET/10(H) Other Station Access) Detailed Settings

For some communications routes that you can select, "(5)Network communication route setting of MNET/10" and "(6)Coexistence network route setting of C24" can not be set.

4.2.5.5 When Ethernet is selected

When "Ethernet" is selected as the computer interface, the communications routes that you can select are as follows.

Table 4-11 Communications Routes That Can Be Selected for "Ethernet"

No.	Communications Route	Special mention
1	Ethernet board communication Access other station via Ethernet module	
2	Ethernet board communication Access Ethernet-serial communication other station via Ethernet module	
3	Ethernet board communication Access Ethernet-CC-Link other station via Ethernet module	
4	Ethernet port direct communication PLC module connection	Version 1.2 or later
5	Ethernet port direct communication Access serial communication other station via PLC module	
6	Ethernet port direct communication Access CC-Link other station via PLC module	
7	Ethernet port IP address designate communication PLC module connection	
8	Ethernet port IP address designate communication Access serial communication other station via PLC module	
9	Ethernet port IP address designate communication Access CC-Link other station via PLC module	

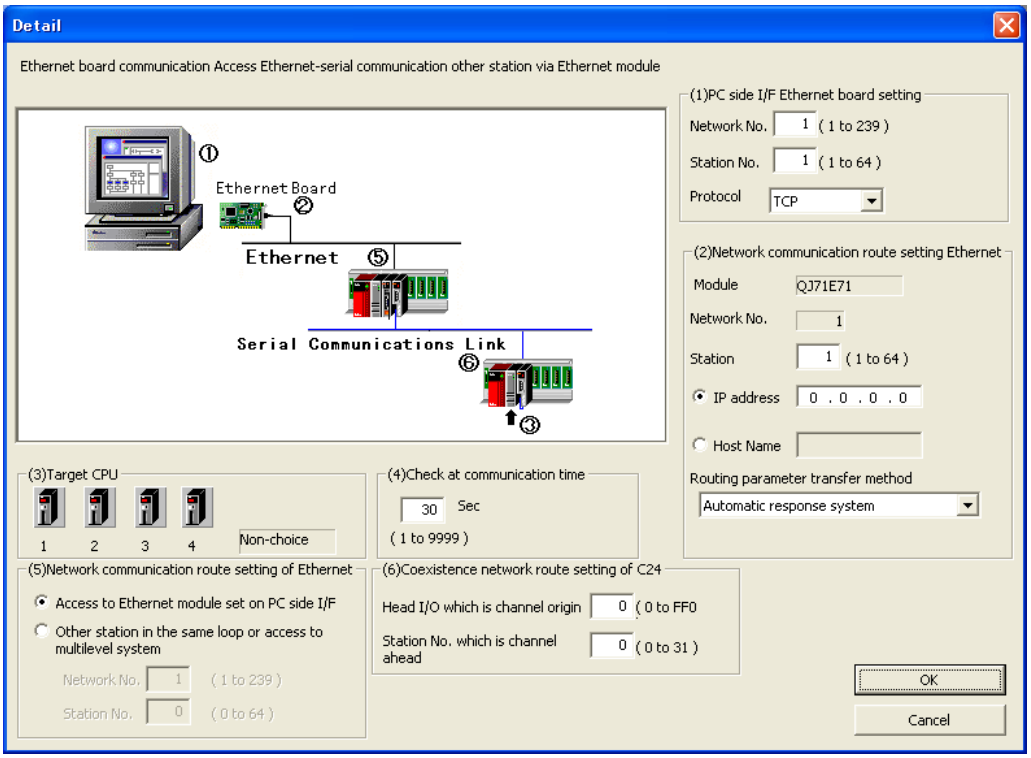


Figure 4-12 Ethernet (Ethernet board communication Access Ethernet-serial communication other station via Ethernet module) Detailed Settings

For some communications routes that you can select, “(1)PC side I/F Ethernet board setting”, “(2)Network communication route setting Ethernet”, "(5)Network communication route setting of MNET/10" and "(6)Coexistence network route setting of C24" can not be set.

4.2.5.6 When Q series bus is selected

When the "Q series bus" is selected as the computer interface, the communication route that can be selected is as follows.

Table 4-12 Communication Route That Can Be Selected with the "Q Series Bus"

No.	Communications Route
1	Q bus communications CPU unit connection

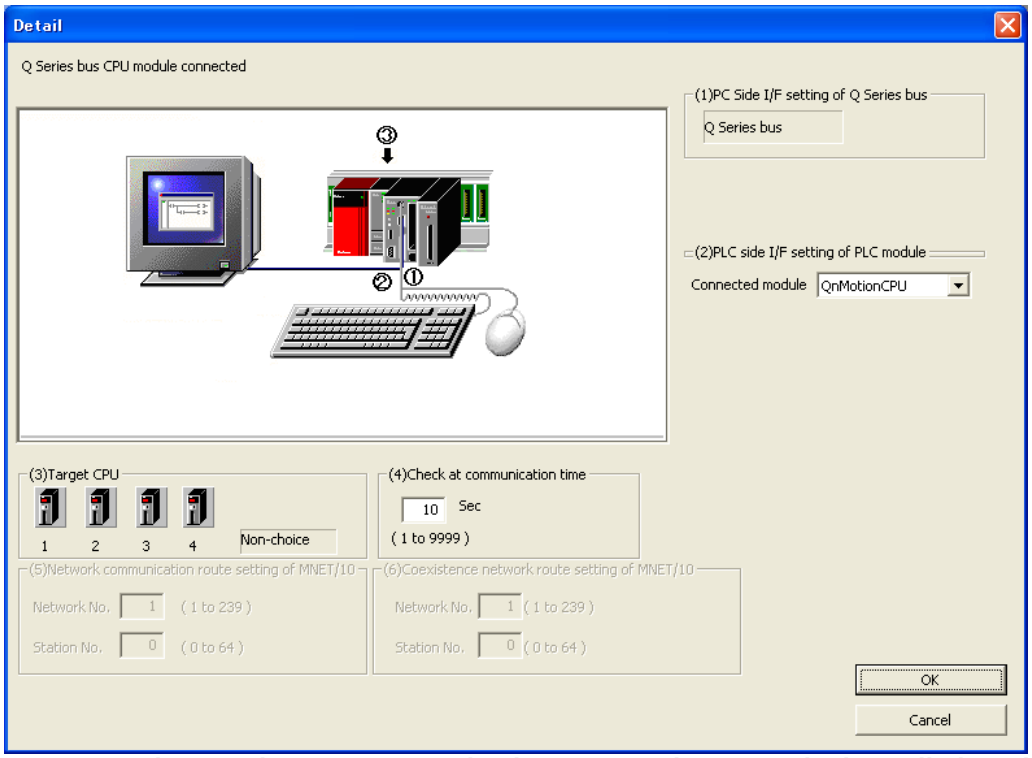


Figure 4-13 Series Bus (Q Bus Communications CPU Unit Connection) Detailed Settings

4.2.6 Robot Controller Connection Setting List

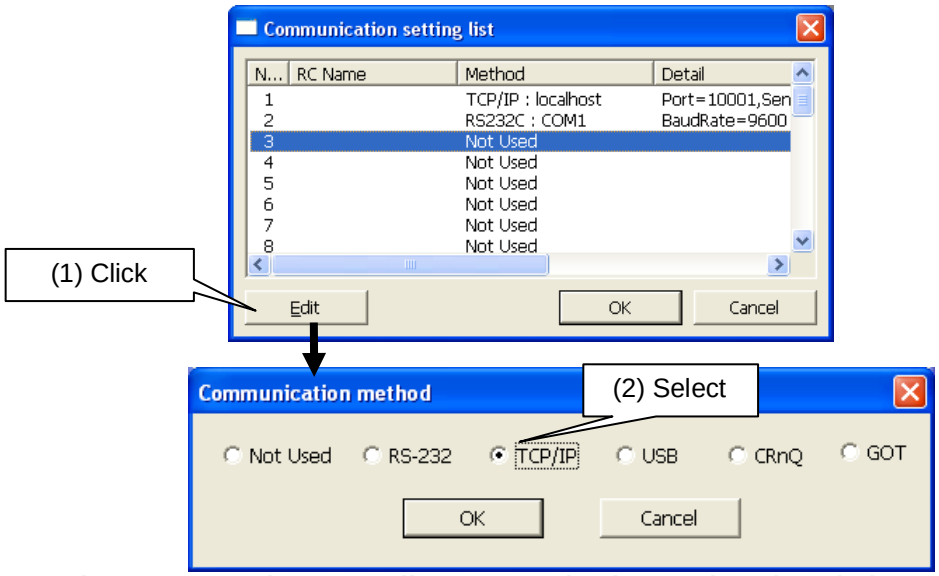


Figure 4-14 : Robot Controller Communication Setting List Window

1. Select the line you want to set, and then click the [Edit] button. A screen for selecting the communication method appears.
2. Select the communication method, and then click the [OK] button. The communication settings window applicable to the communication method you selected appears.
3. When editing is complete, click the [OK] button. The settings are complete.

⚠ CAUTION

- **<u>Do not communicate with a robot controller by connecting duplicate settings for the same robot controller.</u>**
  Doing so may some problems such as a communication error or a missing data file.
  Example of incorrect connection: Connecting the same controller via RS-232 and TCP/IP.
- **<u>If multiple robot controllers are connected and used, verify the operation targets first, and then execute.</u>**

4.3 Information of Connected Robot Controllers

Information of robot controllers that are set up for communication can be displayed in a list.

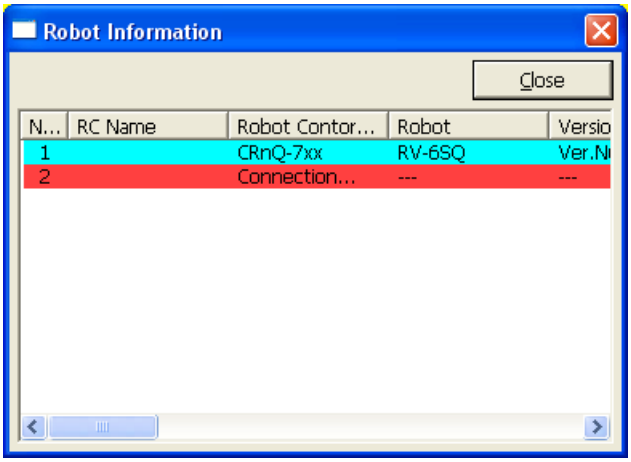


Figure 4-15 : Connected Robot Controller Information List

If robot controllers have been connected, the information of those robot controllers is displayed in the fields following the [Robot Controller] field. If they have not been connected, the description of a communication error is displayed in the [Robot Controller] field.
Double-click [No.] in the list to display the information of the robot controller selected.

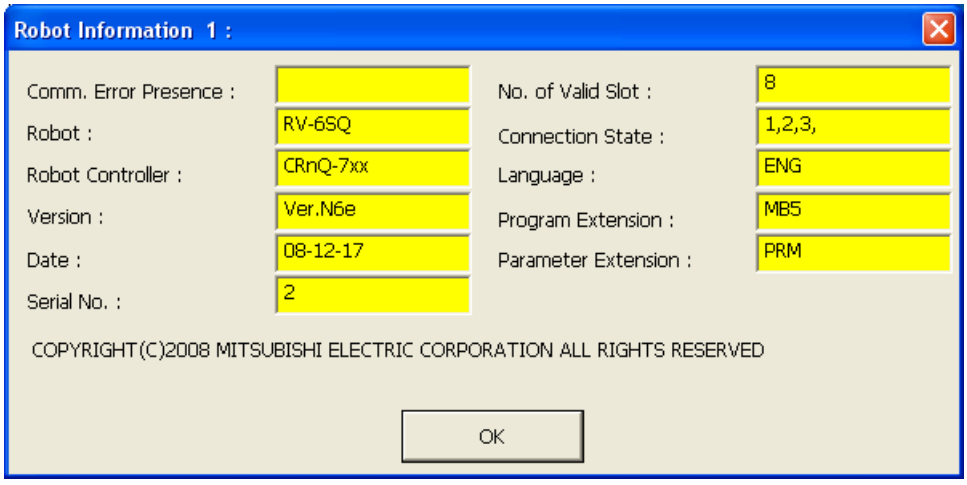


Figure 4-16 : Robot Information Window for a Connected Robot Controller

5 Methods

This chapter provides a list of methods and describes each method in detail.

5.1 Method List

Table 5-1 : Communication Server 2 Related

Method name	Description
ServerLive	Checks the activation status of communication server 2.
ServerStart	Starts communication server 2.
ServerKill	Shuts down communication server 2.
GetRoboComSetting	Gets the identifiers and names of robot controllers set by communication server 2. This method supports the multi-controller function.

Table 5-2 : Request for Service/Request to Cancel

Method name	Description
RequestService RequestServiceM	Sends a request specified. Sends by specifying the length of data required for the request. Gets reception data using a get received data (GetRecvData, GetRecvDataM) method after a reception complete event has occurred. The RequestServiceM method supports the multi-controller function.
RequestService2Null RequestService2NullM	Sends a request specified. Gets reception data using a get received data (GetRecvData, GetRecvDataM) method after a reception complete event has occurred. These methods are the same as the RequestService and RequestServiceM methods above, however, these methods do not specify the length of data required for the request. Also, these methods send requests by replacing the CR-LF of transmission data to be specified with LF. The RequestService2NullM method supports the multi-controller function.
RequestCancel RequestCancelM	Cancels a request. The RequestCancelM method supports the multi-controller function.

Table 5-3 : Data Reception

Method name	Description
GetRecvData GetRecvDataM	Gets received data. The GetRecvDataM method supports the multi-controller function.
GetRecvLen	Gets the length of received data.
GetDataCnt	Gets the item count of received data.
GetOneData GetOneDataCPP GetOneDataBCP	Gets only the specified item of received data. The GetOneDataCPP method supports the Visual C++. The GetOneDataBCP method supports the Borland C++ Builder.
CheckRecvMsg	Gets the number of received messages. The message reception status can be checked without using a reception complete event.

Table 5-4 : Get Robot Controller Information

Method name	Description
ConnectRobot CheckConnectingM	Checks for connection with robot controllers. The CheckConnectingM method supports the multi-controller function.
RoboErrorNo GetRoboErrorNoM	Gets the number of a system error occurring in a robot controller. The GetRoboErrorNoM method supports the multi-controller function.
RoboProgramEx GetRoboProgramExM	Gets the file extension of the robot program file used by a robot controller. The GetRoboProgramExM method supports the multi-controller function.
RoboParameterEx GetRoboParameterExM	Gets the file extension of the parameter file used by a robot controller. The GetRoboParameterExM method supports the multi-controller function.
RoboMechType GetRoboMechTypeM	Gets the mechanical type of the robot. The GetRoboMechTypeM method supports the multi-controller function.
RoboControler GetRoboControllerM	Gets the model name of the robot controller. The GetRoboControllerM method supports the multi-controller function.
RoboDate GetRoboDateM	Gets the date information in the robot controller. The GetRoboDateM method supports the multi-controller function.
RoboVersion GetRoboVersionM	Gets the version information of the robot controller. The GetRoboVersionM method supports the multi-controller function.
RoboLanguage GetRoboLanguageM	Gets the language information of the robot controller. The GetRoboLanguageM method supports the multi-controller function.
RoboMechNo GetRoboMechNoM	Gets the mechanical connection status of the robot controller. The GetRoboMechNoM method supports the multi-controller function.
RoboSlotCnt GetRoboSlotCntM	Gets the slot count of the robot controller. The GetRoboSlotCntM method supports the multi-controller function.
RoboCopyright GetRoboCopyrightM	Gets the copyright information of the robot controller. The GetRoboCopyrightM method supports the multi-controller function.
RoboComment GetRoboCommentM	Gets a comment of the robot controller. <Reserved Future> The GetRoboCommentM method supports the multi-controller function.

5.2 Detailed Description of Methods

The method indicated as ***<Multi-Controller>*** supports the multi-controller function. If only one controller is used, specify "1" for the robot controller identifier of a method that supports the multi-controller function.
[Example] is described by Visual Basic.

CAUTION

Notes when developing by Visual C++

The BSTR type variable passed to MelfaRXM.ocx should open a memory using a SysFreeString( ) function.

5.2.1 Methods Related to Communication Server 2

The following describes the methods related to Communication Server 2.

POINT

For communications with robot controllers,
it is necessary to start communication server.

It is necessary to start communication server in order to perform communications with robot controllers. Start communication server using the start communication server (ServerStart) method.

ServerLive

[Function]
Gets the status of Communication Server 2.

[Syntax]

```
Boolean ServerLive ( )
```

[Return value]
Returns the activation status of Communication Server 2.
1 : Started
0 : Not started

[Description]
This method checks whether or not Communication Server 2 has been started.

[Example]

```
If ctrMelfaRxM.ServerLive = False Then
    ctrMelfaRxM.ServerStart
End If
```

ServerStart

[Function]

Starts Communication Server 2.

[Syntax]

Boolean ServerStart ( )

[Return value]

Returns the result.

1 : Successful

0 : Failed

[Description]

(1) Communication Server 2 must be installed in the same folder as that of applications.
Note that multiple communication servers cannot be started simultaneously. Even if an attempt is made repeatedly to start multiple servers at the same time, only one Communication Server 2 can be started.

(2) Communication Server 2 that has been started must be shut down with the kill Communication Server 2 (ServerKill) method when quitting applications.

[Example]

See Example in ServerLive (get Communication Server 2 status).

ServerKill

[Function]

Shuts down Communication Server 2.

[Syntax]

Boolean ServerKill ( )

[Return value]

Returns the result.

1 : Successful

0 : Failed

[Description]

Communication Server 2 that has been started must be shut down with this method when quitting applications.

[Example]

```
Private Sub Form_Unload(Cancel As Integer)
    MelfaRxM.ServerKill
End Sub
```

GetRoboComSetting <Multi-Controller>

[Function]

Gets the identifiers and names of robot controllers set by Communication Server 2.

[Syntax]

Boolean Getrobocomsetting (long Cnt, String *Data*)

[Arguments]

[OUT] *Cnt* The number of robot controllers set by Communication Server 2 is set.

[OUT] *Data* The identifiers and names of robot controllers set by Communication Server 2 are set.

[Return value]

Returns the connection status with the robot controller.

1 : Being connected

0 : Not connected

[Description]

The following strings are set for argument data:

Identifier 1 <LF> name 1 <LF>
Identifier 2 <LF> name 2 <LF>
:
Identifier N <LF> name N

[Example]

```
Dim bState As Boolean
Dim lCnt As Long
Dim sRobotInf As String
Dim sID As String
Dim sName As String
Dim sData As String
Dim I As Long

choRobotID.Clear
bState = ctrMelfaRxM.GetRoboComSetting(lCnt, sRobotInf)
If bState = True Then
    I = 0
    Do While I < lCnt
        sID = ctrMelfaRxM.GetOneData(I * 2, sRobotInf)
        sName = ctrMelfaRxM.GetOneData(I * 2 + 1, sRobotInf)
        sData = sID + " : " + sName
        choRobotID.AddItem (sData)
        I = I + 1
    Loop
End If
```

5.2.2 Request for Service/Request to Cancel

These methods send a request or cancel a request.

RequestService
RequestServiceM <Multi-Controller>

[Function]

Sends a request specified.

[Syntax]

long RequestService (long *MsgID*, long *Length*,
String *Data*, long *Display*, long *Cycle*)
long RequestServiceM (long *RobotID*, long *MsgID*, long *Length*,
String *Data*, long *Display*, long *Cycle*, short *Priority*)

[Arguments]

[IN] *RobotID*	Specify the identifier (1 to 32) of the robot controller.
[IN] *MsgID*	Specify the ID to identify a request. For more information about the message ID, see **5.3**, **"Request IDs Specified by Request for Service Methods"**.
[IN] *Length*	Specify the length of data to be transmitted.
[IN] *Data*	Specify the transmission data. The contents of data vary depending on the request ID. For more information, see **5.3**, **"Request IDs Specified by Request for Service Methods"**.
[IN] *Display*	Specify a number other than 0 to display a progress bar during communication. Specify 0 to not display a progress bar during communication.
[IN] *Cycle*	Specify the reception cycle for monitoring the status of robot controllers, etc. For more information about the mechanism of monitoring, see Description (3). Specify 0 if no monitoring is required. Unit: [msec]
[IN] *Priority*	Specify 0.

[Return value]

Returns the transmission result to Communication Server 2.

1 : Successfully received by Communication Server 2
0 : Transmission failed
-1 : Communication Server 2 has not been started.

[Description]

(1) Both of these methods are used to specify the length of data required for a request and to send the request.

(2) When one of these methods is called, Communication Server 2 performs transmission processing, and sends a request to the robot controller(s). When MelfaRXM.ocx receives a transmission complete message, a reception complete event (MsgRecvEvent) occurs. After the reception complete event (MsgRecvEvent) occurred, get it the using a get received data (GetRecvData, GetRecvDataM) method. Also, use these methods to get received data at the same time.

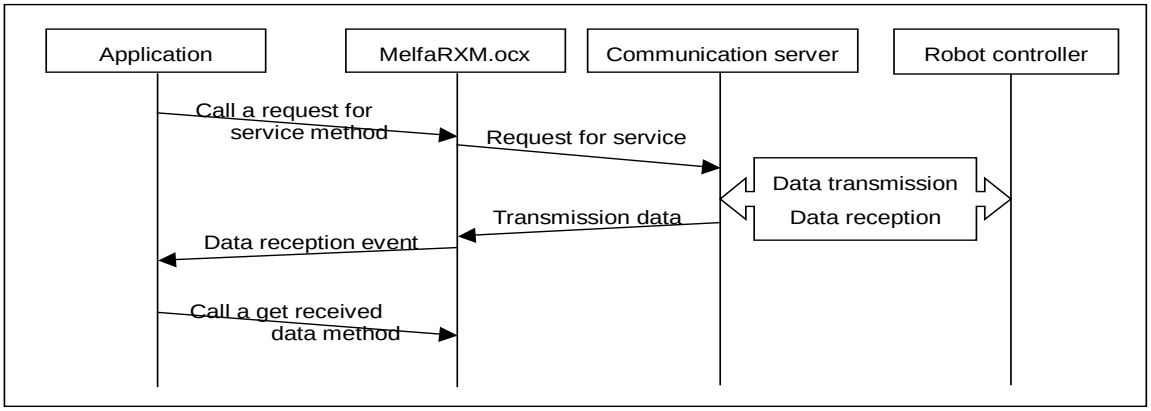


Figure 5-1 : Flow from Request for Service to Data Reception

(3) If a value is specified in *Cycle*, the content requested by one request for service (RequestService, RequestServiceM) method is continuously received on the specified cycle. This is used to monitor the status of robot controllers, etc.
To end monitoring, it is necessary to send a request to cancel using a request to cancel (RequestCancel, RequestCancelM) method.

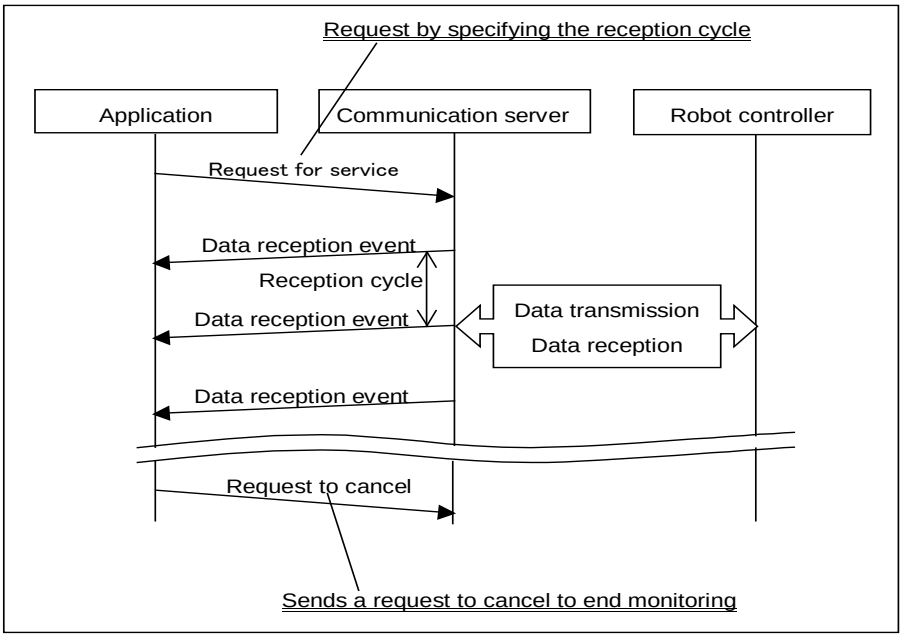


Figure 5-2 : Mechanism of Monitoring

(4) If the RequestService method is used in a multi-controller environment, it is the same as specifying “1” in *RobotID* of the RequestServiceM method.

[Example]

(1) Normal request

```
Dim sData As String
Dim lStatus As Long
sData = "1"
lStatus = MelfaRxM.RequestServiceM(1, 100, Len(sData), sData, 1, 0, 0)
```

(2) Monitoring request

```
Dim sData As String
Dim lStatus As Long
sData = ""
lStatus = MelfaRxM.RequestServiceM(1, 203, 0, sData, 0, 1000, 0)
```

RequestService2Null
RequestService2NullM <Multi-Controller>

[Function]

Sends a request specified.

[Syntax]

long RequestService2Null (long *MsgID*, String *Data*,
long *Display*, long *Cycle*)
long RequestService2NullM (long *RobotID*, long *MsgID*, String *Data*,
long *Display*, long *Cycle*, long *Priority*)

[Arguments]

[IN] *RobotID*	Specify the identifier (1 to 32) of the robot controller.
[IN] *MsgID*	Specify the ID to identify a request. For more information about the message ID, see **5.3**, **"Request IDs Specified by Request for Service Methods"**.
[IN] *Data*	Specify the transmission data. The contents of data vary depending on the request ID. For more information, see **5.3**, **"Request IDs Specified by Request for Service Methods"**.
[IN] *Display*	Specify a number other than 0 to display a progress bar during communication. Specify 0 to not display a progress bar during communication.
[IN] *Cycle*	Specify the reception cycle for monitoring the status of robot controllers, etc. For more information about the mechanism of monitoring, see Description (4). Specify 0 if no monitoring is required. Unit: [msec]
[IN] *Priority*	Specify 0.

[Return value]

Returns the transmission result to Communication Server 2.

1 : Successfully received by Communication Server 2
0 : Transmission failed
-1 : Communication Server 2 has not been started.

[Description]

(1) Both of these methods send data up to the NULL character in *Data*. (The data length is not specified.)
The contents of these methods are the same as those of the RequestService and RequestServiceM methods. However, the CR-LF (carriage return and line feed) sequence in *Data* is replaced with LF (line feed character), so text edited in text boxes and others can be used as transmission data as is. This replacement does not affect the return value.

(2) When one of these methods is called, Communication Server 2 performs transmission processing, and sends a request to the robot controller(s). When MelfaRXM.ocx receives a transmission complete message, a reception complete event (MsgRecvEvent) occurs. After the reception complete event (MsgRecvEvent) occurred, get it using a get received data (GetRecvData, GetRecvDataM) method. Also, use these methods to get received data at the same time.

(3) If a value is specified in *Cycle*, the content requested by one request for service (RequestService2Null, RequestService2NullM) method is continuously received on the specified cycle. This is used to monitor the status of robot controllers, etc.
To end monitoring, it is necessary to send a request to cancel using a request to cancel (RequestCancel, RequestCancelM) method.

(4) If the RequestService2Null method is used in a multi-controller environment, it is the same as specifying “1” in *RobotID* of the RequestService2NullM method.

[Example]

(1) Normal request

```
Dim sData As String
Dim lStatus As Long
sData = "1"
lStatus = MelfaRxM.RequestService2NullM(1, 100, sData, 1, 0, 0)
```

(2) Monitoring request

```
Dim sData As String
Dim lStatus As Long
sData = ""
lStatus = MelfaRxM.RequestService2NullM(1, 203, sData, 0, 1000, 0)
```

RequestCancel
RequestCancelM <Multi-Controller>

[Function]

Cancels a request specified or all requests.

[Syntax]

long RequestCancel (long *MsgID*)
long RequestCancelM (long *RobotID*, long *MsgID*)

[Arguments]

[IN] *RobotID* Specify the identifier (1 to 32) of the robot controller.

[IN] *MsgID* Specify the ID to identify a request. For more information about the message ID, see **5.3**, **"Request IDs Specified by Request for Service Methods"**.
To cancel all requests issued to the corresponding OCX, specify 0.

[Return value]

Returns the transmission result.

1 : Successfully received by Communication Server 2
0 : Transmission failed
-1 : Communication Server 2 has not been started.

[Description]

(1) To end monitoring requested by a request for service (RequestService, RequestServiceM, RequestService2Null, RequestService2NullM) method, it is necessary to cancel the request using this method.

(2) If the RequestCancel method is used in a multi-controller environment, it is the same as specifying "1" *RobotID* of the RequestCancelM method.

[Example]

```
Dim lStatus As Long
lStatus = MelfaRxM.RequestCancelM(1, 106)
```

5.2.3 Data Reception

These methods are used to handle data reception. They are used after a reception complete event (MsgRecvEvent) occurs.

GetRecvData
GetRecvDataM <Multi-Controller>

[Function]

Gets received data.

[Syntax]

long GetRecvData (long *MsgID*, String *Data*, long *Status*, long *Error*)
long GetRecvDataM (long *RobotID*, long *MsgID*, String *Data*,
long *Status*, long *Error*)

[Arguments]

[OUT] *RobotID* The identifier (1 to 32) of the robot controller is set.

[OUT] *MsgID* The ID to identify a request is set. For more information about the message ID, see **5.3**, **"Request IDs Specified by Request for Service Methods"**.

[OUT] *Data* Reception data is set. The contents of data vary depending on the request ID. For more information, see **5.3**, **"Request IDs Specified by Request for Service Methods"**.

[OUT] *Status* The reception status is set.
If a number other than 1 (received successfully) was set, an error message is set in *Data*.
1 : Received successfully
2 : Transmission error
3 : Reception timeout
4 : Transmission canceled(when a request was canceled by pressing the [Cancel] button of Communication Server 2)
5 : Execution error (execution result was an error)
10 : Undefined request (when an undefined request ID was specified)
11 : Invalid argument for a request (when a request ID and transmission data do not match)

[OUT] *Error* When "5" is set in *Status*, an error number of the robot controller is set.

[Return value]

Returns the reception result.
1 : Successful
0 : Invalid data

[Description]

(1) If a request for service (RequestService, RequestServiceM, RequestService2Null, RequestService2NullM) method is called, a reception complete event (MsgRecvEvent) occurs. Get the transmission result and received data with this method.

(2) If the GetRecvData method is used in a multi-controller environment, it is the same as specifying "1" in *RobotID* of the GetRecvDataM method.

[Example]

```
Private Sub MelfaRXM_MsgRecvEvent()
    Dim lState As Long
    Dim lID As Long
    Dim sRecvData As String
    Dim lStatus As Long
    Dim lError As Long
    Dim I As Long
    Dim lCnt As Long
    Dim sData As String
    Dim sString As String

    lState = MelfaRxM.GetRecvDataM(1, lID, sRecvData, lStatus, lError)
    If lStatus = 1 Then
        sString = ""
        lCnt = MelfaRxM.GetDataCnt(sRecvData)
        For I = 0 To (lCnt - 1)
            sData = MelfaRxM.GetOneData(I, sRecvData)
            sString = sString + sData + Chr$(13) + Chr$(10)
        Next
        tbRecvData.Text = sString
            :
            :
End Sub
```

GetRecvLen

[Function]

Gets the length of received data to be acquired by the get received data (GetRecvData) method.

[Syntax]

```
long GetRecvLen ( )
```

[Return value]

Returns the length of received data.

[Description]

This method can be used to get the data length in advance before calling the get received data method.

[Example]

```
Private Sub MelfaRXM_MsgRecvEvent()
    Dim lState As Long
    Dim lSize As Long
    Dim lID As Long
    Dim sRecvData As String
    Dim lStatus As Long
    Dim lError As Long

    lSize = GetRecvLen()
    sRecvData = Space$(lSize)
    lState = MelfaRxM.GetRecvDataM(1, lID, sRecvData, lStatus, lError)
        :
        :
End Sub
```

GetDataCnt

[Function]

Gets the received data count.

[Syntax]

long GetDataCnt (String *Data*)

[Arguments]

[IN] *Data* Specify the received data acquired by the get received data (GetRecvData) method. The contents of data vary depending on the request ID. For details, see **5.3**, **"Request IDs Specified by Request for Service Methods"**.

[Return value]

Returns the item count of the received data specified.

[Description]

This method gets the item count of received data.

[Example]

See Examples of GetRecvData and GetRecvDataM (get received data).

GetOneData

[Function]

Gets only the specified item of received data.

This method can not be used by Visual C++ and Borland C++Builder.

[Syntax]

String GetOneData (long *Point*, String *Data*)

[Arguments]

[IN] *Point* Specify the index of the item to be acquired. Index starts with 0.

[IN] *Data* Specify the received data acquired by the get received data (GetRecvData) method. The contents of data vary depending on the request ID. For more information, see **5.3**, **"Request IDs Specified by Request for Service Methods"**.

[Return value]

Returns the item of the specified index.

[Description]

This method gets only the specified item of received data.

[Example]

See Examples of get received data (GetRecvData and GetRecvDataM) methods.

GetOneDataCPP

[Function]

Gets only the specified item of received data.

This method supports the Visual C++, and can not be used by Borland C++Builder.

[Syntax]

long GetOneDataCPP (long *Point*, LPCTSTR *Data*, BSTR* *Onedata*)

[Arguments]

[IN] *Point*	Specify the index of the item to be acquired. Index starts with 0.
[IN] *Data*	Specify the received data acquired by the get received data (GetRecvData) method. The contents of data vary depending on the request ID. For more information, see **5.3**, **"Request IDs Specified by Request for Service Methods"**.
[OUT] Onedata	The item of the specified index is set.

[Return value]

Returns the result.

1 : Successful

0 : Failed

[Description]

This method gets only the specified item of received data.

This method supports the Visual C++, and can not be used by Borland C++Builder.

[Example](Visual C++)

```
void CMRxSampleView::OnMsgRecvEventMelfarxmctrl1() {
    long msgcnt, i;
    BSTR data;
    CString tmp, tmp2;
    data = NULL;
    m_cMrx.GetRecvDataM(&m_nRecvRobot, &m_nRecvID, &data,
                                &m_nRecvStatus, &m_nRecvErr);
    tmp = data;
    ::SysFreeString(data);
    msgcnt = m_cMrx.GetDataCnt(tmp);
    for (i = 0; i < msgcnt; i ++) {
        data = NULL;
        m_cMrx.GetOneDataCPP(i, tmp, &data);
                tmp2 += data;
                ::SysFreeString(data);
                tmp2 += _T("¥r¥n");
    }
    UpdateData(FALSE);
    GetDlgItem(IDC_RECV_DATA)->SetWindowText(tmp2);
        :
        :
}
```

GetOneDataBCP

[Function]

Gets only the specified item of received data.

This method supports the Borland C++Builder, and can not be used by Visual C++.

[Syntax]

long GetOneDataBCP (long *Point*, BSTR* *Data*, BSTR* *Onedata*)

[Arguments]

[IN] *Point*	Specify the index of the item to be acquired. Index starts with 0.
[IN] *Data*	Specify the received data acquired by the get received data (GetRecvData) method. The contents of data vary depending on the request ID. For more information, see **5.3**, **"Request IDs Specified by Request for Service Methods"**.
[OUT] Onedata	The item of the specified index is set.

[Return value]

Returns the result.

1 : Successful

0 : Failed

[Description]

This method gets only the specified item of received data.

This method supports the Borland C++Builder, and can not be used by Visual C++.

[Example](Borland C++Builder)

```
void __fastcall TFormMain::MsgReceive(TObject *Sender){
    long lState, lReqID, lStatus, lError;
    WideString sRecvData="", sOneData="";
    long i, lCnt, lRobotID;
    AnsiString sAnsi="";
    lState = MelfaRxM1->GetRecvDataM(&lRobotID, &lReqID,
                                     &sRecvData, &lStatus, &lError);
    if (lStatus == 1) {
        sAnsi = "";
        lCnt = MelfaRxM1->GetDataCnt(sRecvData);
        for (i=0; i<lCnt; i++) {
            MelfaRxM1->GetOneDataBCP(i, &sRecvData, &sOneData);
            sAnsi = sAnsi + sOneData + "¥r¥n";
        }
        MemoRecv->Lines->Text = sAnsi;
            :
            :
    }
}
```

CheckRecvMsg

[Function]

Gets the number of received messages.

[Syntax]

long CheckRecvMsg ( )

[Return value]

Returns the number of received messages.

[Description]

Normally, the reception of a message is checked by the occurrence of a reception complete event (MsgRecvEvent). This method can also be used to check the message reception status without using the reception complete event.

[Example]

```
Dim lState As Long
Dim lID As Long
Dim sRecvData As String
Dim lStatus As Long
Dim lError As Long

If MelfaRxM.CheckRecvMsg() > 0 Then
    lState = MelfaRxM.GetRecvDataM(1, lID, sRecvData, lStatus, lError)
        :
        :
End If
```

5.2.4 Getting the Status of Robot Controllers

These methods get the status of robot controllers.

ConnectRobot
CheckConnectingM <Multi-Controller>

[Function]

Checks the connection status with robot controllers.

[Syntax]

Boolean ConnectRobot ( )
Boolean CheckConnectingM (long *RobotID*)

[Arguments]

[IN] *RobotID* Specify the identifier (1 to 32) of the robot controller.

[Return value]

Returns the connection status with the robot controller.

1 : Being connected
0 : Not connected

[Description]

If the ConnectRobot method is used in a multi-controller environment, it is the same as specifying "1" in *RobotID* of the CheckConnectingM method.

[Example]

```
Dim lStatus As Long
lStatus = MelfaRxM.CheckConnectingM(1)
```

RoboErrorNo
GetRoboErrorNoM <Multi-Controller>

[Function]

Gets the number of a system error occurring in a robot controller.

[Syntax]

Boolean RoboErrorNo (long *Data*)
Boolean GetRoboErrorNoM (long *RobotID*, long *Data*)

[Arguments]

[IN] *RobotID* Specify the identifier (1 to 32) of the robot controller.
[OUT] *Data* The error number of the robot controller is set.

[Return value]

Returns the connection status with the robot controller.

1 : Being connected
0 : Not connected

[Description]

If the RoboErrorNo method is used in a multi-controller environment, it is the same as specifying "1" in *RobotID* of the GetRoboErrorNoM method.

[Example]

```
Dim lStatus As Long
Dim sData As String
lStatus = MelfaRxM.GetRoboErrorNoM(1, sData)
```

RoboProgramEx
GetRoboProgramExM <Multi-Controller>

[Function]

Gets the file extension of the robot program file used in a robot controller.

[Syntax]

Boolean RoboProgramEx (String *Data*)
Boolean GetRoboProgramExM (long *RobotID*, String *Data*)

[Arguments]

[IN] *RobotID* Specify the identifier (1 to 32) of the robot controller.
[OUT] *Data* The file extension of the robot program file is set.

[Return value]

Returns the connection status with the robot controller.

1 : Being connected
0 : Not connected

[Description]

If the RoboProgramEx method is used in a multi-controller environment, it is the same as specifying "1" in *RobotID* of the GetRoboProgramExM method.

[Example]

```
Dim lStatus As Long
Dim sData As String
lStatus = MelfaRxM.GetRoboProgramExM(1, sData)
```

RoboParameterEx
GetRoboParameterExM <Multi-Controller>

[Function]

Gets the file extension of the parameter file used in a robot controller.

[Syntax]

Boolean RoboParameterEx (String *Data*)
Boolean GetRoboParameterExM (long *RobotID*, String *Data*)

[Arguments]

[IN] *RobotID* Specify the identifier (1 to 32) of the robot controller.
[OUT] *Data* The file extension of the parameter file is set.

[Return value]

Returns the connection status with the robot controller.

1 : Being connected
0 : Not connected

[Description]

If the RoboParameterEx method is used in a multi-controller environment, it is the same as specifying "1" in *RobotID* of the GetRoboParameterExM method.

[Example]

```
Dim lStatus As Long
Dim sData As String
lStatus = MelfaRxM.GetRoboParameterExM(1, sData)
```

RoboMechType
GetRoboMechTypeM <Multi-Controller>

[Function]

Gets the machine type of a robot.

[Syntax]

Boolean RoboMechType (String *Data*)
Boolean GetRoboMechTypeM (long *RobotID*, String *Data*)

[Arguments]

[IN] *RobotID* Specify the identifier (1 to 32) of the robot controller.

[OUT] *Data* The machine type of the robot is set.

[Return value]

Returns the connection status with the robot controller.

1 : Being connected

0 : Not connected

[Description]

If the RoboMechType method is used in a multi-controller environment, it is the same as specifying "1" in *RobotID* of the GetRoboMechTypeM method.

[Example]

```
Dim lStatus As Long
Dim sData As String
lStatus = MelfaRxM.GetRoboMechTypeM(1, sData)
```

RoboControler
GetRoboControllerM <Multi-Controller>

[Function]

Gets the model name of robot controller.

[Syntax]

Boolean RoboControler (String *Data*)
Boolean GetRoboControllerM (long *RobotID*, String *Data*)

[Arguments]

[IN] *RobotID* Specify the identifier (1 to 32) of the robot controller.

[OUT] *Data* The model name of the robot controller is set.

[Return value]

Returns the connection status with the robot controller.

1 : Being connected

0 : Not connected

[Description]

If the RoboControler method is used in a multi-controller environment, it is the same as specifying "1" in *RobotID* of the GetRoboControllerM method.

[Example]

```
Dim lStatus As Long
Dim sData As String
lStatus = MelfaRxM.GetRoboControllerM(1, sData)
```

RoboDate
GetRoboDateM <Multi-Controller>

[Function]

Gets the date information in a robot controller.

[Syntax]

Boolean RoboDate (String *Data*)
Boolean GetRoboDateM (long *RobotID*, String *Data*)

[Arguments]

[IN] *RobotID* Specify the identifier (1 to 32) of the robot controller.
[OUT] *Data* The date information in the robot controller is set.

[Return value]

Returns the connection status with the robot controller.

1 : Being connected
0 : Not connected

[Description]

If the RoboDate method is used in a multi-controller environment, it is the same as specifying "1" in *RobotID* of the GetRoboDateM method.

[Example]

```
Dim lStatus As Long
Dim sData As String
lStatus = MelfaRxM.GetRoboDateM(1, sData)
```

RoboVersion
GetRoboVersionM <Multi-Controller>

[Function]

Gets the version information in a robot controller.

[Syntax]

Boolean RoboVersion (String *Data*)
Boolean GetRoboVersionM (long *RobotID*, String *Data*)

[Arguments]

[IN] *RobotID* Specify the identifier (1 to 32) of the robot controller.

[OUT] *Data* The version information in the robot controller is set.

[Return value]

Returns the connection status with the robot controller.

1 : Being connected

0 : Not connected

[Description]

If the RoboVersion method is used in a multi-controller environment, it is the same as specifying "1" in *RobotID* of the GetRoboVersionM method.

[Example]

```
Dim lStatus As Long
Dim sData As String
lStatus = MelfaRxM.GetRoboVersionM(1, sData)
```

RoboLanguage
GetRoboLanguageM <Multi-Controller>

[Function]

Gets the information of the language currently set by a robot controller.

[Syntax]

Boolean RoboLanguage (String *Data*)
Boolean GetRoboLanguageM (long *RobotID*, String *Data*)

[Arguments]

[IN] *RobotID* Specify the identifier (1 to 32) of the robot controller.
[OUT] *Data* The information of the language currently set by the robot controller is set.

[Return value]

Returns the connection status with the robot controller.

1 : Being connected
0 : Not connected

[Description]

(1) The following language information is set in *Data*:
JPN: Japanese
ENG: English

(2) If the RoboLanguage method is used in a multi-controller environment, it is the same as specifying "1" in *RobotID* of the GetRoboLanguageM method.

[Example]

```
Dim lStatus As Long
Dim sData As String
lStatus = MelfaRxM.GetRoboLanguageM(1, sData)
```

RoboMechNo
GetRoboMechNoM <Multi-Controller>

[Function]

Gets the mechanical connection information set at the robot controller.

[Syntax]

Boolean RoboMechNo (long *Data*)
Boolean GetRoboMechNoM (long *RobotID*, long *Data*)

[Arguments]

[IN] *RobotID* Specify the identifier (1 to 32) of the robot controller.

[OUT] *Data* The mechanical connection information in the robot controller is set.

[Return value]

Returns the connection status with the robot controller.

1 : Being connected

0 : Not connected

[Description]

If the RoboMechNo method is used in a multi-controller environment, it is the same as specifying "1" in *RobotID* of the GetRoboMechNoM method.

[Example]

```
Dim lStatus As Long
Dim sData As String
lStatus = MelfaRxM.GetRoboMechNoM(1, sData)
```

RoboSlotCnt
GetRoboSlotCntM <Multi-Controller>

[Function]

Gets the number of slots in a robot controller.

[Syntax]

Boolean RoboSlotCnt (long *Data*)
Boolean GetRoboSlotCntM (long *RobotID*, long *Data*)

[Arguments]

[IN] *RobotID* Specify the identifier (1 to 32) of the robot controller.
[OUT] *Data* The number of slots in the robot controller is set.

[Return value]

Returns the connection status with the robot controller.

1 : Being connected
0 : Not connected

[Description]

If the RoboSlotCnt method is used in a multi-controller environment, it is the same as specifying "1" in *RobotID* of the GetRoboSlotCntM method.

[Example]

```
Dim lStatus As Long
Dim sData As String
lStatus = MelfaRxM.GetRoboSlotCntM(1, sData)
```

RoboCopyright
GetRoboCopyrightM <Multi-Controller>

[Function]

Gets the copyright information of a robot controller.

[Syntax]

Boolean RoboCopyright (String *Data*)
Boolean GetRoboCopyrightM (long *RobotID*, String *Data*)

[Arguments]

[IN] *RobotID* Specify the identifier (1 to 32) of the robot controller.

[OUT] *Data* The copyright information of the robot controller is set.

[Return value]

Returns the connection status with the robot controller.

1 : Being connected

0 : Not connected

[Description]

If the RoboCopyright method is used in a multi-controller environment, it is the same as specifying "1" in *RobotID* of the GetRoboCopyrightM method.

[Example]

```
Dim lStatus As Long
Dim sData As String
lStatus = MelfaRxM.GetRoboRoboCopyrightM(1, sData)
```

RoboComment ＜Reserved Future＞
GetRoboCommentM <Multi-Controller>＜Reserved Future＞

[Function]

Gets the comment string of a robot controller.

[Syntax]

Boolean RoboComment (String *Data*)
Boolean GetRoboCommentM (long *RobotID*, String *Data*)

[Arguments]

[IN] *RobotID* Specify the identifier (1 to 32) of the robot controller.
[OUT] *Data* The comment string of the robot controller is set.

[Return value]

Returns the connection status with the robot controller.

1 : Being connected
0 : Not connected

[Description]

If the RoboComment method is used in a multi-controller environment, it is the same as specifying "1" in *RobotID* of the GetRoboCommentM method.

Currently, both of these methods are reserved. "Null" is set in the comment string.

[Example]

```
Dim lStatus As Long
Dim sData As String
lStatus = MelfaRxM.GetRoboCommentM(1, sData)
```

5.3 Request IDs Specified by Request for Service Methods

This section describes the request IDs specified by request for service (RequestService, RequestServiceM, RequestService2Null, RequestService2NullM) methods.

If there is data to be received, use a get received data (GetRecvData, GetRecvDataM) method to get it after a reception complete event (MsgRecvEvent) has occurred.

⚠ Caution

When "Error L6010 (Illeagal comman)" was replied to transmitted ID, this request ID cannot be used in the connected robot contoller.

Table 5-5 : IDs Related to Editing

ID	Function name
100	Read robot program
101	Read command line
102	Read position data line
103	Write robot program (insert)
104	Write robot program (overwrite)
105	Rename robot program
106	Read robot program list
107	Copy robot program
108	Delete robot program
109	(Not used)
110	Step operation
111	Direct execution
112	Open and read robot program online
113	Close robot program online
114	Cancel online editing of robot program
115	Write one line of robot program online
116	Read one line of robot program online
117	Automatic online assignment of line numbers
118	Online renumbering of line numbers
119 to 121	(Not used)
122	Set attributes of robot program
123	Specify execution line number
124	Write command line
125	Open robot program online (open only)
126	Check existence of robot program
127	Read a robot program (The read content is specified)
128	Write potision data
129	Write external position data
130	Write instruction (command) lines and position variable
131	Write instruction (command) lines and external poison variable]
132	Read robot program list and the remainder memory
133	Read the specified part in the robot program
134 to 136	(Not used)
137	Insert one instruction (command) line into the robot program in edit slot
138	Read robot program list (all program language)
139	Set the Breakpoints
140	Release the Break points
141	Get the status of the Breakpoints
142	Open and read the robot program to the Editing slot (each kind of positional variable)
143 to 144	(Not used)

Table 5-6 : IDs Related to Monitoring

ID	Function name
200	Get slot operation status
201	Get mechanical operation status
202	Get stop signal status
203	Get currently occurring error
204	Get detailed error information
205	Get error log
206	Clear error log
207	(Not used)
208	Get execution line content of robot program
209	Get execution line number
210	Get current position/target position
211	Get variable list
212	Get variable value
213	Change variable value
214	Get I/O signal information (specify port number)
215	Forced output of I/O signal
216	Start general-purpose pseudo-input signal
217	End general-purpose pseudo-input signal
218	Write general-purpose pseudo-input signal
219	(Not used)
220	Servo Get ABS encoder information
221	Servo Get current information
222	Servo Get power supply information
223	Get total operation time
224	Clear total operation time
225	Get production information (cycle time)
226	(Not used)
227	Get mechanical connection information
228	(Not used)
229	Get hand open/close status
230	Get start date and time of error log aggregation
231	Get I/O signal information (specify signal number)
232	Servo Get position information
233	Servo Get speed information
234	Servo Get effective current information
235	Get current position
236	Get option card setting information
237	Get input register value <for CC-link>
238	Get output register value <for CC-link>
239	Write pseudo-input of input register <for CC-link>
240	Forced output of output register <for CC-link>
241	Get slot count of option card
242	Get slot operation status (simplified version)
243 to 254	(Not used)
255	Getting the current position in character string
256 to 264	(Not used)

Table 5-7 : IDs Related to Maintenance

ID	Function name
300	Get parameter list
301	Get parameter value (consider open level)
302	Write parameter value
303	(Not used)
304	Read file list
305	Read file
306	Write file
307	Set password
308	Get movement range information
309	Get slot information
310	Set slot information
311 to 314	(Not used)
315	Initialize all parameters
316	Get robot origin data
317	Write robot origin data
318	Get parameter value (do not consider open level)
319 to 326	(Not used)
327	Check writing of additional axis origin
328 to 342	(Not used)
343	Setting the origin data
344	Reading the origin data
345	(Not used)
346	Writing parameter(with the necessity of rebooting the robot controller)
347 to 349	(Not used)
350	Reading the serial number of robot and controller
351 to 362	(Not used)

Table 5-8 : IDs Related to Operation

ID	Function name
400	Execute program
401	Stop
402	(Not used)
403	Servo ON/OFF
404	Set speed override
405	Jog operation
406	Stop cycle
407	Reset error
408	Initialize slot
409	Reset program
410	Connect robot
411	Delete all robot programs
412	Enabling/Disabling operation right
413 to 414	(Not used)
415	Clear the task slots
416	Selecting the robot program
417	Reading the speed override of OP
418	Getting clock time
419	Setting clock time
420	(Not used)
421	Open / Close the robot hand
422 to 425	(Not used)
426	Change the override
427	The robot hand is lined up
428 to 431	(Not used)

* ID numbers indicated as "not used" cannot be used.

The following describes details:

Each item in each of the ID tables shows the following contents:

Outline : Outline of a request

Request ID : A numeric value to be specified in *MsgID* of the Request for service (RequestService) method, etc.

Argument when requesting : A string to be specified in *Data* of the request for service (RequestService) method, etc.

Reception data : A string to be set in *Data* of the get received data (GetRecvData) method, etc. when there is data to be received. The string ends with "NULL".

Each of arguments and reception data is one string in which elements are delimited by <LF>.

Example: Notation in table 1<LF>1<LF>10 M1=0

Example of coding:

```
Dim Data As String
Data = "1" + Chr$(10) + "1" + Chr$(10) + "10 M1=0"
```

5.3.1 Requests Related to Editing

* <LF> indicates the line feed character (0AH).

[Read robot program]

Item	Description
Overview	Read a robot program from the robot controller.
Request ID	100
Argument at request	Robot program name
Reception data	Robot program name<LF> Number of instruction lines<LF> Instruction on the first line<LF>the second line<LF> … the Nth line<LF> Number of position data lines<LF> Position data on the first line<LF>the second line<LF> … the Nth line
Remark	No extension is required for robot programs.
Example of argument	1
Example of reception data	1<LF> 100<LF> 10 DEF PLT 1,P1,P2,P3,P4,3,5,2<LF> 20 M1=1<LF> : 1000 END<LF> 10<LF> P1=(+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)<LF> P2=(+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)<LF> : P10=(+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)

[Read command line]

Item	Description
Overview	Read only the command lines of a robot program from the robot controller.
Request ID	101
Argument at request	Robot program name
Reception data	Robot program name<LF> Number of command lines<LF> Command on the first line<LF>the second line<LF> … the Nth line
Remark	No extension is required for robot programs.
Example of argument	1
Example of reception data	1<LF> 100<LF> 10 DEF PLT 1,P2,P3,P4,P5,3,5,2<LF> 20 M1=1<LF> : 1000 END

[Read position data Line]

Item	Description
Overview	Read only the position data lines of a robot program from the robot controller.
Request ID	102
Argument at request	Robot program name
Reception data	Robot program name<LF> Number of position data lines<LF> Position data on the first line<LF>the second line<LF> … the Nth line
Remark	No extension is required for robot programs.
Example of argument	1
Example of reception data	1<LF> 10<LF> P1=(+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)<LF> P2=(+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)<LF> : P10=(+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)

[Write robot program (insert)]

Item	Description
Overview	Write a robot program to the robot controller. If the specified program already exists in the robot controller, the program is inserted into the existing information.
Request ID	103
Argument at request	Robot program name<LF> Number of data items to be written<LF> Data on the first line<LF>the second line<LF> … the Nth line
Reception data	None
Remark	(1) No extension is required for robot programs. (2) There will not be any problems even if both instruction and position data lines are included among the data to be written.
Example of argument	1<LF> 110<LF> 10 DEF PLT 1,P1,P2,P3,P4,3,5,2<LF> 20 M1=1<LF> : 1000 END<LF> P1=(+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)<LF> P2=(+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)<LF> : P10=(+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)

[Write robot program (overwrite)]

Item	Description
Overview	Write a robot program to the robot controller. If the specified program already exists in the robot controller, the existing information is deleted first, and then the program is written.
Request ID	104
Argument at request	Robot program name<LF> Number of data items to be written<LF> Data on the first line<LF>the second line<LF> … the Nth line
Reception data	None
Remark	(1) No extension is required for robot programs. (2) There will not be any problems even if both instruction and position data lines are included among the data to be written.
Example of argument	1<LF> 110<LF> 10 DEF PLT 1,P1,P2,P3,P4,3,5,2<LF> 20 M1=1<LF> : P10=(+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)

[Rename robot program]

Item	Description
Overview	Change the name of the robot program in the robot controller.
Request ID	105
Argument at request	The robot program name to be changed<LF> The new robot program name
Reception data	None
Remark	No extension is required for robot programs.
Example of argument	1<LF> 10

[Read robot program list]

Item	Description
Overview	Read a list of robot programs in the robot controller.
Request ID	106
Argument at request	None
Reception data	Number of program information items<LF> Program information 1<LF>Program information 2<LF> … Program information N
Remark	(1) Details of program information: Robot program name<LF> Comment (NULL is set because this function is not currently supported.) <LF> Size<LF> Date<LF>Time<LF> Program attribute<LF> Number of instruction lines<LF> Number of position data lines<LF> Latest tact time [10 msec] <LF> Operation time [10 msec] <LF> Number of cycles<LF> Average tact time [10 msec] (2) The program attribute information is a numerical data value, where a decimal number is used to represent the binary data. The following information (1. Set protect, 0: Cancel protect) is set. Bit 0: Protect instructions Bit 1: Protect variables
Example of reception data	10<LF> 1.MB4<LF> <LF> ……NULL 921<LF> 02/01/08<LF> 18<LF> 15<LF> …… Both instructions and variables are protected. 100<LF> 10<LF> 0<LF> 0<LF> 0<LF> 0<LF> 2.MB4<LF> :

[Copy robot program]

Item	Description
Overview	Copy a robot program to another robot program in the robot controller.
Request ID	107
Argument at request	Copy source robot program name<LF> Copy destination robot program name
Reception data	None
Remark	No extension is required for robot programs.
Example of argument	1<LF> 10

[Delete robot program]

Item	Description
Overview	Delete a robot program in the robot controller.
Request ID	108
Argument at request	The name of the robot program to be deleted
Reception data	None
Remark	No extension is required for robot programs.
Example of argument	10

[Step operation]

Item	Description
Overview	Request step operation.
Request ID	110
Argument at request	Operation type
Reception data	None
Remark	(1) This request is made to the editing slot. See **8.1**, **"About Task Slots"** for the detailed explanation of slots. (2) Operation type: 0: Feed one line 1: Continuous feed 2: Step back one line 3: Stop (3) This request is only valid when the key switch of the robot controller is in the AUTO (Ext.) mode. (4) It is necessary to open the target robot program in the editing slot online before executing the step operation.
Example of argument	0

[Direct execution]

Item	Description
Overview	Request direct execution.
Request ID	111
Argument at request	Slot number<LF> Line to be executed
Reception data	None
Remark	(1) Specify the appropriate number (the number of slots + 1) for the slot number in order to send the request to the editing slot. See **8.1**, **"About Task Slots"** for the detailed explanation of slots. (2) This request is only valid when the key switch of the robot controller is in the AUTO (Ext.) mode. (3) It is necessary to turn the servo on beforehand in order to execute a line that operates the robot directly.
Example of argument	1<LF> M1=100

[Open and read robot program online]

Item	Description
Overview	Open a robot program in the editing slot and read the contents of the program.
Request ID	112
Argument at request	Robot program name
Reception data	Robot program name<LF> Number of instruction lines<LF> Instruction on the first line<LF>the second line<LF> … the Nth line<LF> Number of position data lines<LF> Position data on the first line<LF>the second line<LF> … the Nth line
Remark	(1) The robot program is kept open in the editing slot. See **8.1**, **"About Task Slots"** for the detailed explanation of slots. (2) No extension is required for robot programs.
Example of argument	1
Example of reception data	1<LF> 100<LF> 10 DEF PLT 1,P2,P3,P4,P5,3,5,2<LF> 20 M1=1<LF> : 1000 END<LF> 10<LF> P1=(+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00) <LF> P2=(+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00) <LF> : P10=(+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)

[Close robot program online]

Item	Description
Overview	Save the data edited in the robot program currently opened in the editing slot and close the robot program.
Request ID	113
Argument at request	None
Reception data	None
Remark	See **8.1**, **"About Task Slots"** for the detailed explanation of slots.

[Cancel online editing of robot program]

Item	Description
Overview	Cancel the editing of the data in the robot program currently opened in the editing slot and close the robot program (without saving).
Request ID	114
Argument at request	None
Reception data	None
Remark	See **8.1**, **"About Task Slots"** for the detailed explanation of slots.

[Write one line of robot program online]

Item	Description
Overview	Write one line to the robot program currently opened in the editing slot.
Request ID	115
Argument at request	An instruction or position data line
Reception data	None
Remark	It is necessary to open the robot program in the editing slot before this request can be executed. See **8.1**, **"About Task Slots"** for the detailed explanation of slots.
Example of argument	(In case of an instruction line) 50 OVRD 100 (In case of a position data line) P1=(100,200,300,400,500,600)(7,0)

[Read one line of robot program online]

Item	Description
Overview	Read one line from the robot program currently opened in the editing slot.
Request ID	116
Argument at request	Line number
Reception data	Instruction line
Remark	It is necessary to open the robot program in the editing slot before this request can be executed. See **8.1**, **"About Task Slots"** for the detailed explanation of slots.
Example of argument	50
Example of reception data	50 OVRD 100

[Automatic online assignment of line numbers]

Item	Description
Overview	Assign the line numbers of the robot program currently opened in the editing slot automatically. Specify the start line number and line interval the first time this request is executed, and retrieve the next line number assigned the second and subsequent times it is executed.
Request ID	117
Argument at request	Type<LF>Start line number<LF>Line interval
Reception data	The start line number (first time) or the next line number (second and subsequent times)
Remark	(1) Type: 0: Start line (first time) 1: Get the next line number (second and subsequent times) (2) Specify the start line number and line interval the first time this request is executed only. These values are ignored from the second time and onward. (3) See **8.1**, **"About Task Slots"** for the detailed explanation of slots.
Example of argument	(First time) 0<LF> 10<LF> 10 (Second time) 1<LF> <LF> …… NULL …… NULL
Example of reception data	(First time) 10 DEF PLT 1,P2,P3,P4,P5,3,5,2 (Second and subsequent times) 20

[Online renumbering of line numbers]

Item	Description
Overview	Renumber the robot program opened in the editing slot.
Request ID	118
Argument at request	Renumbering start line number<LF> Renumbering end line number<LF> New start line number<LF> Line interval
Reception data	None
Remark	(1) Specify 0 for the renumbering start line number in order to perform the renumbering from the first line. Specify 0 for the renumbering end line number in order to perform the renumbering until the last line. (2) See **8.1**, **"About Task Slots"** for the detailed explanation of slots.
Example of argument	0<LF> 0<LF> 100<LF> 10

[Set attributes of robot program]

Item	Description
Overview	Set the attribute of a robot program in the robot controller.
Request ID	122
Argument at request	Robot program name<LF> Program attribute
Reception data	None
Remark	(1) No extension is required for robot programs. (2) The program attribute information is a numerical data value, where a decimal number is used to represent the binary data. The following information (1. Set protect, 0: Cancel protect) is set. Bit 0: Protect instructions Bit 1: Protect variables
Example of argument	1<LF> 2 …… Only variables are protected.

[Specify execution line number]

Item	Description
Overview	Specify the line number of be executed.
Request ID	123
Argument at request	Slot number<LF> Line number
Reception data	(1) Specify the appropriate number (the number of slots + 1) for the slot number in order to send the request to the editing slot. See **8.1**, **"About Task Slots"** for the detailed explanation of slots. (2) It is necessary to turn the servo on beforehand if the line number of an execution line that operates the robot is specified.
Remark	Specify the line number of be executed.
Example of argument	1<LF> 10

[Write command line]

Item	Description
Overview	Write only command lines of a robot program to the robot controller. If the specified program already exists in the robot controller, only the command lines with the specified line numbers are deleted before writing.
Request ID	124
Argument at request	Robot program name<LF> Number of command lines<LF> Command on the first line<LF>the second line<LF> … the Nth line
Reception data	None
Remark	(1) If command lines are written to a robot program containing position data lines, those position data lines are not deleted. (2) No extension is required for robot programs.
Example of argument	1<LF> 100<LF> 10 DEF PLT 1,P1,P2,P3,P4,3,5,2<LF> 20 M1=1<LF> : 1000 END

[Open robot program online (open only)]

Item	Description
Overview	Open a robot program in the editing slot.
Request ID	125
Argument at request	Robot program name
Reception data	Robot program name
Remark	(1) The robot program is kept open in the editing slot. See **8.1**, **"About Task Slots"** for the detailed explanation of slots. (2) No extension is required for robot programs.
Example of argument	1
Example of reception data	1

[Check existence of robot program]

Item	Description
Overview	Check whether or not a specific robot program exists in the robot controller.
Request ID	126
Argument at request	Robot program name
Reception data	Result<LF> File name
Remark	(1) Result: 1: Yes 2: No (2) No extension is required for robot programs.
Example of argument	10
Example of reception data	1<LF> 10

[Read a robot program (The read content is specified)]

Item	Description
Overview	Read a robot program from robot controller. The read content (instruction(command) line, Local position, External positon) can be specified.
Request ID	127
Argument at request	Robot program name<LF> Read instruction(Command) lines (0/1=Not read/Read)<LF> Read position data (0/1=Not read/Read)<LF> Read external position data (0/1=Not read/Read)
Reception data	Robot program name<LF> Number of instruction lines<LF> Number of local position variable (XYZ)<LF> Number of local joint variable(Joint)<LF> Number of external position variable(XYZ)<LF> Number of external joint variable(Joint)<LF> Instruction on the first line<LF>the second line<LF> … the Nth line<LF> Local position variable on the first line<LF>…<LF>the Nthe line<LF> Loacl joint variable on the first line<LF>…<LF> the Nth line<LF> External position variable on the first line<LF>…<LF> the Nth line<LF> External joint variable on the first line<LF>…<LF> the Nth line
Remark	(1) The difference with ID 100 is to be able to specify the read content (instruction (command) line, local position, External positon). (2) No extension is required for robot programs.
Example of argument	2<LF> 1<LF> 0<LF> 1
Example of reception data	2<LF> 44<LF> 1 Mov P1<LF> 2 Mov P2<LF> 3 HOpen 1<LF> : 43 Mov P1<LF> 44 Mov P_11<LF> 0<LF> 0<LF> 1<LF> P_11=(+207.36,+372.46,+644.22,-180.00,-0.07,-94.39,+0.00,+0.00)(7,0) <LF> 0

[Write potision data]

Item	Description
Overview	Write positon variable to the robot controller.
Request ID	128
Argument at request	Robot program name<LF> Number of position variable<LF> Position variable on the first line<LF>…<LF>the Nthe line
Reception data	None
Remark	No extension is required for robot programs.
Example of argument	2<LF> 3<LF> P11=(+287.00,-314.50,+644.00,-180.00,-0.07,+157.17)(7,0)<LF> P12=(+187.00,-314.50,+344.00,-180.00,-0.07,+157.17)(7,0)<LF> P13=(+235.00,-314.50,+144.00,-180.00,-0.07,+157.17)(7,0)

[Write external position data]

Item	Description
Overview	Write external position data to the robot controller.
Request ID	129
Argument at request	Robot program name <LF> Number of external position variable <LF> External position variable on the first line<LF>…<LF> the Nth line
Reception data	None
Remark	No extension is required for robot programs.
Example of argument	2<LF> 1<LF> P_15=(+287.00,-314.50,+644.00,-180.00,-0.07,+157.17)(7,0)

[Write instruction (command) lines and position variable]

Item	Description
Overview	Write instruction (command) lines and position variable to the robot controller.
Request ID	130
Argument at request	Robot program name <LF> Number of instruction lines and position variable<LF> Instruction on the first line(instruction or position variable)<LF> the second line<LF> … the Nth line
Reception data	None
Remark	(1) The program is changed when the program already exists. (2) No extension is required for robot programs.
Example of argument	12<LF> 5<LF> 1 Mov P1<LF> 2 Mov P2<LF> 3 Mov P3<LF> P1=(+200.00,-300.00,+644.00,-180.00,-0.07,+157.17)(7,0) <LF> P2=(+120.00,-314.50,+200.00,-180.00,-0.07,+157.17)(7,0)

[Write instruction (command) lines and external poison variable]

Item	Description
Overview	Write instruction (command) lines and external position variable to the robot controller.
Request ID	131
Argument at request	Robot program name <LF> Number of instruction lines and external position variable<LF> Instruction on the first line(instruction or external position variable)<LF> the second line<LF> … the Nth line
Reception data	None
Remark	(1) The program is changed when the program already exists. (2) No extension is required for robot programs.
Example of argument	13<LF> 5<LF> 1 Mov P_7<LF> 2 Mov P_8<LF> 3 Mov P3<LF> P_7=(+200,-300,+.00,-90.00,-0.07,+157.17)(7,0) <LF> P_8=(+100,-314.50,+200,00,-0.07,+157.17)(7,0)

[Read robot program list and the remainder memory]

Item	Description
Overview	Read robot program list and the remainder memory from the robot controller. The difference with ID 106 is to read the remainder memory from the robot controller.
Request ID	132
Argument at request	None
Reception data	Number of program information items<LF> The remainder memory<LF> Program information 1<LF>Program information 2<LF> … Program information N
Remark	(1) Details of program information: Robot program name<LF> Comment (NULL is set because this function is not currently supported.) <LF> Size<LF> Date<LF>Time<LF> Program attribute<LF> Number of instruction lines<LF> Number of position data lines<LF> Latest tact time [10 msec] <LF> Operation time [10 msec] <LF> Number of cycles<LF> Average tact time [10 msec] (2) The program attribute information is a numerical data value, where a decimal number is used to represent the binary data. The following information (1. Set protect, 0: Cancel protect) is set. Bit 0: Protect instructions Bit 1: Protect variables
Example of reception data	2<LF> 104857600<LF> 1.MB5<LF> <LF> 360<LF> 08/10/08<LF>16:23:38<LF> 12<LF>6<LF>0<LF>0<LF>0<LF>0<LF>0<LF> 2.MB5<LF> <LF> 771<LF> 08/10/08<LF>16:23:26<LF> 12<LF>13<LF>6<LF>0<LF>0<LF>0<LF>0

[Read the specified part in the robot program]

Item	Description
Overview	Read the instruction (command) lines in the program that has been opened. The instruction (command) lines within the range specified by the argument is read.
Request ID	133
Argument at request	Type of reading position (0/1/2/3 = top/tail/step number/line number)LF> Reading position<LF> Number of instruction(command) line for reading
Reception data	Type of reading position (0/1/2/3 = top/tail/step number/line number)LF> Reading position<LF> Instruction on the first line<LF>the second line<LF> … the Nth line
Remark	
Example of argument	2<LF> 33<LF> 5
Example of reception data	1<LF> 5<LF> 33 Mvs P5<LF> 34 Dly 0.5<LF> 35 HOpen 1<LF> 36 Mvs P4<LF> 37 Mov P1

[Insert one instruction (command) line into the robot program in edit slot]

Item	Description
Overview	Insert one instruction (command) line into the robot program in edit slot
Request ID	137
Argument at request	Inserted step number<LF> Inserted instruction (command) line
Reception data	None
Remark	This ID can be used only in MELFA-BASIC V.
Example of argument	5<LF> Dly 0.2

[Read robot program list (all program language)]

Item	Description
Overview	Read robot program list (all program language(MMC,MB4,MB5))
Request ID	138
Argument at request	None
Reception data	Number of program information items<LF> The remainder memory<LF> Program information 1<LF>Program information 2<LF> … Program information N
Remark	*This ID can be used only in CR750/CRn-700 series controller. (1) Details of program information: Robot program name<LF> Comment (NULL is set because this function is not currently supported.) <LF> Size<LF> Date<LF>Time<LF> Program attribute<LF> Number of instruction lines<LF> Number of position data lines<LF> Latest tact time [10 msec] <LF> Operation time [10 msec] <LF> Number of cycles<LF> Average tact time [10 msec] (2) The program attribute information is a numerical data value, where a decimal number is used to represent the binary data. The following information (1. Set protect, 0: Cancel protect) is set. Bit 0: Protect instructions Bit 1: Protect variables
Example of reception data	2<LF> 104857600<LF> 1.MB5<LF> <LF> 360<LF> 08/10/08<LF>16:23:38<LF> 12<LF>6<LF>0<LF>0<LF>0<LF>0<LF>0<LF> 2.MB5<LF> <LF> 771<LF> 08/10/08<LF>16:23:26<LF> 12<LF>13<LF>6<LF>0<LF>0<LF>0<LF>0

[Set the Breakpoints]

Item	Description
Overview	Set the Breakpoint in the robot program opened for debugging.
Request ID	139
Argument at request	Number of the Breakpoints<LF> Robot program name <LF> Step number of the 1st Breakpoint<LF> Type of the 1st Breakpoint (0/1 = Permanent / One-time)<LF>…<LF> Step number of the Nth Breakpoint<LF> Type of the Nth Breakpoint (0/1 = Permanent / One-time)
Reception data	None
Remark	(1) No extension is required for robot programs. (2) This ID can be used only in MELFA-BASIC V. (3) Permanent : After stopping, the breakpoint keeps being set. One-time : After stopping, the breakpoint is automatically deleted at the same time as stopping.
Example of argument	2<LF> 2<LF> 5<LF>1<LF> 10<LF>0

[Release the Break points]

Item	Description
Overview	Release the Breakpoint in the robot program opened for debugging.
Request ID	140
Argument at request	Number of the released Breakpoint<LF> Robot program name <LF> Step number of the 1st Breakpoint<LF>…<LF> Step number of the Nth Breakpoint<LF>
Reception data	None
Remark	(1) No extension is required for robot programs. (2) This ID can be used only in MELFA-BASIC V.
Example of argument	1<LF> 2<LF> 3

[Get the status of the Breakpoints]

Item	Description
Overview	Get the status of the Breakpoints in the robot program opend for debugging.
Request ID	141
Argument at request	Robot program name <LF> Rerieval beginning line(NULL character is the top of the robot program)<LF> Retrieval end line(NULL character is the tail of the robot program)
Reception data	Number of the Breakpoint<LF> Robot programs of the 1st Breakpoint<LF> Step number of the 1st Breakpoint<LF> Type of the 1st Breakpoint (0/1 = Permanent / One-time)<LF>…<LF> Robot programs of the Nst Breakpoint<LF> Step number of the Nth Breakpoint<LF> Type of the Nth Breakpoint (0/1 = Permanent / One-time)
Remark	(1) No extension is required for robot programs. (2) This ID can be used only in MELFA-BASIC V.
Example of argument	2<LF> <LF>
Example of reception data	2<LF> 2<LF>3<LF>0<LF> 2<LF>7<LF>1

[Open and read the robot program to the Editing slot (each kind of positional variable)]

Item	Description
Overview	Open a robot program in the editing slot and read the contents of the program.
Request ID	142
Argument at request	Robot program name
Reception data	Robot program name <LF> Number of the instruction (command) lines<LF> Instruction on the first line<LF>the second line<LF> … the Nth line<LF> Number of position variable lines<LF> Position variable on the first line<LF>the second line<LF> … the Nth line Number of Joint variable<LF> Joint variable on the first line<LF>the second line<LF> … the Nth line
Remark	(1) No extension is required for robot programs. (2) The difference with ID 112 is to read Postion variable (XYZ) separately as Joint variable.
Example of argument	2
Example of reception data	2<LF> 43<LF> 1 Mov P1<LF> 2 Mov P2<LF> 3 HOpen 1<LF> : 41 HOpen 1<LF> 42 Mvs P4<LF> 43 Mov P1<LF> 16<LF> P1=(+287.78,-314.50,+644.62,-180.00,-0.07,+157.17)(7,0) <LF> P2=(+287.78,-314.50,+644.62,-180.00,-0.07,+157.17)(7,0) <LF> P3=(+287.78,-314.50,+189.98,-180.00,-0.07,+157.17)(7,0) <LF> P4=(+207.36,+372.46,+644.22,-180.00,-0.07,-94.39)(7,0) <LF> : P15=(+207.36,+372.46,+229.31,-180.00,-0.07,-94.39)(7,0) <LF> P0=(+400.13,+0.00,+644.62,+180.00,-0.07,+180.00)(7,0) <LF> 0

5.3.2 Requests Related to Monitoring

* <LF> indicates the line feed character (0AH).

Specify the reception interval when sending requests related to monitoring. Data is continuously received at the specified interval. Send a cancel request in order to end the monitoring while it is in progress.

[Get slot operation status]

Item	Description
Overview	Get the slot operation status.
Request ID	200
Argument at request	Slot number
Reception data	Slot number<LF> Slot status<LF> Robot program name<LF> Not used (0 is set.) <LF> Connected mechanism number<LF> Start condition<LF>Operation mode<LF> Task priority order<LF> Controlled mechanism number<LF> Operation status
Remark	(1) Specify the appropriate number (the number of slots + 1) for the slot number in order to send the request to the editing slot. See **8.1**, **"About Task Slots"** for the detailed explanation of slots. (2) The slot status information is a numerical data value, where a decimal number is used to represent the binary data. The following information is set for each of the bits (1: On, 0: Off). The least significant bit is bit 0 below. Bit 0: A program is opened in the editing slot Bit 1: Operating Bit 2: There is a change (set when the data in a program or the value of a variable in a program is changed.) (3) Start condition: REP: Continuous CYC: One cycle (4) Operation mode: START: Normal ERROR: At error occurrence ALWAYS: Always (5) Specify a value in the range from 1 to 31 for the task priority order. This number indicates the number of lines executed each time the task runs. It has the same meaning as the number of executed lines in the PRIORITY instruction of MELFA-BASIC IV. (6) The operation status information is a numerical data value, where a hexadecimal number is used to represent the binary data. The following values are set for each of the bits (1: On, 0: Off). The least significant bit is bit 0 below. Bit 0: During emergency stop Bit 1: Paused or stopped *1 Bit 2: Paused Bit 3: The stop signal is turned on Bit 4: Status where a program can be selected *2 Bit 5: Not used Bit 6: Not used Bit 7: Not used Bit 8: Cycle/continuous operation (0/1) Bit 9: Cycle being operated/stopped (0/1) Bit 10: Machine lock off/on (0/1) Bit 11: AUTO/TEACH mode (0/1) *3 Bit 12: A step operation or jog operation is being executed Bit 13: Servo on/off status (0/1) Bit 14: Stopped/operating (0/1) Bit 15: Invalid/valid operation right (0/1) *1 1 is set if a program is not executed. *2 Status where a program is not being executed. A program can be selected and executed from the start. *3 Status of the key switch. The bit is set to 0 if the key status is AUTO (OP or EXT) and 1 if the key status is TEACH.

Example of argument	1
Example of reception data	1<LF> 4<LF> 1.MB4<LF> 0<LF> 1<LF> START<LF> REP<LF> 1<LF> 0<LF> 112

[Get mechanical operation status]

Item	Description
Overview	Get the operation status of a mechanism.
Request ID	201
Argument at request	Mechanism number<LF>
Reception data	Mechanism number<LF> Machine lock status<LF> Open/close status of hand 1<LF>status of hand 2<LF> … status of hand 8<LF> Servo status<LF> Tip speed [mm/sec]
Remark	(1) Machine lock status: 0: Off 1: On (2) Open/close status of hand: 1: Open 2: Closed -1: No hand (3) Servo status: 0: Off 1: On
Example of argument	1
Example of reception data	1<LF> 0<LF> 1<LF>1<LF>1<LF>1<LF>-1<LF>-1<LF>-1<LF>-1<LF> 0<LF> 0.000

[Get stop signal status]

Item	Description
Overview	Get the status of the stop signals.
Request ID	202
Argument at request	None
Reception data	External stop signal status<LF> External emergency stop signal status
Remark	(1) The status information of the external stop signals is a numerical data value, where a hexadecimal number is used to represent the binary data. The following values are set for each of the bits (1: On, 0: Off). The least significant bit is bit 0 below. Bit 0: T/B Bit 1: PC Bit 2: I/O(I/O signal or CC-Link) Bit 3: O/P (2) The status information of the external emergency stop signals is a numerical data value, where a hexadecimal number is used to represent the binary data. The following values are set for each of the bits (1: On, 0: Off). The least significant bit is bit 0 below. Bit 0: I/O Bit 1: O/P Bit 2: T/B
Example of reception data	4<LF> 0

[Get currently occurring error]

Item	Description
Overview	Get the currently occurring error information.
Request ID	203
Argument at request	None
Reception data	Number of errors<LF> Error information 1<LF>Error information 2<LF> … Error information N
Remark	(1) Details of error information: Error number<LF> Error message<LF> Date<LF>Time<LF> Error level<LF> Robot program name<LF> Line number where the error occurred<LF> Detailed error number (2) Error level: 1: High 2: Low 3: Caution
Example of reception data	1<LF> 2602<LF> The target position is outside the operation range.<LF> 02-06-21<LF> 15:52:14<LF> 2<LF> 1.MB4<LF> 40<LF> 260200100<LF>

[Get detailed error information]

Item	Description
Overview	Get the detailed information of errors.
Request ID	204
Argument at request	Number of errors<LF> Error number 1<LF>Error number 2<LF> … Error number N
Reception data	Number of errors<LF> Error number 1<LF>Cause of error 1<LF>Method to recover from error 1<LF> Error number 2<LF>Cause of error 2<LF>Method to recover from error 2<LF> : Error number N<LF>Cause of error N<LF>Method to recover from error N
Remark	
Example of argument	1<LF> 2602
Example of reception data	1<LF> 2602<LF> The target position is outside the operation range.<LF> Adjust the position.

[Get error log]

Item	Description
Overview	Get error logs.
Request ID	205
Argument at request	Error level
Reception data	Error level<LF> Logging start date<LF>Time<LF> Number of errors<LF> Error information 1<LF>Error information 2<LF> … Error information N
Remark	(1) Error level: 0: All (when used as the argument at the request only) 1: High 2: Low 3: Caution (2) No error information details are set if the number of errors is 0. Details of error information: Error number<LF> Error message<LF> Date<LF>Time<LF> Error level<LF> Robot program name<LF> Line number where the error occurred<LF> Detailed error number
Example of argument	2
Example of reception data	2<LF> 01-05-30<LF> 18:47:58<LF> 7<LF> 2602<LF> The target position is outside the operation range.<LF> 02-06-21<LF> 15:52:14<LF> 2<LF> 1.MB4<LF> 40<LF> 260200100<LF> :

[Clear error log]

Item	Description
Overview	Clear error logs.
Request ID	206
Argument at request	Error level
Reception data	None
Remark	Error level: 0: All 1: High 2: Low 3: Caution
Example of argument	0

[Get execution line content of robot program]

Item	Description
Overview	Get the contents of lines in the program being executed. It is possible to get up to 16 lines, which must consist of the line currently being executed and lines before and after it.
Request ID	208
Argument at request	Slot number<LF> Number of lines before the line currently being executed to be acquired<LF> Number of lines after the line currently being executed to be acquired<LF>
Reception data	Slot number<LF> Robot program name<LF> Number of mechanism being used<LF> Start condition<LF> Task priority<LF> Line number being executed<LF> Number of lines read<LF> Program line 1<LF>Program line 2<LF> … Program line N
Remark	(1) Specify the appropriate number (the number of slots + 1) for the slot number in order to send the request to the editing slot. See **8.1**, **"About Task Slots"** for the detailed explanation of slots. (2) Only the line currently being executed is acquired if 0 is specified for the numbers of lines before and after the line currently being executed. (3) Start condition: START: Normal ERROR: At error ALWAYS: Always
Example of argument	1<LF> 1<LF> 1
Example of reception data	1<LF> 1.MB4<LF> 1<LF> START<LF> 1<LF> 50<LF> 3<LF> 40 P2=P_CURR<LF> 50 *LOOP<LF> 60 M1=M1+1

[Get execution line number]

Item	Description
Overview	Get the line number of the line being executed.
Request ID	209
Argument at request	Slot number
Reception data	Slot number<LF> Line number
Remark	Specify the appropriate number (the number of slots + 1) for the slot number in order to send the request to the editing slot. See **8.1**, **"About Task Slots"** for the detailed explanation of slots.
Example of argument	1
Example of reception data	1<LF> 70

[Get current position/target position]

Item	Description
Overview	Get the current position and the target position.
Request ID	210
Argument at request	Slot number (referenced when the mechanism number is not specified) <LF> Mechanism number (specify -1 in order to specify the slot number)
Reception data	Slot number<LF> Mechanism number<LF> Axis name of axis 1<LF>Axis2<LF> … Axis8<LF>Structure flag 1<LF>Structure flag 2<LF> Jog override<LF> Tip speed [mm/sec] <LF> Overrun limit status<LF> Current position of axis 1 (in orthogonal coordinates)<LF>Axis2<LF> … Axis8<LF> Axis name of axis 1<LF>Axis2<LF> … Axis8<LF>Structure flag 1<LF>Structure flag 2<LF> Jog override<LF> Tip speed [mm/sec] <LF> Overrun limit status<LF> Current position of axis 1 (in joint coordinates)<LF>Axis 2<LF> … Axis 8<LF> Axis name of axis 1<LF>Axis2<LF> … Axis8<LF>Structure flag 1<LF>Structure flag 2<LF> Jog override<LF> Tip speed [mm/sec] <LF> Overrun limit status<LF> Target position of axis 1 (in orthogonal coordinates)<LF>Axis2<LF> … Axis8<LF> Axis name of axis 1<LF>Axis2<LF> … Axis8<LF>Structure flag 1<LF>Structure flag 2<LF> Jog override<LF> Tip speed [mm/sec] <LF> Overrun limit status<LF> Target position of axis 1 (in joint coordinates)<LF>Axis2<LF> … Axis 8
Remark	(1) Specify either a slot number or a mechanism number for the request argument. The mechanism number is referenced if a value is specified for the mechanism number; in this case, the slot number is ignored. The slot number is referenced if -1 is specified for the mechanism number. See **8.1**, **"About Task Slots"** for the detailed explanation of slots. (2) Specify the appropriate number (the number of slots + 1) for the slot number in order to send the request to the editing slot. (3) Axis name: Orthogonal coordinates : X, Y, Z, A, B, C, L1, L2 Joint coordinates : J1, J2, J3, J4, J5, J6, J7, J8 NULL is set for non-existing axes. (4) The overrun limit status information is a numerical data value, where a hexadecimal number is used to represent the binary data indicating whether or not each axis is within the movement range. The following values are set for each of the bits (1: Within the movement range, 0: Outside the movement range). The least significant bit is bit 0 below. Bits 0 to 15 : Not used (0 is set.) Bits 16 to 23 : States of the - directions of J1 to J8 Bits 24 to 31 : States of the + directions of J1 to J8
Example of argument	1<LF> -1

Example of reception data	1<LF> -1<LF> X<LF>Y<LF>Z<LF>A<LF>B<LF>C<LF><LF><LF> 0<LF>0<LF> 50.000<LF> 0.000<LF> 0<LF> 1451.200<LF>0.200<LF>1651.490<LF>0.000<LF>0.000<LF>0.000<LF>0.000<LF>0.000<LF> J1<LF>J2<LF>J3<LF>J4<LF><LF><LF><LF><LF> 0<LF>0<LF> 50.000<LF> 0.000<LF> 0<LF> 0.000<LF>0.010<LF>0.150<LF>0.000<LF>0.000<LF>0.000<LF>0.000<LF>0.000<LF> :

[Get variable list]

Item	Description
Overview	Get a list of variables.
Request ID	211
Argument at request	Slot number<LF> Variable type
Reception data	Slot number<LF> Variable type<LF> Number of variables<LF> Variable name 1<LF>Variable name 2<LF> … Variable name N
Remark	(1) Specify the appropriate number (the number of slots + 1) for the slot number in order to send the request to the editing slot. See **8.1**, **"About Task Slots"** for the detailed explanation of slots. (2) Variable type 0: Integer type, single/double precision real number type 1: Character string type 2: Position type 3: Joint type
Example of argument	1<LF> 2
Example of reception data	1<LF> 2<LF> 2<LF> P1<LF>P2

[Get variable value]

Item	Description
Overview	Get the values of variables.
Request ID	212
Argument at request	Slot number<LF> Number of variables<LF> Variable name 1<LF>Variable type 1<LF> Variable name 2<LF>Variable type 2<LF> : Variable name N<LF>Variable type N
Reception data	Slot number<LF> Number of variables<LF> Variable information 1<LF>Variable information 2<LF> … Variable information N
Remark	(1) Specify the appropriate number (the number of slots + 1) for the slot number in order to send the request to the editing slot. See **8.1**, **"About Task Slots"** for the detailed explanation of slots. (2) Variable type 0: Integer type, single/double precision real number type 1: Character string type 2: Position type 3: Joint type (3) The definition of a variable is set for the variable information. See the example of reception data.
Example of argument	1<LF> 2<LF> M1<LF>0<LF> P1<LF>2
Example of reception data	1<LF> 2<LF> M1=+15<LF> P1=(+1451.21,+0.20,+1651.48,+0.00,+0.00,+0.00)(0,0)

[Change variable value]

Item	Description
Overview	Change the values of variables.
Request ID	213
Argument at request	Robot program name<LF> Number of variables<LF> Variable information 1<LF>Variable information 2<LF> … Variable information N
Reception data	None
Remark	(1) No extension is required for robot programs. (2) Specify the definition of a variable for the variable information.
Example of argument	1<LF> 2<LF> M1=10<LF> P1=(100,200,300,0,0,0)(0,0)

[Get I/O signal information (specify port number)]

Item	Description
Overview	Specify the port name and get the input/output signal information.
Request ID	214
Argument at request	Number of input ports<LF> First port number<LF>Second port number<LF> … Mth port number<LF> Number of output ports<LF> First port number<LF>Second port number<LF> … Nth port number
Reception data	Number of input ports<LF> First port number<LF>First port input information<LF> Second port number<LF>Second port input information<LF> : Mth port number<LF>Mth port input information<LF> Number of output ports<LF> First port number<LF>First port output information<LF> Second port number<LF>Second port output information<LF> : Nth port number<LF>Nth port output information<LF>
Remark	(1) A port number represents a group of 16 signal number bits. Port numbers correspond to signal numbers as follows. Port numbers start from 0. Port number — Signal number 0 — 0 to 15 1 — 16 to 31 : — : (2) The port input/output information consists of numerical data values, where hexadecimal numbers are used to represent the 16-bit binary data of individual ports.
Example of argument	2<LF> 0<LF>2<LF> 2<LF> 1<LF>3
Example of reception data	2<LF> 0<LF>0018<LF> 2<LF>0000<LF> 2<LF> 1<LF>0002<LF> 3<LF>0000

[Forced output of I/O signal]

Item	Description
Overview	Request to output a 16-bit signal forcefully.
Request ID	215
Argument at request	The head signal number<LF> Output information
Reception data	None
Remark	(1) Signal numbers start from 0. (2) The output information is a numerical data value, where a hexadecimal number is used to represent the 16-bit binary data to be output.
Example of argument	0<LF> 1800

[Start general-purpose pseudo-input signal]

Item	Description
Overview	Start the operation of a general-purpose pseudo-input signal. It blocks input/output from the outside. This operation is used for debugging purposes.
Request ID	216
Argument at request	None
Reception data	None
Remark	Send this request when a general-purpose pseudo-input signal should be started.

[End general-purpose pseudo-input signal]

Item	Description
Overview	End the operation of a general-purpose pseudo-input signal. When this request is sent, the robot controller cancel error 6800, "Cancel the pseudo-input state. Turn OFF the power." occurs. Turn the power supply of the controller off and on again in order to restart input/output from outside. This operation is used for debugging purposes.
Request ID	217
Argument at request	None
Reception data	None
Remark	Send this request when a general-purpose pseudo-input signal should be ended.

[Write general-purpose pseudo-input signal]

Item	Description
Overview	Write a general-purpose pseudo-input signal. This operation is used for debugging purposes.
Request ID	218
Argument at request	The head signal number<LF> Input information
Reception data	None
Remark	The input information is a numerical data value, where a hexadecimal number is used to represent the 16-bit binary input data.
Example of argument	0<LF> 1800

[Servo Get ABS encoder information]

Item	Description
Overview	Get the information of the ABS encoder.
Request ID	220
Argument at request	Mechanism number
Reception data	Mechanism number<LF> ABS encoder current value [pulses] J1<LF>J2<LF> … J8<LF> Data in one rotation [pulses] J1<LF>J2<LF> … J8<LF> Position droop [pulses] J1<LF>J2<LF> … J8<LF> Maximum position droop [pulses] J1<LF>J2<LF> … J8<LF> Current speed [pulses] J1<LF>J2<LF> … J8<LF> Speed maximum J1<LF>J2<LF> … J8
Remark	
Example of argument	1
Example of reception data	1<LF> -386895<LF>-103483<LF>-431775<LF>-852516<LF>790304<LF>145568<LF>0<LF>0<LF> 1873<LF>5162<LF>5810<LF>549<LF>4302<LF>1889<LF>0<LF>0<LF> 1<LF>491<LF>-369<LF>1<LF>409<LF>11<LF>0<LF>0<LF> 433<LF>-2035<LF>4927<LF>-62<LF>-3841<LF>-376<LF>0<LF>0 0<LF>101<LF>-80<LF>0<LF>86<LF>0<LF>0<LF>0<LF> -95<LF>-458<LF>1095<LF>11<LF>-865<LF>86<LF>0<LF>0

[Servo Get current information]

Item	Description
Overview	Get the current information of the servo.
Request ID	221
Argument at request	Mechanism number
Reception data	Mechanism number<LF> Current feedback [arms] J1<LF>J2<LF> … J8<LF> Maximum current (past 2 seconds) J1<LF>J2<LF> … J8<LF> Maximum current (when the power supply was turned on) J1<LF>J2<LF> … J8<LF> Axis load level [Alarm %] J1<LF>J2<LF> … J8<LF> Maximum axis level [Alarm %] J1<LF>J2<LF> … J8
Remark	
Example of argument	1
Example of reception data	1<LF> -0.030<LF>-0.160<LF>0.080<LF>0.010<LF>0.020<LF>0.020<LF>0.000<LF>0.000<LF> 0.110<LF>0.260<LF>0.190<LF>0.020<LF>0.040<LF>0.030<LF>0.000<LF>0.000<LF> 0.120<LF>0.320<LF>0.300<LF>0.030<LF>0.070<LF>0.030<LF>0.000<LF>0.000<LF> 0.000<LF>7.000<LF>2.000<LF>0.000<LF>2.000<LF>0.000<LF>0.000<LF>0.000<LF> 0.000<LF>7.000<LF>3.000<LF>0.000<LF>2.000<LF>0.000<LF>0.000<LF>0.000

[Servo Get power supply information]

Item	Description
Overview	Get the power supply information of the servo.
Request ID	222
Argument at request	Mechanism number
Reception data	Mechanism number<LF> Power supply voltage [V] <LF> Maximum power supply voltage [V] <LF> Minimum power supply voltage [V] <LF> Regeneration level [Alarm level %] J1<LF>J2<LF> … J8
Remark	
Example of argument	1
Example of reception data	1<LF> 285<LF> 289<LF> 282<LF> 0<LF>0<LF>0<LF>0<LF>0<LF>0<LF>0<LF>0

[Get total operation time]

Item	Description
Overview	Get the total operation time.
Request ID	223
Argument at request	None
Reception data	Date<LF>Current time<LF> Power-on time [hours] <LF> Mechanical connection information<LF> Servo-on time of mechanism 1 [hours] <LF>Mechanism2<LF> … Mechanism16<LF> Operation time [hours] <LF> Not used (0 is set.)<LF> Battery usage time [hours] <LF> Estimated battery replacement time [hours]
Remark	The mechanical connection information is a numerical data value, where a hexadecimal number is used to represent the binary data. The connection states of mechanism 1, 2, and so forth are set starting from bit 0.
Example of reception data	02-06-24<LF>15:36:16<LF> 32<LF> 1<LF> 25<LF>25<LF> … 0<LF> 21<LF> 0<LF> 842<LF> 13758

[Clear total operation time]

Item	Description
Overview	Clear the total operation time.
Request ID	224
Argument at request	Clear type
Reception data	None
Remark	Clear type 0: All 1: Power-on time 2: Motor-on time 3: Operation time 4: Program unit 5: Battery connection time
Example of argument	0

[Get production information (cycle time)]

Item	Description
Overview	Get the production information (cycle time) of a robot program.
Request ID	225
Argument at request	Robot program name
Reception data	Robot program name<LF> Latest tact time [10 msec] <LF> Operation time [10 msec] <LF> Number of cycles<LF> Average tact time [10 msec]
Remark	No extension is required for robot programs.
Example of argument	1
Example of reception data	1<LF> 43235<LF> 366646<LF> 9<LF> 40738

[Get mechanical connection information]

Item	Description
Overview	Get the mechanical connection information.
Request ID	227
Argument at request	None
Reception data	Number of connected mechanisms<LF> Mechanism information 1<LF>Mechanism2<LF> … Mechanism N
Remark	(1) Details of mechanism information: Mechanism number<LF> Orthogonal axis pattern<LF> Joint axis pattern<LF> Structure flag 1<LF>Structure flag 2<LF> Jog override<LF> Mechanism name<LF> Parameter file name<LF> Servo status<LF> Mechanism lock status (2) Mechanism number 0 indicates the common mechanism information. The common mechanism information refers to information common to all robots; 0 is set for all the information items other than the parameter file name. (3) The axis pattern information is a numerical data value, where a decimal number is used to represent the binary data indicating whether each axis exists or not. The states of existence of X, Y, and so on are set starting from bit 0 in case of orthogonal axes. The states of existence of J1, J2, and so on are set starting from bit 0 in case of joint axes. (4) Servo status: 0: Off 1: On (5) Mechanism lock status: 0: Off 1: On
Example of reception data	2<LF> 0<LF> ……. Common mechanism information1 0<LF> 0<LF> 0<LF>0<LF> 0.000<LF> 0<LF> COMMON.LST<LF> 0<LF> 0<LF> 1<LF> ……. Mechanism information 1 39<LF> 15<LF> 0<LF>0<LF> 100.000<LF> RV-100TH<LF> RV-100TH.LST<LF> 0<LF> 0<LF>

[Get hand open/close status]

Item	Description
Overview	Get the open/close states of hands.
Request ID	229
Argument at request	Mechanism number
Reception data	Mechanism number<LF> Open/close status of hand 1<LF>Solenoid type<LF>Signal number<LF> Open/close status of hand 2<LF>Solenoid type<LF>Signal number<LF> : Open/close status of hand 8<LF>Solenoid type<LF>Signal number
Remark	(1) Open/close status of hand: 1: Open 0: Close -1: No hand (2) Solenoid type: 0: Single solenoid 1: Double solenoid -1: No hand (3) The head signal number is set for the hand signal number in case of double solenoids. -1 is set when there are no hands.
Example of argument	1
Example of reception data	1<LF> 1<LF>1<LF>900<LF> 1<LF>1<LF>902<LF> 1<LF>1<LF>904<LF> 1<LF>1<LF>906<LF> -1<LF>-1<LF>-1<LF> -1<LF>-1<LF>-1<LF> -1<LF>-1<LF>-1<LF> -1<LF>-1<LF>-1

[Get start date and time of error log aggregation]

Item	Description
Overview	Get the date and time when aggregation of error logs was started.
Request ID	230
Argument at request	Aggregation type
Reception data	Aggregation type<LF> Aggregation start date<LF> Aggregation start time
Remark	Aggregation type: 0: Error logs
Example of argument	0
Example of reception data	0<LF> 01-05-30<LF> 18:47:58

[Get I/O signal information (specify signal number)]

Item	Description
Overview	Get the 16-bit input/output signal information from the specified signal number.
Request ID	231
Argument at request	Number of input signals<LF> Head signal number of the first port<LF>the second port<LF> … the Mth port<LF> Number of output signals<LF> Head signal number of the first port<LF>the second port<LF> … the Nth port
Reception data	Number of input signals<LF> Head signal number of the first port<LF>Input status of the first port<LF> Head signal number of the second port<LF>Input status of the second port<LF> : Head signal number of the Mth port<LF>Input status of the Mth port<LF> Number of output signals<LF> Head signal number of the first port<LF>Output status of the first port<LF> Head signal number of the second port<LF>Output status of the second port<LF> : Head signal number of the Nth port<LF>Output status of the Nth port
Remark	(1) Signal numbers start from 0. (2) The input/output status information consists of numerical data values, where hexadecimal numbers are used to represent the 16-bit binary data.
Example of argument	2<LF> 0<LF>10 1<LF> 0
Example of reception data	2<LF> 0<LF>0018<LF> 10<LF>0000<LF> 1<LF> 0<LF>0002

[Servo Get position information]

Item	Description
Overview	Get the position information of the servo.
Request ID	232
Argument at request	Mechanism number
Reception data	Mechanism number<LF> Fdt command [pulses/IT] J1<LF>J2<LF> … J8<LF> Position command [pulses] J1<LF>J2<LF> … J8
Remark	
Example of argument	1
Example of reception data	1<LF> 0<LF>111<LF>167<LF>0<LF>-145<LF>1<LF>0<LF>0<LF> 1081<LF>179961<LF>398678<LF>-530<LF>86960<LF>120800<LF>0<LF>0

[Servo Get speed information]

Item	Description
Overview	Get the speed information of the servo.
Request ID	233
Argument at request	Mechanism number
Reception data	Mechanism number<LF> Speed feedback [rpm] J1<LF>J2<LF> … J8<LF> Maximum speed [rpm] J1<LF>J2<LF> … J8<LF> Speed command [rpm] J1<LF>J2<LF> … J8
Remark	
Example of argument	1
Example of reception data	1<LF> 0<LF>0<LF>2<LF>0<LF>0<LF>0<LF>0<LF>0<LF> 101<LF>-761<LF>1095<LF>-13<LF>-901<LF>-95<LF>0<LF>0<LF> 0<LF>1<LF>0<LF>0<LF>0<LF>0<LF>0<LF>0

[Servo Get effective current information]

Item	Description
Overview	Get the effective current information of the servo.
Request ID	234
Argument at request	Mechanism number
Reception data	Mechanism number<LF> Current command [Arms] J1<LF>J2<LF> … J8<LF> Allowable current command on the + side [Arms] J1<LF>J2<LF> … J8<LF> Allowable current command on the - side [Arms] J1<LF>J2<LF> … J8<LF> Effective current [Arms] J1<LF>J2<LF> … J8
Remark	
Example of argument	1
Example of reception data	1<LF> 0.090<LF>-0.220<LF>-0.010<LF>0.020<LF>0.020<LF>0.020<LF>0.000<LF>0.000<LF> 1.080<LF>0.870<LF>0.900<LF>0.290<LF>0.290<LF>0.260<LF>0.000<LF>0.000<LF> -1.080<LF>-0.870<LF>-0.900<LF>-0.290<LF>-0.290<LF>-0.260<LF>0.000<LF>0.000<LF> 0.050<LF>0.160<LF>0.090<LF>0.010<LF>0.030<LF>0.020<LF>0.000<LF>0.000

[Get current position]

Item	Description
Overview	Get the current position.
Request ID	235
Argument at request	Slot number (referenced when the mechanism number is not specified) <LF> Mechanism number (specify -1 in order to specify the slot number) <LF> Coordinate type
Reception data	Slot number<LF> Mechanism number<LF> Coordinate type<LF> Axis name of axis 1<LF>Axis name of axis 2<LF> … Axis name of axis 8<LF> Structure flag 1<LF>Structure flag 2<LF> Jog override<LF> Tip speed [mm/sec] <LF> Overrun limit status<LF> Current position of axis 1<LF>Axis2<LF> … Axis8<LF>
Remark	(1) Specify either a slot number or a mechanism number for the request argument. The mechanism number is referenced if a value is specified for the mechanism number; in this case, the slot number is ignored. The slot number is referenced if -1 is specified for the mechanism number. See **8.1**, **"About Task Slots"** for the detailed explanation of slots. (2) Specify the appropriate number (the number of slots + 1) for the slot number in order to send the request to the editing slot. (3) Specify the coordinate system of the acquired position via the coordinate type. 0: Joint 1: Orthogonal (4) The overrun limit status information is a numerical data value, where a hexadecimal number is used to represent the binary data indicating whether or not each axis is within the movement range. The following values are set for each of the bits (1: Within the movement range, 0: Outside the movement range). The least significant bit is bit 0 below. Bits 0 to 15: Not used (0 is set.) Bits 16 to 23: States of the - directions of J1 to J8 Bits 24 to 31: States of the + directions of J1 to J8
Example of argument	1<LF> -1<LF> 1
Example of reception data	1<LF> -1<LF> 1<LF> X<LF>Y<LF>Z<LF>A<LF>B<LF>C<LF><LF><LF>0<LF>0<LF> 50.000<LF> 0.000<LF> 0<LF> 1451.220<LF>0.200<LF>1651.470<LF>0.000<LF>0.000<LF>0.000<LF>0.000<LF>0.000

[Get option card setting information]

Item	Description
Overview	Get the option card setting information.
Request ID	236
Argument at request	Slot number in which a card is mounted
Reception data	Slot number in which a card is mounted<LF> Type of card<LF> Option card status 1<LF>Option card status 2<LF> … Operation card status N
Remark	(1) Slot numbers in which a card is mounted begin from 1. (2) The name of the card mounted is set for the type of card as a character string. (3) The amount of option card status information varies depending on the type of the mounted card.
Example of argument	1
Example of reception data	1<LF> Ethernet<LF> IP Addr 192.168.0.1<LF> PortNo(EZSocket) 10001<LF> :

[Get input register value]

Item	Description
Overview	Get the values of batches of 16 input registers from the specified register number (for CC-Link).
Request ID	237
Argument at request	Number of registers<LF> Head register number 1<LF> … Head register number N
Reception data	Number of registers<LF> Head register number 1<LF> Register value 1-1<LF>Register value 1-2<LF>…Register value 1-16<LF> : Head register number N<LF> Register value N-1<LF>Register value N-2<LF> … Register value N-16
Remark	
Example of argument	2<LF> 6000<LF> 6016
Example of reception data	2<LF> 6000<LF> 100<LF>200<LF> … 300<LF> 6016<LF> 100<LF>200<LF> … 300

[Get output register value]

Item	Description
Overview	Get the values of output registers (for CC-Link).
Request ID	238
Argument at request	Number of registers<LF> Head register number 1<LF> … Head register number N
Reception data	Number of registers<LF> Head register number 1<LF> Register value 1-1<LF>Register value 1-2<LF> … Register value 1-16<LF> : Head register number N<LF> Register value N-1<LF>Register value N-2<LF> … Register value N-16
Remark	
Example of argument	2<LF> 6000<LF> 6016
Example of reception data	2<LF> 6000<LF> 100<LF>200<LF> … 300<LF> 6016<LF> 100<LF>200<LF> … 300

[Write pseudo-input of input register]

Item	Description
Overview	Request to write pseudo-inputs to input registers (for CC-Link). This operation is used for debugging purposes.
Request ID	239
Argument at request	Number of registers to be written<LF> Register number 1<LF>Register value 1<LF> Register number 2<LF>Register value 2<LF> : Register number N<LF>Register value N
Reception data	Number of registers written<LF> Register number 1<LF>Register value 1<LF> Register number 2<LF>Register value 2<LF> : Register number N<LF>Register value N
Remark	The response data returns the values written.
Example of argument	2<LF> 6000<LF>100<LF> 6001<LF>200
Example of reception data	2<LF> 6000<LF>100<LF> 6001<LF>200

[Forced output of output register]

Item	Description
Overview	Request to forcefully output registers (for CC-Link). This operation is used for debugging purposes.
Request ID	240
Argument at request	Number of registers to be output<LF> Register number 1<LF>Register value 1<LF> Register number 2<LF>Register value 2<LF> : Register number N<LF>Register value N
Reception data	Number of registers output<LF> Register number 1<LF>Register value 1<LF> Register number 2<LF>Register value 2<LF> : Register number N<LF>Register value N
Remark	The response data returns the current values.
Example of argument	2<LF> 6000<LF>100<LF> 6001<LF>200
Example of reception data	2<LF> 6000<LF>100<LF> 6001<LF>200

[Get slot count of option card]

Item	Description
Overview	Get the number of slots in which option cards are mounted.
Request ID	241
Argument at request	None
Reception data	Number of slots
Remark	
Example of reception data	3

[Get slot operation status (simplified version)]

Item	Description
Overview	Get the slot operation status. This is a simplified version.
Request ID	242
Argument at request	Slot number
Reception data	Slot number<LF>Robot program name<LF>Execution line number<LF>Override<LF>Slot status<LF>Operation status<LF>Error numbers currently generated
Remark	(1) Specify the appropriate number (the number of slots + 1) for the slot number in order to send the request to the editing slot. See **8.1**, **"About Task Slots"** for the detailed explanation of slots. (2) The slot status information is a numerical data value, where a decimal number is used to represent the binary data. The following information is set for each of the bits (1: On, 0: Off). The least significant bit is bit 0 below. Bit 0: A program is opened in the editing slot Bit 1: Operating Bit 2: There is a change (set when the data in a program or the value of a variable in a program is changed.) (3) The operation status information is a numerical data value, where a hexadecimal number is used to represent the binary data. The following values are set for each of the bits (1: On, 0: Off). The least significant bit is bit 0 below. Bit 0: During emergency stop Bit 1: Paused or stopped *1 Bit 2: Paused Bit 3: The stop signal is turned on Bit 4: Status where a program can be selected *2 Bit 5: Not used Bit 6: Not used Bit 7: Not used Bit 8: Cycle/continuous operation (0/1) Bit 9: Cycle being operated/stopped (0/1) Bit 10: Machine lock off/on (0/1) Bit 11: AUTO/TEACH mode (0/1) *3 Bit 12: A step operation or jog operation is being executed Bit 13: Servo on/off status (0/1) Bit 14: Stopped/operating (0/1) Bit 15: Invalid/valid operation right (0/1) *1 1 is set if a program is not executed. *2 Status where a program is not being executed. A program can be selected and executed from the start. *3 Status of the key switch. The bit is set to 0 if the key status is AUTO (OP or EXT) and 1 if the key status is TEACH. (4) 0 is set for the error numbers currently generated if there are no generated errors currently.
Example of argument	1
Example of reception data	1<LF> 1.MB4<LF> 10<LF> 100<LF> 4<LF> 112<LF> 0

[Getting the current position in character string]

Item	Description
Overview	Getting the current position in character string.
Request ID	255
Argument at request	Mechanism number<LF>Coordinates type(0/1 =Joint/XYZ)
Reception data	Mechanism number<LF> Coordinates type (0/1 = Joint/XYZ)<LF> Name of axis 1<LF>…<LF>Name of axis 8<LF> Structure flag(FL1)<LF>Structure flag(FL2)<LF> Jog override<LF> Tip speed[mm/sec]<LF> Overrun limit status<LF> Current position 1<LF>…<LF>Current position 8
Remark	The difference with ID 235 is to be set '*' in the axis to which the origin data is not set.
Example of argument	1<LF> 1
Example of reception data	1<LF> 1<LF> X<LF>Y<LF>Z<LF>A<LF>B<LF>C<LF><LF><LF> 7<LF>0<LF> 100.000<LF> 0.000<LF> 0<LF> 400.13<LF> 0.00<LF> 644.62<LF> 180.00<LF> -0.07<LF> 180.00<LF> <LF>

5.3.3 Requests Related to Maintenance

* <LF> indicates the line feed character (0AH).

[Get parameter list]

Item	Description
Overview	Get a parameter list.
Request ID	300
Argument at request	Mechanism number
Reception data	Mechanism number<LF> Parameter file name<LF> Number of parameter information items<LF> Parameter information 1<LF>Parameter 2<LF> … ParameterN
Remark	The parameter information is set in the following format: "Parameter name" = "user access level" ; "description" ; "message" ; *
Example of argument	1
Example of reception data	1<LF> RV-100TH.LST<LF> 529<LF> NAME=MS;Table name;No message;*<LF> MEPSEL=Ms;5-character parameter name to be displayed at the selection of model;No message;*<LF> :

[Get parameter value (consider open level)]

Item	Description
Overview	Get the values of the specified parameters. If a password is not entered, only parameters at the user access level can be read.
Request ID	301
Argument at request	Mechanism number<LF> Number of parameters<LF> Parameter name 1<LF>Parameter name 2<LF> … Parameter name N
Reception data	Mechanism number<LF> Number of parameters<LF> Parameter name 1<LF>Number of data items 1<LF>Not used (0 is set.)<LF>Data 1<LF> Parameter name 2<LF>Number of data items 2<LF>Not used<LF>Data 2<LF> : Parameter name N<LF>Number of data items N<LF>Not used<LF>Data N
Remark	The number of elements in the parameter data is set for the number of data items.
Example of argument	1<LF> 2<LF> SLT1<LF> SLT2
Example of reception data	1<LF> 2<LF> SLT1<LF> 4<LF> 0<LF> 1, REP, START, 1<LF> SLT2<LF> 4<LF> 0<LF> 90, REP, ALWAYS, 1

[Write parameter value]

Item	Description
Overview	Write the values of the specified parameters to the robot controller.
Request ID	302
Argument at request	Mechanism number<LF> Number of parameters<LF> Parameter name 1<LF>Data 1<LF> Parameter name 2<LF>Data 2<LF> : Parameter name N<LF>Data N
Reception data	None
Remark	
Example of argument	1<LF> 2<LF> SLT1<LF> 2, REP, START, 1<LF> SLT2<LF> 2, REP, ALWAYS, 1

[Read file list]

Item	Description
Overview	Read a list of files with a specific extension from the robot controller.
Request ID	304
Argument at request	Extension (a period is required.)
Reception data	Number of files<LF> File information 1<LF>File information 2<LF> … File information N
Remark	(1) The period of the extension is required. Specify "*.*" in order to specify all files. This argument is case insensitive. (2) Details of file information: File name<LF> Size (the number of data characters) <LF> Date<LF>Time<LF> File attribute (3) File attribute: 0: Protection disabled 1: Protection enabled
Example of argument	.TXT
Example of reception data	2<LF> Test1.TXT<LF> 36<LF> 02/06/24<LF>14:49:54<LF> 0<LF> TEST2.TXT<LF> 8323<LF> 02/06/24<LF>14:12:54<LF> 0

[Read file]

Item	Description
Overview	Read a file from the robot controller.
Request ID	305
Argument at request	File name
Reception data	File name<LF> Number of data characters in the file<LF> Content of the file
Remark	The period of the extension is required. This argument is case insensitive.
Example of argument	Test.txt
Example of reception data	Test.txt<LF> 36<LF> --Test Data-- This is data for test.

[Write file]

Item	Description
Overview	Write the specified file to the robot controller.
Request ID	306
Argument at request	File name<LF> Number of data characters in the file<LF> Content of the file
Reception data	None
Remark	The period of the extension is required. This argument is case insensitive.
Example of argument	Test.txt<LF> 36<LF> --Test Data-- This is data for test.

[Set password]

Item	Description
Overview	Set the password used to restrict the access level within the robot controller.
Request ID	307
Argument at request	Password type<LF> Password
Reception data	None
Remark	Password type: p: Parameter
Example of argument	p<LF> *** (Please make an inquiry to the manufacturer.)

[Get movement range information]

Item	Description
Overview	Get the movement range information of the specified mechanism.
Request ID	308
Argument at request	Mechanism number
Reception data	Mechanism number<LF> Joint movement range J1 (-) [degrees]<LF>J1 (+) <LF>… J8 (-) <LF>J8 (+) <LF> ABS movement range J1 (-) [degrees]<LF>J1 (+) <LF>… J8(-) <LF>J8 (+) <LF> Orthogonal movement range X (-) [mm] <LF>X (+) <LF>Y (-) <LF>Y (+) <LF>Z (-) <LF> Z (+)
Remark	
Example of argument	1
Example of reception data	1<LF> -2.794<LF>2.794<LF>-0.738<LF>1.690<LF>…-80000.000<LF>80000.000<LF> -0.017<LF>0.017<LF>-0.017<LF>0.017<LF>…-1.000<LF>1.000<LF>-10000.000 <LF>10000.000<LF>-10000.000<LF>10000.000<LF>-2000.000<LF>2000.000

[Get slot information]

Item	Description
Overview	Get the information of the specified slot.
Request ID	309
Argument at request	Slot number
Reception data	Slot number<LF> Robot program name<LF> Operation mode<LF> Start condition<LF> Task priority order
Remark	(1) Specify the appropriate number (the number of slots + 1) for the slot number in order to send the request to the editing slot. See **8.1**, **"About Task Slots"** for the detailed explanation of slots. (2) Operation mode: REP: Continuous CYC: One cycle (3) Start condition: START: Normal ERROR: At error occurrence ALWAYS: Always (4) Specify a value in the range from 1 to 31 for the task priority order. This number indicates the number of lines executed each time the task runs. It has the same meaning as the number of executed lines in the PRIORITY instruction of MELFA-BASIC IV.
Example of argument	1
Example of reception data	1<LF> 1<LF> REP<LF> START<LF> 1

[Set slot information]

Item	Description
Overview	Set the information of the specified slot.
Request ID	310
Argument at request	Slot number<LF> Robot program name<LF> Operation mode<LF> Start condition<LF> Task priority order
Reception data	None
Remark	(1) Specify the appropriate number (the number of slots + 1) for the slot number in order to send the request to the editing slot. See **8.1**, **"About Task Slots"** for the detailed explanation of slots. (2) No extension is required for robot programs. (3) Operation mode: REP: Continuous CYC: One cycle (4) Start condition: START: Normal ERROR: At error occurrence ALWAYS: Always (5) Specify a value in the range from 1 to 31 for the task priority order. This number indicates the number of lines executed each time the task runs. It has the same meaning as the number of executed lines in the PRIORITY instruction of MELFA-BASIC IV.
Example of argument	1<LF> 1<LF> CYC<LF> START<LF> 1

[Initialize all parameters]

Item	Description
Overview	Return all parameter values to their default values.
Request ID	315
Argument at request	None
Reception data	None
Remark	This request resets the origin data as well; use it with caution.

[Get robot origin data]

Item	Description
Overview	Get the origin data of the robot.
Request ID	316
Argument at request	Mechanism number
Reception data	Checksum information<LF> Origin data J1<LF>J2<LF> … J8
Remark	(1) NULL is set for the origin data of non-existing axes. (2) This request is only valid when the key switch of the robot controller is in the AUTO (Ext.) mode.
Example of argument	1
Example of reception data	Z2DNL%<LF> 0065U? <LF> 005RO9<LF> 008GKC<LF> 005GCR<LF> Z2X4RA<LF> 000000<LF> <LF> ……. NULL ……. NULL

[Write robot origin data]

Item	Description
Overview	Write the origin data of the robot.
Request ID	317
Argument at request	Mechanism number<LF> Checksum information<LF> Origin data J1<LF>J2<LF> … J8
Reception data	None
Remark	(1) NULL is set for the origin data of non-existing axes. (2) This request is only valid when the key switch of the robot controller is in the AUTO (Ext.) mode.
Example of argument	1<LF> Z2DNL%<LF> 0065U? <LF> 005RO9<LF> 008GKC<LF> 005GCR<LF> Z2X4RA<LF> 000000<LF> <LF> ……. NULL ……. NULL

[Get parameter value (do not consider open level)]

Item	Description
Overview	Get the values of the specified parameters. All the parameters can be read regardless of whether or not a password has been entered.
Request ID	318
Argument at request	Mechanism number<LF> Number of parameters<LF> Parameter name 1<LF>Parameter name 2<LF> … Parameter name N
Reception data	Mechanism number<LF> Number of parameters<LF> Parameter name 1<LF>Number of data items 1<LF>Not used (0 is set.)<LF>Data 1<LF> Parameter name 2<LF>Number of data items 2<LF>Not used<LF>Data 2<LF> : Parameter name N<LF>Number of data items N<LF>Not used<LF>Data N
Remark	The number of elements in the parameter data is set for the number of data items.
Example of argument	1<LF> 2<LF> SLT1<LF> SLT2
Example of reception data	1<LF> 2<LF> SLT1<LF> 0<LF> 0<LF> 1, REP, START, 1<LF> SLT2<LF> 0<LF> 0<LF> 90, REP, ALWAYS, 1

[Check writing of additional axis origin]

Item	Description
Overview	Check whether or not it is possible to write additional axis' origin.
Request ID	327
Argument at request	Mechanism number
Reception data	Result
Remark	Result: 0: Writing allowed 1: Writing prohibited
Example of argument	1
Example of reception data	1

[Setting the origin data]

Item	Description
Overview	Setting the oritin data of the robot whith specified method.
Request ID	343
Argument at request	Mechanism number<LF> The method of setting the origin data (0/1/3/4 = User origin methodMechanical stopper/Tool/ABS)<LF> The axis pattern of the origin data(Decimal number)
Reception data	None
Remark	It is not possible to set the origin data while Servo ON. Set it after turning servo off.
Example of argument	1<LF> 1<LF> 63

[Reading the origin data]

Item	Description
Overview	Reading the origin data of the robot.
Request ID	344
Argument at request	Mechanism(Robot) number
Reception data	0/1 = Unsetting the origin data/The origin data set completion<LF> The axis pattern of the origin data.
Remark	
Example of argument	1<LF>
Example of reception data	1<LF> 63

[Writing parameter(with the necessity of rebooting the robot controller)]

Item	Description
Overview	Writing the specified parameter into the robot controller. (Getting the necessity of rebooting the robot controller.)
Request ID	346
Argument at request	Mechanism number <LF> Number of parameters<LF> Parameter name 1<LF>Data 1<LF> Parameter name 2<LF>Data 2<LF> : Parameter name N<LF>Data N
Reception data	0/1/2 = Unnecessary/Necessity/Not possible to reboot
Remark	(1) This ID can be used only in MELFA-BASIC V. (2) The difference with ID 302 is that the reply data has whether rebooting is necessary or unnecessary.
Example of argument	1<LF> 2<LF> SLT1<LF> 2, REP, START, 1<LF> SLT2<LF> 2, REP, ALWAYS, 1
Example of reception data	1

[Reading the serial number of robot and controller]

Item	Description
Overview	Reading the serial number of the robot and the robot controller.
Request ID	350
Argument at request	None
Reception data	Number of connected mechanism(Robot) to the robot controller<LF> Serial number of the robot controller<LF> Date and time when the robot controller's serial number was set<LF> Serial number of the mechanism(Robot) 1<LF> Date and time when the mechanism(Robot) 1 was set<LF>…<LF> Serial number of the mechanism(Robot) N<LF> Date and time when the mechanism(Robot) N was set
Remark	This ID can be used only in MELFA-BASIC V.
Example of reception data	1<LF> AR0703001<LF> 2007/10/26 12:00:00

5.3.4 Requests Related to Operation

* <LF> indicates the line feed character (0AH).

⚠ DANGER

Use the requests to operate the robot with due caution.

When you send the requests to operate the robot, make sure to check the requests and the transmission data. The robot may move in unexpected ways, causing accidents that may result in injury or physical damage.

[Execute program]

Item	Description
Overview	Execute the robot program in the specified slot.
Request ID	400
Argument at request	Slot number<LF> Robot program name<LF> Execution method
Reception data	None
Remark	(1) The programs in all the slots are executed if 0 is specified for the slot number. See **8.1**, **"About Task Slots"** for the detailed explanation of slots. (2) Specify the appropriate number (the number of slots + 1) for the slot number in order to send the request to the editing slot. (3) No extension is required for robot programs. (4) The robot program in the specified slot is executed if NULL is specified for the robot program name. Note, however, that if 0 is specified for the slot number, the robot program selected immediately before is executed. (5) Execution method: 0: Continuous operation 1: Cycle operation (6) This request is valid only when the key switch of the robot controller is in the AUTO (Ext.) mode.
Example of argument	1<LF> 1<LF> 0

[Stop]

Item	Description
Overview	Stop the robot program being executed in the specified slot.
Request ID	401
Argument at request	Slot number
Reception data	None
Remark	(1) The programs in all the slots are stopped if 0 is specified for the slot number. See **8.1**, **"About Task Slots"** for the detailed explanation of slots. (2) Specify the appropriate number (the number of slots + 1) for the slot number in order to send the request to the editing slot.
Example of argument	1

[Servo ON/OFF]

Item	Description
Overview	Turn the servo on or off.
Request ID	403
Argument at request	Mechanism number<LF> Servo on/off
Reception data	None
Remark	(1) Servo on/off: 0: Off 1: On (2) This request is valid only when the key switch of the robot controller is in the AUTO (Ext.) mode.
Example of argument	1<LF> 1

[Set speed override]

Item	Description
Overview	Set the speed override.
Request ID	404
Argument at request	The value of the speed override [%]
Reception data	None
Remark	Please note that this is not the setting for jog override.
Example of argument	50

[Jog operation]

Item	Description
Overview	Execute the jog operation.
Request ID	405
Argument at request	Mechanism number<LF> Jog type<LF> Level type<LF> Jog override [%] <LF> Feed direction 1<LF>Feed direction 2<LF> ... Feed direction 8<LF> Fixed length
Reception data	None
Remark	(1) Jog type: 0: Joint 1: Orthogonal 2: Tool (2) Level type: 0: Edge 1: Level Specify 1 (level) for the level type if the fixed-length is set to 0 (off). Specify 0 (edge) for the level type if the fixed-length is set to 1 (high) or 2 (low). (3) The jog override function can only be used with robot controllers of version B5 or later. Specify 100 for versions before B4. (4) Feed direction: -1: Move in the minus direction 0: Do not move 1: Move in the plus direction Specify in the order of X, Y, Z, and so forth if the jog type is 1 (orthogonal) or 2 (tool), and in the order of J1, J2, J3 and so forth if it is 0 (joint). (5) Fixed-length : 0: Off 1: High 2: Low If 0 (off) is specified for the fixed-length, the robot continues the jog operation until the request is canceled and the robot is requested to stop. See "How to use" below. If low or high is specified for the fixed-length, the robot stops after performing one move. (6) This request is only valid when the key switch of the robot controller is in the AUTO (Ext.) mode.
How to use	How to make the robot continue the jog operation while pressing the button control ● The jog operation is started when the button is pressed. Specify 0 (off) for the fixed-length and request the jog operation. The robot, in turn, continues the jog operation. ● The request is canceled when the button is released, and the robot is stopped via a stop request. Send a cancellation of the request of the jog operation using the RequestCancel method. Then, send the stop request (request ID = 401) to all the slots using the RequestService or RequestService2Null method. **Note that the robot will continue to move on its own due to inertia if the request is simply canceled. Make sure to send a stop request as well.**
Example of argument	1<LF> 0<LF> 0<LF> 50<LF> 1<LF>1<LF>1<LF>1<LF>1<LF>1<LF>0<LF>0<LF> 2

[Stop cycle]

Item	Description
Overview	Specify/cancel stopping the cycles.
Request ID	406
Argument at request	Specification/cancellation of stopping the cycles
Reception data	None
Remark	Specification/cancellation of stopping the cycles: 0: Specify 1: Cancel
Example of argument	0

[Reset error]

Item	Description
Overview	Reset the currently generated errors.
Request ID	407
Argument at request	None
Reception data	None
Remark	

[Initialize slot]

Item	Description
Overview	Initialize slots.
Request ID	408
Argument at request	None
Reception data	None
Remark	(1) This request is only valid when the key switch of the robot controller is in the AUTO (Ext.) mode. (2) See **8.1**, **"About Task Slots"** for the detailed explanation of slots.

[Reset program]

Item	Description
Overview	Initialize the robot program in the specified slot.
Request ID	409
Argument at request	Slot number
Reception data	None
Remark	(1) Specify the appropriate number (the number of slots + 1) for the slot number in order to send the request to the editing slot. See **8.1**, **"About Task Slots"** for the detailed explanation of slots. (2) This request is only valid when the key switch of the robot controller is in the AUTO (Ext.) mode.
Example of argument	1

[Connect robot]

Item	Description
Overview	Establish a connection between the robot and the robot controller.
Request ID	410
Argument at request	None
Reception data	Robot program extension<LF> Parameter extension<LF> Model name<LF> Controller model name<LF> The character string "MELFA" <LF> Date<LF> Version<LF> Language information<LF> Copyright information<LF> Mechanism connection information<LF> Not used (1 is set.) <LF> Number of valid slots
Remark	(1) Language information: ENG: English JPN: Japanese (2) The mechanism connection information is a numerical data value, where a hexadecimal number is used to represent the binary data. The connection states of mechanism 1, mechanism 2 and so on are set starting from bit 0.
Example of reception data	MB4<LF> PRM<LF> RV-100TH<LF> CRn-5xx<LF> MELFA<LF> 02-03-11<LF> Ver.H4<LF> ENG<LF> COPYRIGHT(C)1999 MITSUBISHI ELECTRIC CORPORATION ALL RIGHTS RESERVED<LF> 1<LF> 1<LF> 8

[Delete all robot programs]

Item	Description
Overview	Delete all robot programs.
Request ID	411
Argument at request	None
Reception data	None
Remark	

[Enabling/Disabling operation right]

Item	Description
Overview	Enable or disable the external operation right.
Request ID	412
Argument at request	Enable/disable
Reception data	None
Remark	Enable/disable: 0: Disable 1: Enable
Example of argument	1

[Clear the task slots]

Item	Description
Overview	Releasing the selection of the robot program in the specified task slots.
Request ID	415
Argument at request	Number of the cleared task slot<LF> Cleared task slot number (1st)<LF>…<LF> Cleared task slot number (Nth)
Reception data	None
Remark	The task slot cannot be cleared while the error is occurring.
Example of argument	2<LF> 1<LF> 3

[Selecting the robot program]

Item	Description
Overview	Setting the robot program in the specified task slot.
Request ID	416
Argument at request	Slot number<LF> Robot program name 名
Reception data	None
Remark	(1) It is necessary to get the right to operate before setting the robot program. (2) No extension is required for robot programs.
Example of argument	1<LF> 2

[Reading the speed override of OP]

Item	Description
Overview	Reading the speed override of OP.
Request ID	417
Argument at request	None
Reception data	Value of the speed override(%).
Remark	
Example of reception data	70

[Getting clock time]

Item	Description
Overview	Getting the day and time from the robot controller.
Request ID	418
Argument at request	None
Reception data	Year<LF>Month<LF>Day<LF>Hour<LF>Minute<LF>Second
Remark	
Example of reception data	9<LF> 2<LF> 24<LF> 14<LF> 58<LF> 3

[Setting clock time]

Item	Description
Overview	The clock in the robot controller is set.
Request ID	419
Argument at request	Year<LF>Month<LF>Day<LF>Hour<LF>Minute<LF>Second
Reception data	None
Remark	
Example of argument	9<LF> 2<LF> 24<LF> 14<LF> 58<LF> 3

[Open / Close the robot hand]

Item	Description
Overview	The selected robot hand is opened or closed.
Request ID	421
Argument at request	Mechanism number <LF> Number of the Hand<LF> 1/0 = Open/Close
Reception data	None
Remark	
Example of argument	1<LF> 1<1LF> 0

[Change the override]

Item	Description
Overview	The speed override is changed.
Request ID	426
Argument at request	0 : The speed override is raised by 10% 1 : The speed override is lowered by 10%.
Reception data	None
Remark	
Example of argument	1

[The robot hand is lined up]

Item	Description
Overview	The robot hand is lined up.
Request ID	427
Argument at request	Mechanism number
Reception data	None
Remark	
Example of argument	1

6 Events

This chapter explains the events that can be generated.

MsgRecvEvent

[Function]

This event is generated when data is received from the Communication Server 2.

[Description]

Note, however, that data is only received from the Communication Server 2 when a request is sent from MelfaRXM.ocx and there is reception data available.

[Example]

```
Private Sub MelfaRXM_MsgRecvEvent()
    Dim lState As Long
    Dim lID As Long
    Dim sRecvData As String
    Dim lStatus As Long
    Dim lError As Long
    Dim I As Long
    Dim lCnt As Long
    Dim sData As String
    Dim sString As String

    lState = MelfaRxM.GetRecvDataM(1, lID, sRecvData, lStatus, lError)
    If lState =1 Then
        sString = ""
        lCnt = MelfaRxM.GetDataCnt(sRecvData)
        For I = 0 To (lCnt - 1)
            sData = MelfaRxM.GetOneData(I, sRecvData)
            sString = sString + sData + Chr$(13) + Chr$(10)
        Next
        tbRecvData.Text = sString
            :
            :
End Sub
```

7 Sample Program

This chapter explains the sample program that can be installed.
The sample programs are written in the following languages.

- Borland C++ Builder 5.0
- Visual Basic 6.0.
- Visual C++ 6.0
- Visual Basic .NET 2003
- Visual C# (Visual Studio 2010)
  (This has confirmed the operation only in a Japanese version.)

7.1 Overview

Table 7-1 provides an overview of the sample program.

Table 7-1 : Overview of Sample Program

Project name	ToolTest
Main functions	The sample program has the following main functions. - Send requests to the controller using the RequestService method and display the data received. - Read a list of robot programs from the robot controller. - Turn the servo on/off. - Execute/stop a robot program. - Execute error reset. - Reset a program. - Monitor the current position and error codes being generated.

7.2 Detailed Specifications

This section explains the detailed specifications for each window (VB sample program).

7.2.1 Request ID Test Window (Object Name: FormMain)

This window is used to send individual requests to the controller via the RequestService method and display the data received.

7.2.1.1 Processing at Event Generation

Table 7-2 shows the processing performed when an event is generated.

Table 7-2 : Processing at Event Generation

Event	Processing
Startup window (Form_Load)	The Communication Server 2 is started, and the information of the robot controller set in the Communication Server 2 is acquired.
Exit window (Form_Unload)	The Communication Server 2 is closed.
Data reception (MsgRecvEvent)	The reception data is acquired and the information displayed via the controls.

7.2.1.2 Control Specifications

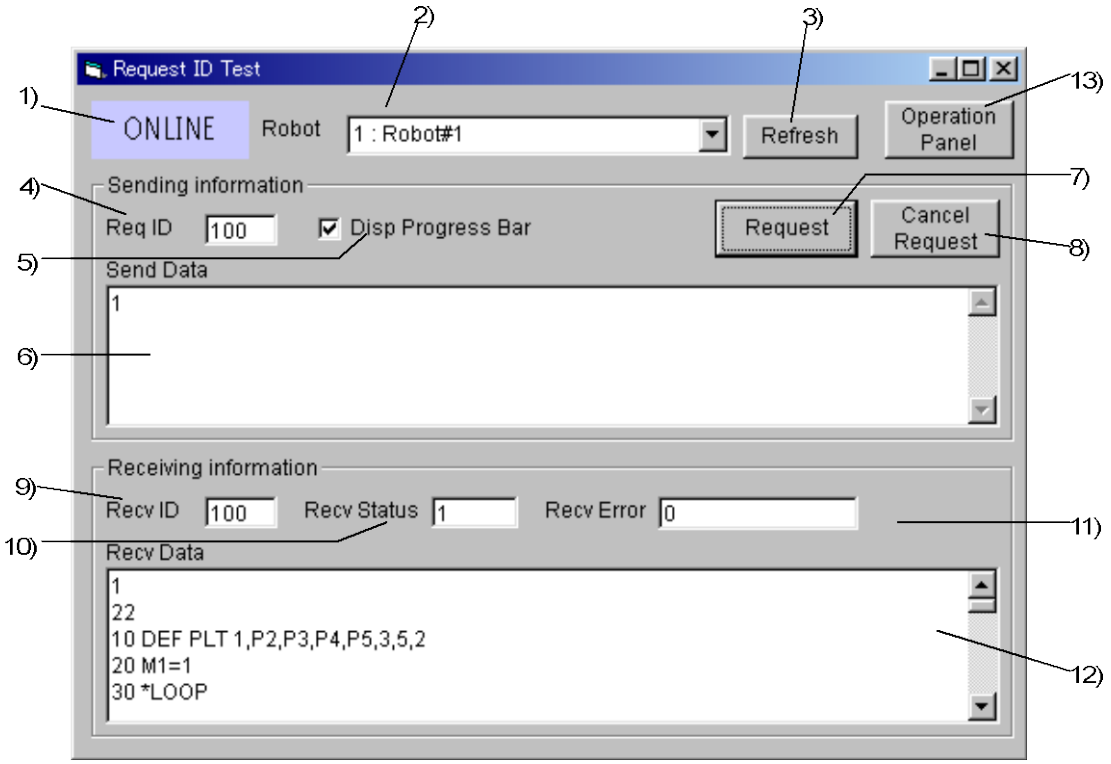


Figure 7-1 : Request ID Test Window

Table 7-3 : Control Specifications

Control	Specification
1) OCX control	Visualizes the attribute; ONLINE is displayed if the Communication Server 2 is started and connected to the robot controller, and OFFLINE if it is not connected (OCX specifications).
2) Robot	Displays the number and name of the controller specified in the Communication Server 2. The controller number can be specified via RequestService.
3) Refresh	Updates the information in the [Robot] combo box. When the number and name of the controller are changed in the Communication Server 2, this button should be clicked to get the information again.
4) Req ID	Enter the request ID to be specified via RequestService.
5) Disp Progress Bar	When this check button is enabled, a progress bar is displayed during communication.
6) Send Data	Enter the transmission data to be specified via RequestService. Use new lines in order to enter multiple transmission data elements.
7) Request	Click this button to execute the RequestService method.
8) Cancel Request	Click this button to execute the RequestCancel method.
9) Recv ID	Displays the request ID of the reception data.
10) Recv status	Displays the status of data reception. 0: Undefined request (at transmission) 1: Reception succeeded 2: Transmission error 3: Reception timeout 4: Transmission canceled 5: Execution error
11) Recv Error	When the status of data reception is 5 (execution error), the error number of the robot controller is displayed.
12) Recv Data	Displays the reception data.
13) Operation Panel	Click this button to start up the Operation Panel window. The Operation Panel window is explained in the next section.

7.2.1.3 Operation Procedure

Operate the application according to the following procedure.

(1) Start up the application.
(2) Perform the communication settings of the Communication Server 2 if it is used for the first time. (See **4**, **"Procedure for Setting Up Communication Server"** for how to specify the communication settings.) Click the [Refresh] button to update the information in the [Robot] combo box if the settings were changed.
(3) Specify the robot controller number, request ID, and transmission data.
(4) Click the [Request] button to send the request.
(5) The reception data is displayed.

7.2.2 Operation Panel Window (Object Name: FormOP)

This window has the following functions.

- Read a list of robot programs from the robot controller.
- Turn the servo on/off.
- Execute/stop a robot program.
- Execute error reset.
- Reset a program.
- Monitor the current position and error codes being generated.

7.2.2.1 Processing at Event Generation

The following processing is performed.

Table 7-4 : Processing at Event Generation

Event	Processing
Startup window (Form_Load)	Requests to get a list of robot programs (request ID = 106). It also requests to get the current position (request ID = 235) and any errors that have been generated (request ID = 203) at a reception interval of 1000 msec.
Exit window (Form_Unload)	Requests to stop the robot program (request ID = 401) and reset the program (request ID = 408). It also executes the RequestCancel method. This method stops monitoring the current position and generated error numbers.
Data reception (MsgRecvEvent)	Acquires the reception data and displays the information via the controls.

7.2.2.2 Control Specifications

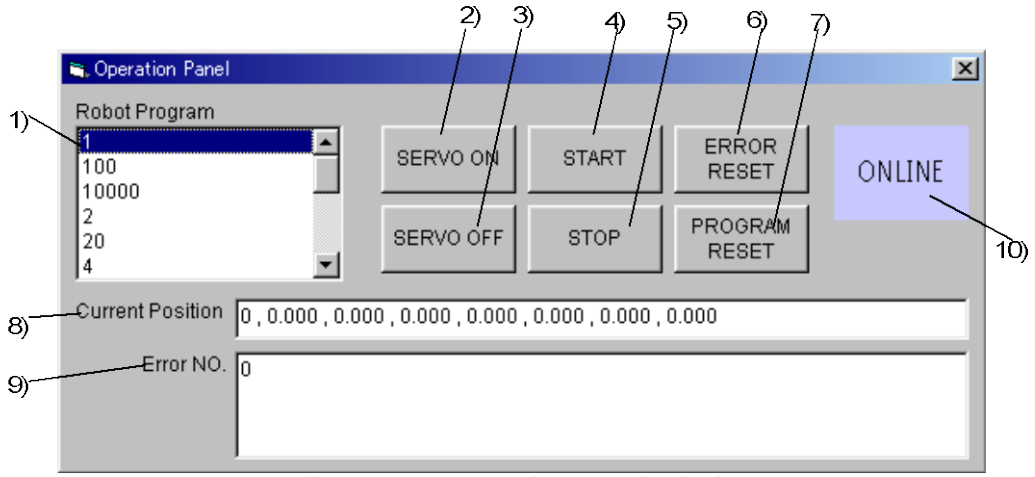


Figure 7-2 : Operation Panel Window

Table 7-5 : Control Specifications

Control	Specification
1) Robot Program	Displays a list of robot programs in the robot controller.
2) SERVO ON	Click this button to turn the servo on.
3) SERVO OFF	Click this button to turn the servo off.
4) START	Click this button to execute the robot program selected in the list of robot programs.
5) STOP	Click this button to stop the robot program being executed.
6) ERROR RESET	Click this button to reset an error.
7) PROGRAM RESET	Click this button to reset the program.
8) Current Position	Displays the current position.
9) Error NO.	Displays the currently generated error number and error message.
10) OCX control	Visualizes the attribute; ONLINE is displayed if the Communication Server 2 is started and connected to the robot controller, and OFFLINE if it is not connected (OCX specifications).

7.2.2.3 Operation Procedure

Operate the application according to the following procedure.

(1) Check that the OP key switch in the front of the robot controller is set to AUTO (Ext.).
(2) Click the [SERVO ON] button to turn the servo on.
(3) Select the program to be executed from the list of robot programs.
(4) Click the [START] button to execute the program.
    If an error occurs, the error number is displayed in the [Error NO.] text box. Click the [ERROR RESET] button to reset the error.
(5) Click the [STOP] button to stop the program.
(6) Click the [PROGRAM RESET] button to reset the program.

8 Appendix

8.1 About Task Slots

When using MelfaRXM.ocx, there are cases where the concept of robot controller task slots is required. This section explains the concept of task slots.

A robot controller employs a concept known as "task slots" in order to implement multi-task functionality. Task slots can be considered as containers, in which robot programs can be placed; 32 user slots (commonly called slots) and one editing slot are provided.

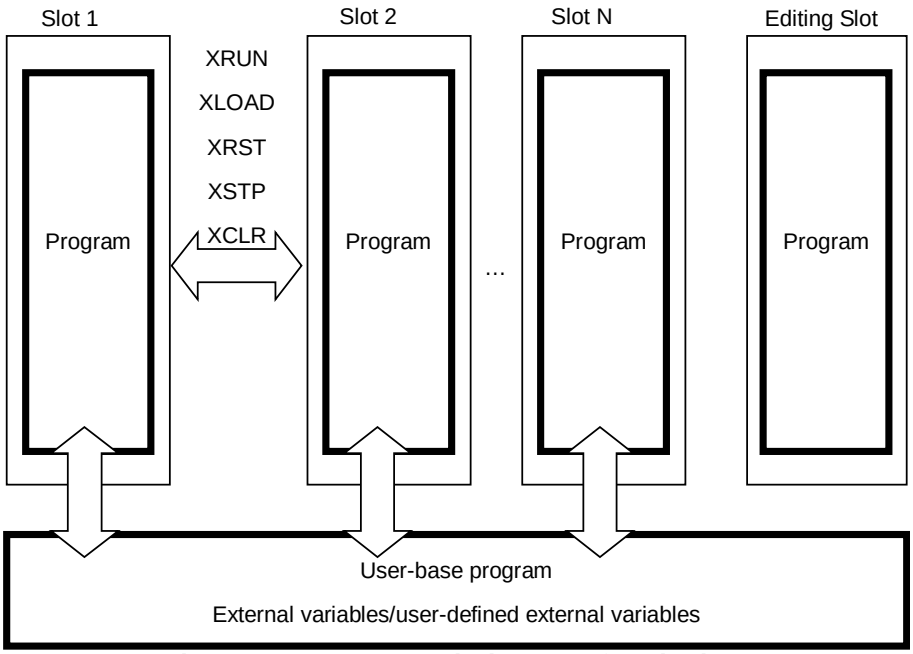


Figure 8-1 : Conceptual Diagram of Task Slots

A user slot is a task slot in which the user can store a program along with attribute information associated with it. When executing a program, the robot program to be executed must be placed in a user slot.
The number of slots to be used is specified in the parameter TASKMAX, and the information of the individual slots is specified in the parameters SLT1 to 32. It is possible to specify up to 32 slots. Slot number 1 is called the main slot. Whenever a slot number is not specified, the operation is performed on the main slot by default.
Multiple user slots are used at the same time when the multi-task functionality is used.

A user slot can be in one of the three states, which are switched between as shown in Figure 8-2.

1) Operating
2) Standing by
3) A program can be selected

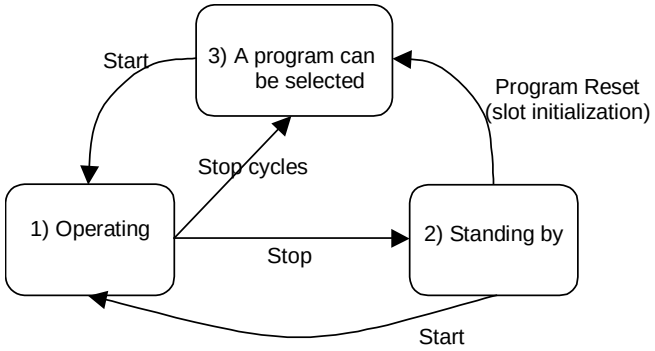


Figure 8-2 : Slot Status Diagram

Unlike the user slots, the editing slot is used when performing direct operation and step operation, as well as when editing a robot program directly (editing a program using the teaching box). These operations should be performed by placing the target robot program in the editing slot.

Each of the task slots is managed by its slot number.
The slot numbers of the user slots are from 1 to 32 (maximum) and the slot number of the editing slot is "number of user slots + 1." MelfaRXM.ocx provides a request to specify the slot number; specify slot number 1 (main slot) if the environment does not require multi-tasking and specify "number of slots + 1" to specify the editing slot.

See the instruction manual of the robot controller for the detailed explanation of the multi-task function.

8.2 Precautions for Distribution

1. Please use the installer for distribution in order to run applications created using MelfaRXM.ocx on another PC. The installer is installed in the folder as follows:

   C:(A folder specified during install operation)/MelfaRXM_Dev/Redist/Installer/SetupRXM.exe

⚠ Caution

Uninstall the older version before installing.

If an older version of the “Files for the distribution" is installed in the computer already, be sure to uninstall the older version before installing the newer version.
Especially, please be sure to uninstall them when you use the “Files for the distribution" in the CD-ROM of RT ToolBox. Even if the upgrade message appears when you try to overwrite, actually the upgrade would be fail.

2. Please install **“RT ToolBox2“** (version 1.1 or later) when you use the CRnQ communication.