MITSUBISHI ELECTRIC

Changes for the Better

Communication Middleware for MITSUBISHI FA

EZSocket

User's Reference Manual (CNC)

EZSocket

IB-1500195(ENG)-E

INTRODUCTION

Thank you for choosing EZSocket, the Mitsubishi communication middleware for Factory Automation controllers. In this user's reference manual, the usage of EZSocket OLE/COM interface is described.

Always read through this manual before use and fully understand the EZSocket function to ensure correct use.

TRADEMARKS

Windows, Windows NT, Visual C++ and Visual Basic are either trademarks or registered trademarks of Microsoft Corporation.
EZSocket is a registered trademark of Mitsubishi Electric Corporation.
Other brands and product names throughout this manual are trademarks or registered trademarks of the respective company.

SAFETY PRECAUTIONS

(Read these precautions before using.)

When using this product, thoroughly read this manual and the associated manuals. Pay careful attention to safety and handle the product properly. The safety precautions described here are in relation to this product. Get acquainted with this CNC unit and thoroughly study the safety information and precautions before use. These SAFETY PRECAUTIONS classify the safety precautions into two categories: "DANGER" and "CAUTION".

DANGER	Indicates that incorrect handling may cause hazardous conditions, resulting in death or severe injury.
CAUTION	Indicates that incorrect handling may cause hazardous conditions, resulting in medium or slight personal injury or physical damage.

Note that the items under " CAUTION" could lead to serious consequences as well depending on the situation. Please follow all items listed in "Safety Precautions" as they are equally important.
Keep this manual handy for your future reference.

[DESIGN PRECAUTIONS]

DANGER

- When connecting with NC unit, install an external safety circuit so that the entire system performs on the safe side when malfunction occurs in the external power supply or personal computer body.
- Incorrect input or incorrect operation may cause an accident.
- Writing function to NC unit directly affects to machine control.
- An unexpected operation may be resulted due to incorrect parameter setting. (EX. Setup)
- Execute after careful check.

[SETUP and MAINTENANCE PRECAUTIONS]

CAUTION

- Operation error could result in damage to machine or accident.
- Some functions may be different or may not be available depending on the version of NC unit.

CONTENTS

1. OVERVIEW 1-1
1.1 Features 1-1
1.2 Supported Products 1-1
1.3 Connection Configuration 1-2
1.3.1 Connection with MELDASMAGIC Series 1-2
1.3.2 Connection with MELDAS600 Series 1-2
1.3.3 Connection with MELDAS C6/C64 Series 1-3
1.3.4 Connection with Mitsubishi CNC700 Series 1-3
1.3.5 Connection with Mitsubishi CNC C70 Series 1-4
1.4 Development Environment 1-5
1.5 Installing EZSocket 1-6
1.6 Using EZSocket 1-6
1.7 EZSocket Interface 1-7
1.7.1 EZSocket custom interface 1-7
1.7.2 EZSocket automation interface 1-8
1.8 EZSocket Programming Procedure 1-9
1.8.1 Programming procedure with VC++ (1) 1-9
1.8.2 Programming procedure with VC++ (2) 1-10
1.8.3 Programming procedure with VB (1) 1-11
1.8.4 Programming procedure with VB (2) 1-12
2. DETAILS OF I/F SPECIFICATION 2-1
2.1 Common Items 2-1
2.2 Method List 2-4
2.3 IEZNcCommunication3 Interface 2-9
2.4 IEZNcSystem Interface 2-24
2.5 IEZNcPosition Interface 2-28
2.6 IEZNcCommand2 Interface 2-36
2.7 IEZNcProgram2 Interface 2-42
2.8 IEZNcTime Interface 2-48
2.9 IEZNcAxisMonitor Interface 2-58
2.10 IEZNcRunStatus Interface 2-79
2.11 IEZNcFile5 Interface 2-85
2.12 IEZNcCommonVariable2 Interface 2-111
2.13 IEZNcLocalVariable2 Interface 2-117
2.14 IEZNcTool3 Interface 2-120
2.15 IEZNcATC2 Interface 2-155
2.16 IEZNcParameter2 Interface 2-167
2.17 IEZNcOperation Interface 2-171
2.18 IEZNcDevice Interface 2-174
2.19 IEZNcGeneric2 Interface 2-182
2.20 IEZNcSubFunction2 Interface 2-190
3. ERROR CODE LIST 3-1

4. HOW TO USE API 4-1
4.1 EZSocketAPI Operation Procedure 4-1
4.2 Initialization for OLE/COM Interface Use 4-2
4.3 Create EZSocket Object 4-3
4.4 EZSocket Include Files 4-3
4.5 Programming EZSocket Automation Interface by VB 4-4
4.5.1 How to use EZSocket OLE automation interface by VB 4-4
4.5.2 How to program by VB (1) 4-6
4.5.3 How to program by VB (2) 4-7
5. APPLICATION INSTALLATION PROCEDURE 5-1
5.1 Overview 5-1
5.2 Redistribution by Using EZSocket Installer for Redistribution 5-1
5.2.1 Path to the folder in which EZSocket installer for redistribution is saved 5-1
5.2.2 Specification and processing of the EZSocket installer for redistribution 5-2
5.2.3 Troubleshooting 5-4
5.3 Rules on EZSocket Redistribution 5-5
5.3.1 Redistributable module group 5-5
5.3.2 Redistributable files of EZSocket 5-5
5.4 Installation 5-6
5.4.1 Upgrading redistributable files 5-6
5.4.2 Install destination directory of EZSocket files 5-6
5.5 Structure of the Install Destination Directory 5-7
5.6 Registry Setting 5-8
5.6.1 Renewing EZSocket registry 5-8
5.7 System Environment Variable Setting 5-8
5.8 COM Information Registry Setting 5-9
5.9 Precaution for Uninstalling 5-9
6. SAMPLE APPLICATION 6-1
6.1 Overview of Sample Application 6-1
6.2 Position Data Display Application 6-1
6.2.1 Performance environment 6-1
6.2.2 Installation and uninstallation 6-1
6.2.3 Executing sample application 6-1
6.2.4 Function list 6-2
6.2.5 Screen configuration and functions 6-2
6.2.6 Set project work space 6-4
6.2.7 IEZNcCommunication object3 6-5
6.2.8 IEZNcPosition object 6-5
6.3 Monitoring Application 6-5
6.3.1 Performance environment 6-5
6.3.2 Installation and uninstallation 6-6
6.3.3 Executing sample application 6-6
6.3.4 Function list 6-6
6.3.5 Screen configuration and functions 6-7
6.3.6 Setting project work space 6-8
6.4 Macro Sample Program 6-8
6.4.1 Performance environment 6-8
6.4.2 Installation and uninstallation 6-8
6.4.3 Executing sample program 6-8

7. CONSOLE PROGRAM SAMPLE 7-1
7.1 The Console Program to Connect to CNC C70 (Via Ethernet) 7-1
7.2 The Console Program which Runs on the Computer on which MELDASMAGIC64 is Mounted 7-4
7.3 The Console Program which Runs MELDASMAGIC64 From the Remote Personal Computer Connected by Ethernet 7-5
7.4 The Macro Sample Program that Uses EZSocketOLE Interface Macro 7-7
8. RESCTRICTION 8-1
8.1 Restriction in Performance on EZSocket Server 8-1

1. OVERVIEW

EZSocket (CNC edition) is the middleware product that enables programming applications with Windows interface for MITSUBISHI numerical control unit, including CNC700 series, MELDAS 600 series, MELDAS C6/C64 series, MITSUBISHI CNC C70 series and MELDASMAGIC series.
Development efficiency can be improved with EZSocket, by using the OLE interface with our MELDAS series, development is possible without knowing internal processes in the numerical controllers.

1.1 Features

- The functions of MITSUBISHI CNC700 series, MELDAS 600 series, MELDAS C6/C64 series, MITSUBISHI CNC C70 series and MELDASMAGIC64 can be easily used on Windows applications by Visual C ++, Visual Basic, and Visual Basic for Applications.
- Complicated processes such as communication protocol between PC and MITSUBISHI CNC700 series, MELDAS 600 series, MELDAS C6/C64 series, MITSUBISHI CNC C70 series or MELMAGIC are done in EZSocket. Thus, solution providers can develop highly functional application very quickly.
- EZSocket also has variations that enable application programs on Windows PC to communicate with the MITSUBISHI FA products other than MITSUBISHI CNC700 series, MELDAS 600 series, MELDAS C6/C64 series, MITSUBISHI CNC C70 series and MELDASMAGIC (including MELSEC sequencer, MELFA manufacturing robot, motion controller, and MELSERVO servo drive unit). Thus, the workload of solution providers can be drastically decreased by using API in the same way.
- Immediately after our new products are released, corresponding new version of EZSocket will be released. By using new version of EZSocket, solution providers can timely and easily develop or enhance application programs that support new products.

1.2 Supported Products

EZSocket supports the products mentioned below.

Table 1-1 Numerical control unit supported by EZSocket

Series	Product name
MITSUBISHI CNC700 series (hereinafter called "CNC700")	M720、M730、M750 M70A、M70B
MELDAS 600 series (hereinafter called "M6x5")	M615M, M635M, M655M, M615L, M635L
MELDASMAGIC series (hereinafter called "Magic64")	MELDASMAGIC64*
MELDAS C6/C64 series (hereinafter called "C64")	MELDAS C6, MELDAS C64
MITSUBISHI CNC C70 series (hereinafter called "C70")	C70

* EZSocket supports both PCI bus and ISA bus connection.
MELDASMAGIC64 products via PCI bus connection and those via ISA bus connection are hereinafter called, respectively, "PCI NC card" and "ISA NC card". For details such as model name, refer to BNP-B2191 ”MELDASMAGIC64 Setup Instruction Manual”.

1.3 Connection Configuration

This chapter explains the connection configuration between Mitsubishi CNC and PC. In order to carry out communication between CNC and PC using EZSocket, prepare necessary devices, such as CNC, PC, cable, etc. For details how to connect the deivces, refer to each CNC's instruction manual.

1.3.1 Connection with MELDASMAGIC Series

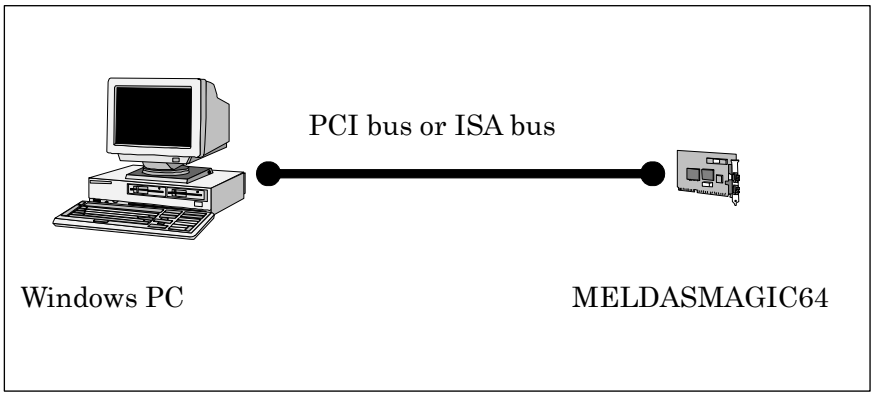


Fig. 1-1 Connection with MELDASMAGIC Series

1.3.2 Connection with MELDAS600 Series

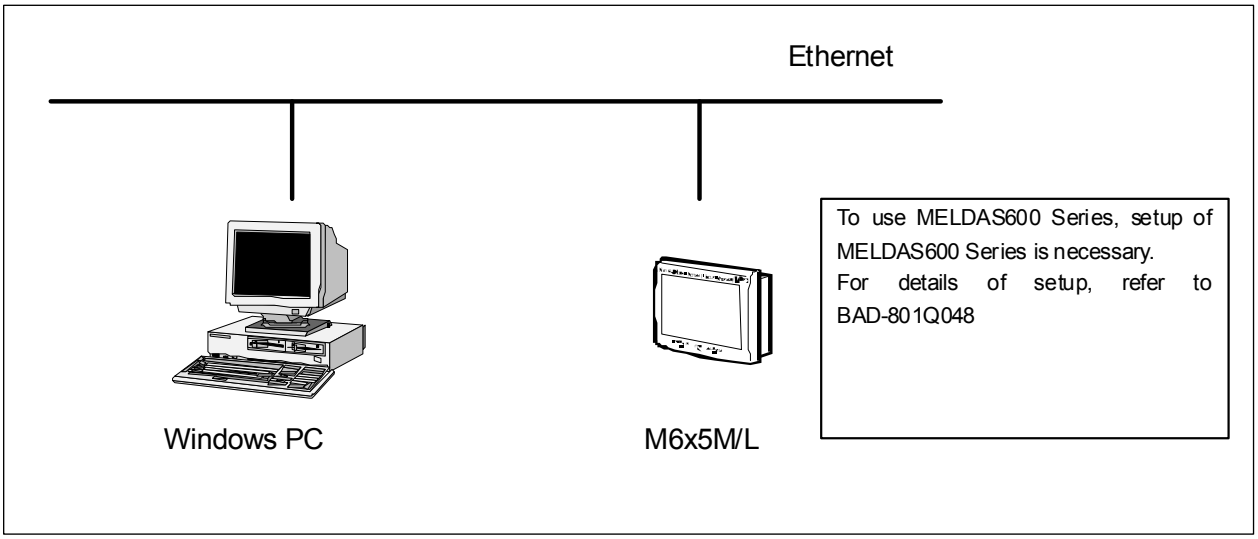


Fig. 1-2 Connection with MELDAS600 Series

1.3.3 Connection with MELDAS C6/C64 Series

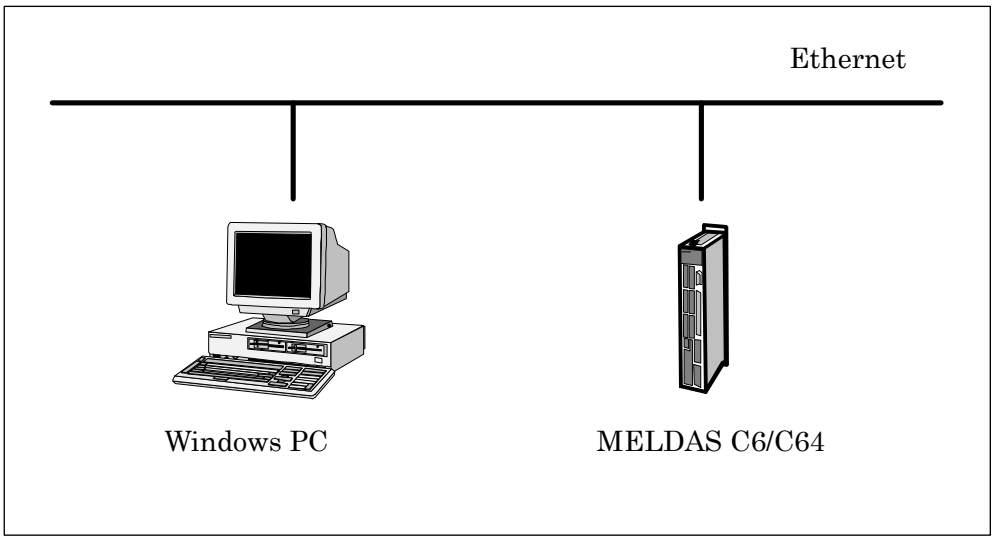


Fig. 1-3 Connection with MELDAS C6/C64 Series

1.3.4 Connection with Mitsubishi CNC700 Series

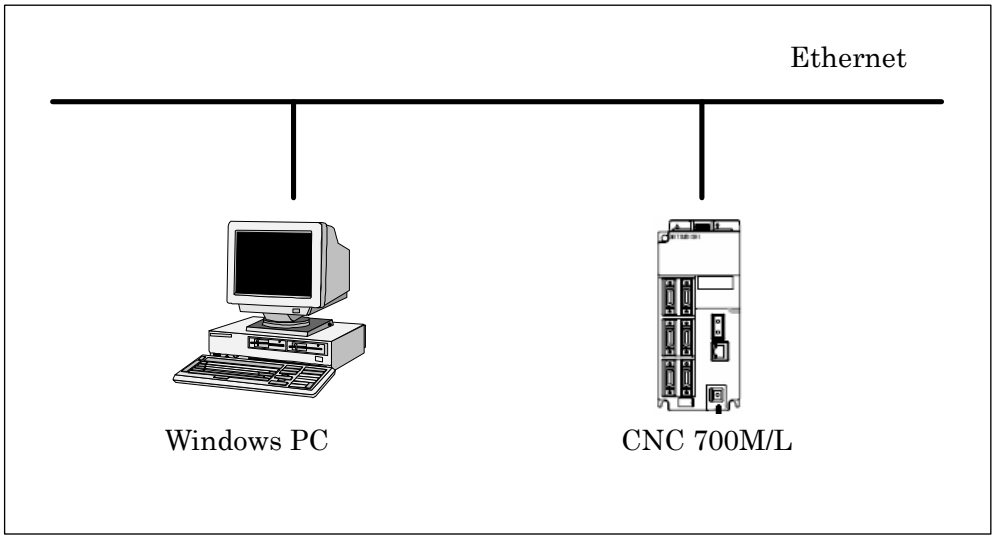


Fig. 1-4 Connection with Mitsubishi CNC700 Series

1.3.5 Connection with Mitsubishi CNC C70 Series

The connection path between Mitsubishi CNC C70 Series and PC is as follows.

Model	Connection path	Connected module	Figure No.
CNC C70	Ethernet connection	CNC C70	Fig. 1-5
	Ethernet connection via QJ71E71	QJ71E71	Fig. 1-5
	Ethernet connection via QnUDE	QnUDE	Fig. 1-5
	USB connection via QnUD	QnUD	Fig. 1-6
	RS-232C connection via QnUD	QnUD	Fig. 1-6
	USB connection via QnUDE	QnUDE	Fig. 1-6
	GOT transparent (USB connection)	Base unit	Fig. 1-7
	GOT transparent (RS-232C connection)	Base unit	Fig. 1-7

Table 1-1 Connection path with Mitsubishi CNC C70 Series

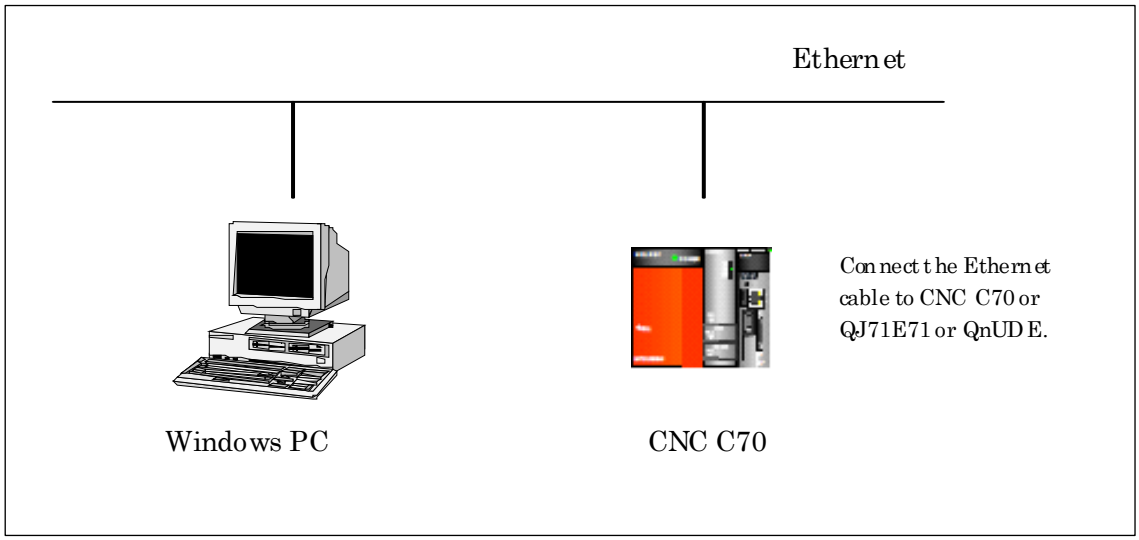


Fig. 1-5 Ethernet connection with Mitsubishi CNC C70 Series

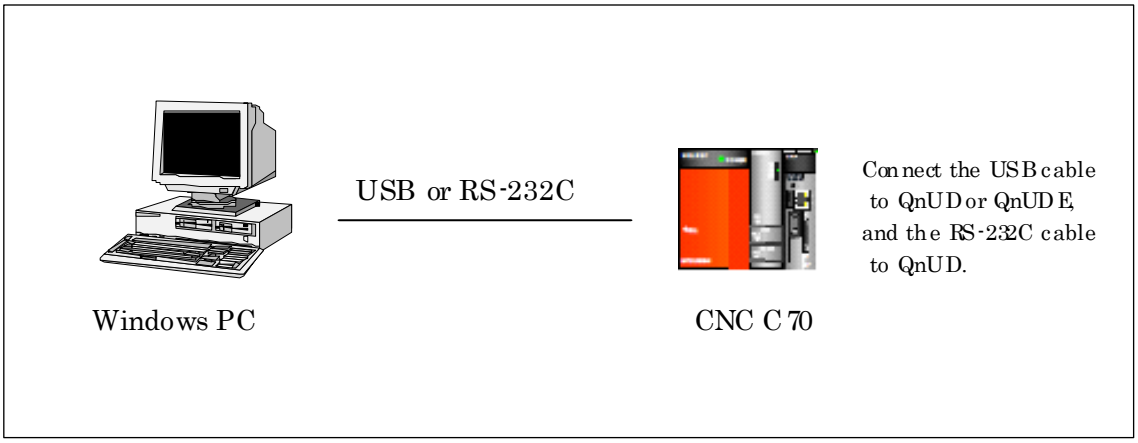


Fig. 1-6 Connection with Mitsubishi CNC C70 Series via QnUD

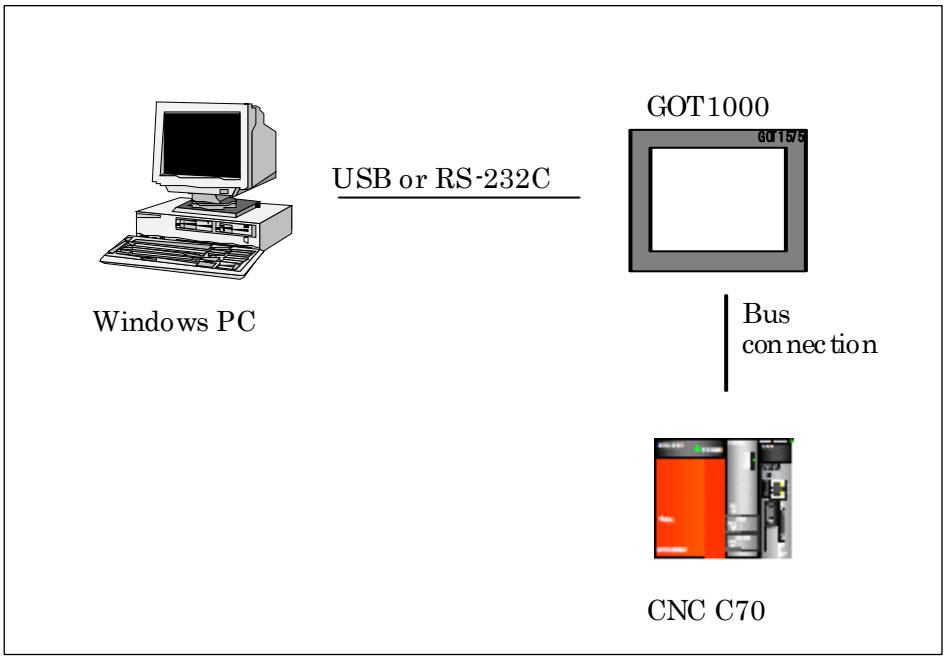


Fig. 1-7 GOT transparent with Mitsubishi CNC C70 Series

1.4 Development Environment

The required environment for development with EZSocket is as follows.

Table 1-2 Development and operation environment

Items	Spec.	
Personal computer	IBM PC / AT compatible (x86 processor)	
CPU	Intel Pentium90MHz or more (Pentium 133MHz or more is recommended)	
Bus	ISA bus or PCI bus when mounting MELDASMAGIC64	
OS	Microsoft WindowsXP Home Edition, Japanese edition/English edition (*3) Microsoft WindowsXP Professional, Japanese edition/English edition (*3) Microsoft Windows2000 SP3 and later, Japanese edition/English edition (*3) Microsoft Windows NT Ver4.0 SP4 and later, Japanese edition/English edition (*1) (*6) (*7) Microsoft WindowsMe, Japanese edition/English edition (*1) (*3) (*6) (*7) Microsoft Windows98SE, Japanese edition/English edition (*1) (*3) (*6) Microsoft Windows 95OSR2 and later, Japanese edition/English edition (*1) (*4) (*6) (*7)	
Memory	In development	96MB or more (128MB or more is recommended)
	In operation	32MB or more (64MB or more is recommended)
Disk space	42MB or more (60MB or more is recommended)	
Peripheral device	Floppy disk drive corresponding to 3.5 inches / 1.44MB format CD-ROM drive	
Programming language	Microsoft Visual C++.NET 2003 (*2) Microsoft Visual C++ Ver.5.0, 6.0 Microsoft Visual Basic Ver.5.0 (*5), 6.0 Microsoft Visual Basic for Applications Ver.5.0 (Equivalent to Excel97VBA) (*5), Ver.6.0 (Equivalent to Excel2000VBA)	
Remark	(*1) Development unavailable with Microsoft Visual C++.NET 2003. (*2) Development must be done by the native code (VC++). Manage code is not available. (*3) Not supported by MELDASMAGIC64 (ISA NC card). (*4) Not supported by MELDAS C6/C64. (*5) As VB5.0 and VBA Ver.5.0 don't support DCOM, they cannot be used for MELDAS M600 series. (*6) Not supported by MITSUBISHI CNC 700 Series and MITSUBISHI CNC C70 series. (*7) Not supported by MELDASMAGIC64(PCI NC card).	

1.5 Installing EZSocket

EZSocket Dynamic Link Library (DLL) is necessary to use EZSocket.
Please refer to "Release note" that shows how to install EZSocket.

1.6 Using EZSocket

When creating applications with VC++, VB, or VBA by using EZSocket, the following include file or module is necessary. Table 1-3 shows the folder and file names when the CD-ROM, EZSocketNc, is installed on C drive.

Table 1-3 List of include files for each development language

	VC++	VB, VBA
Install folder	C:\EZSocket\EZSoketNc\include\Vc	C:\EZSocket\EZSoketNc\include\Vb
File	EZSocketNc.h EZSocketNcStr.h EZSocketNc_i.c EZSocketNcDef.h EZSocketNcErr.h EZSocketCommonErr.h	EZNcDef.bas EZNcErr.bas EZComErr.bas

In order to use EZSocket with Mitsubishi CNC C70 Series, following installation and inclusion are required beforehand; installing EZSocket for MELSEC PLC (EZSocketPc), and including EasysocketDef.h. For redistributable files of EZSocket for MELSEC PLC (EZSocketPc), and how to install, refer to User's Manual of EZSocket for MELSEC PLC.

1.7 EZSocket Interface

EZSocket provides 2 kinds of interfaces as DLL type inprocess server, EZSocket custom interface and EZSocket automation interface. The both interfaces have nearly the same functions as for the data access.
EZSocket custom interface is fit for programming with VC++, and EZSocket automation interface is fit for programming with VB or VBA.
Please choose one of them depending on the programming language you use.
EZSocket interface is based on Microsoft COM (Compact Object Model). When using EZSocket, general knowledge of COM is essential though this manual is not explaining COM.

1.7.1 EZSocket custom interface

Table 1-4 is EZSocket custom interface list.
Interface name has to be "IEZNcXxxx", meaning it is EZSocket custom interface for MELDAS CNC.

Table 1-4 EZSocket custom interface list

Component	Interface	Classification
EZNcCommunication		
	IEZNcCommunication3	Communication
	IEZNcSystem	NC system
	IEZNcPosition	Position
	IEZNcCommand2	Command (Command value)
	IEZNcProgram2	Program
	IEZNcTime	Time
	IEZNcAxisMonitor	Axis monitor
	IEZNcStatus	Status
	IEZNcFile5	File
	IEZNcCommonVariable2	Common variable
	IEZNcLocalVariable2	Local variable
	IEZNcTool3	Tool
	IEZNcATC2	ATC
	IEZNcParameter2	Parameters
	IEZNcOperation	Operation
	IEZNcDevice	PLC device
	IEZNcGeneric2	Generic (for general purpose)
EZNcSubFunction		
	IEZNcSubFunction2	Sub function

(Note 1) Due to upgrade of EZSocket, the interface name may be changed. However, as the new interface succeeds the old interface, the old one remains available.
E.g.) IEZNCFile4→IEZNcFile5
In addition, as the old interface is for backward compatibility, when newly introducing EZSocket, use the new interface.
(Note 2) Mitsubishi CNC C70 Series doesn't support the automation interface.

1.7.2 EZSocket automation interface

Table 1-5 is EZSocket automation interface list.
EZSocket automation interface includes all the functions in one interface, and it makes programming with VB easier.
Interface name has to be "IDispEZNcXxxxx", meaning it is EZSocket automation interface for MELDAS CNC.

Table 1-5 EZSocket automation interface list

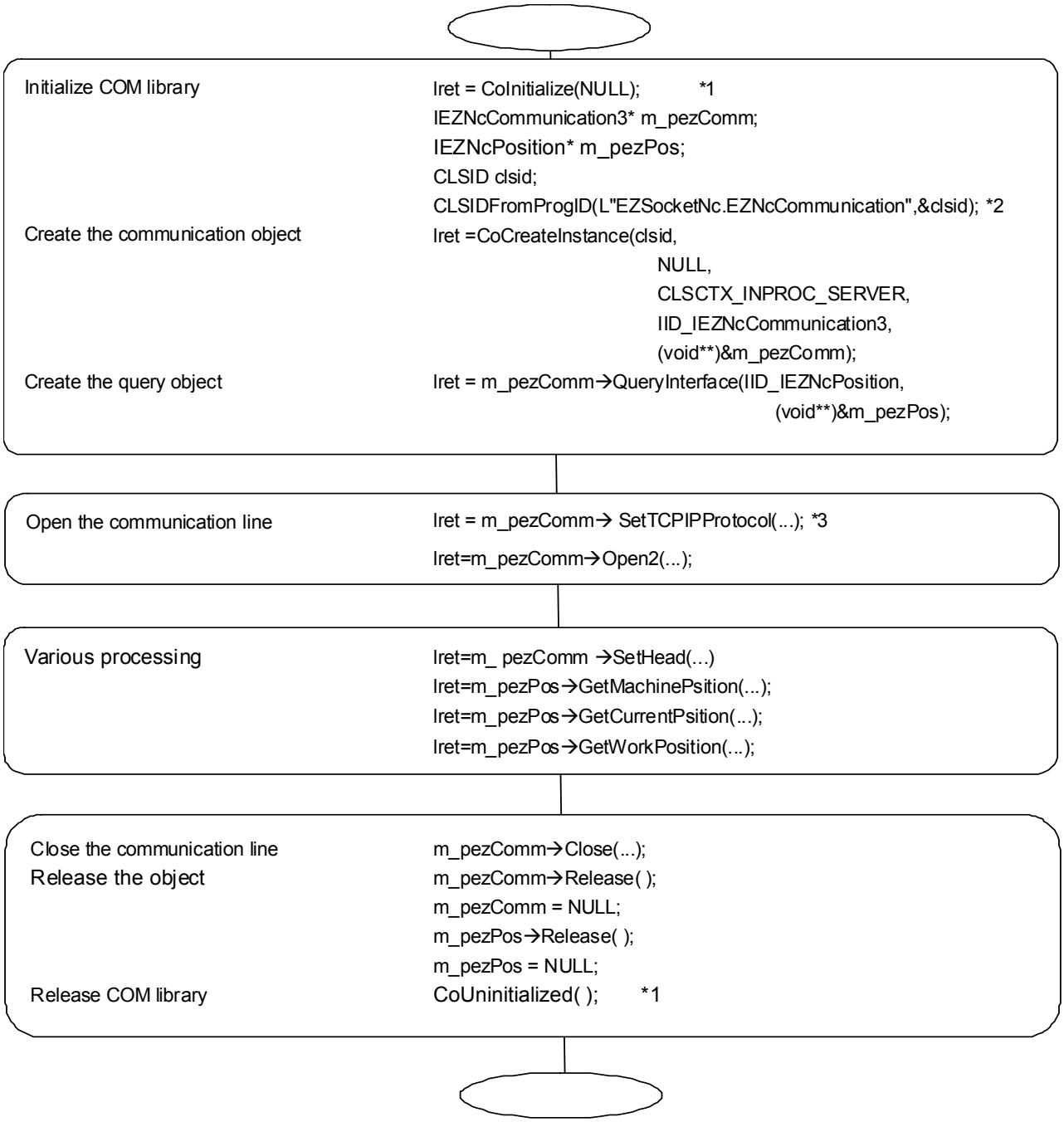
Component	Interface	Classification
DispEZNcCommunication		
	IDispEZNcCommunication	Communication
		NC system
		Position
		Command (Command value)
		Program
		Time
		Axis monitor
		Status
		File
		Common variable
		Local variable
		Tool
		ATC
		Parameters
		PLC device
		Operation
		Generic (for general purpose)
DispEZNcSubFunction		
	IDispEZNcSubFunction	Sub function

(Note) Mitsubhisi CNC C70 Series doesn’t support the automation interface.

1.8 EZSocket Programming Procedure

1.8.1 Programming procedure with VC++ (1)

The following flowchart shows the programming procedure with EZSocket custom interface with VC++ in creating an application that works with the PCs with MELDASMAGIC64 and an application for Mitsubishi CNC 700 Series, MELDAS C6/64 Series, and Mitsubishi CNC C70 Series.



*1 When using EZSocket in the thread, call COM library variable CoInitialize( ) before using EZSocket, and then call CoUninitialize( ) after using EZSocket. After using each object of EZSocket, call Release( ) to release the object (decrement of reference count).

*2 In the case of a program to be connected to MELDAS600 Series, match PROGID with EZSocket embedded in the body of MELDAS600 Series . When designating PROGID, explicitly designate the interface version, too.

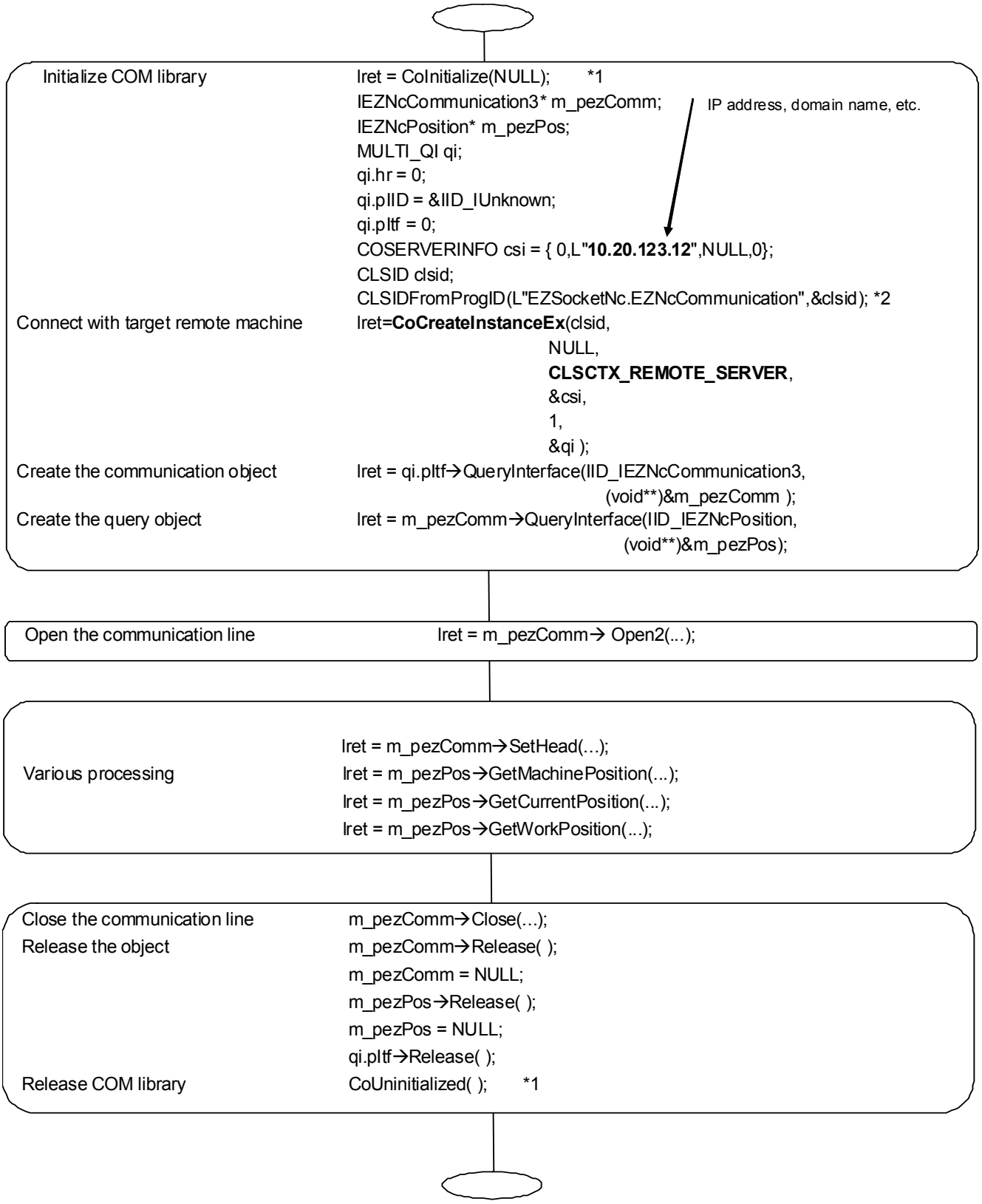
E.g.: "EZSocketNc.EZNcCommunication.7"

As for PROGID of EZSocket embedded in the body of MELDAS600 Series, contact us via your seller of MELDAS600 Series.

*3 When creating an application for CNC700 or C64 Series, call SetTCPIPProtocol before Open. However, when creating an application for C70, do not call SetTCPIPProtocol but SetMelsecProtocol.

1.8.2 Programming procedure with VC++ (2)

The following flowchart shows the programming procedure with EZSocket custom interface with VC++ in creating an application remote-controlled by a PC connected by Ethernet. However, in the case of CNC700, C64 Series or C70, follow the procedure in 1.7.1. even when connected by Ethernet.
To perform this procedure, EZSocket must be installed in the PC connected by Ethernet.



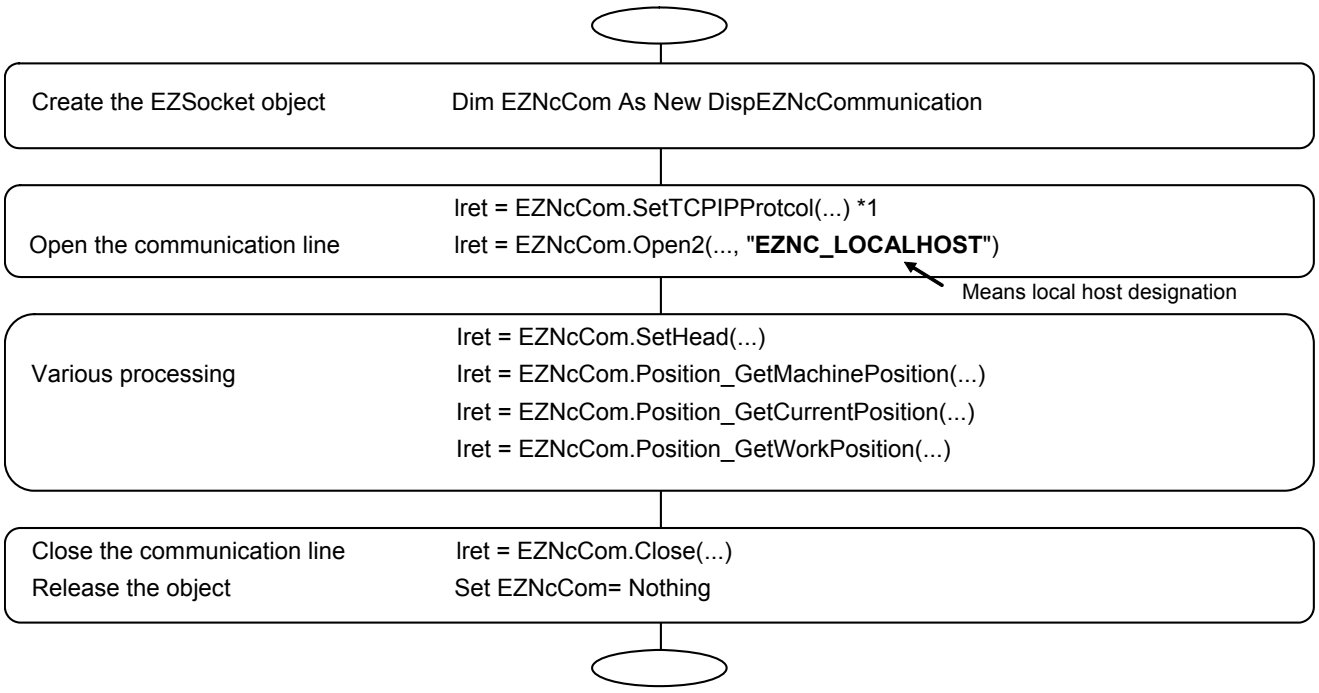
*1 Refer to *1 of 1.8.1.
*2 Refer to *2 of 1.8.1.

1.8.3 Programming procedure with VB (1)

The following flowchart shows the programming procedure with EZSocket automation interface with VB in creating an application that works with the PCs with MELDASMAGIC64 or an application for CNC700 or C64 Series.

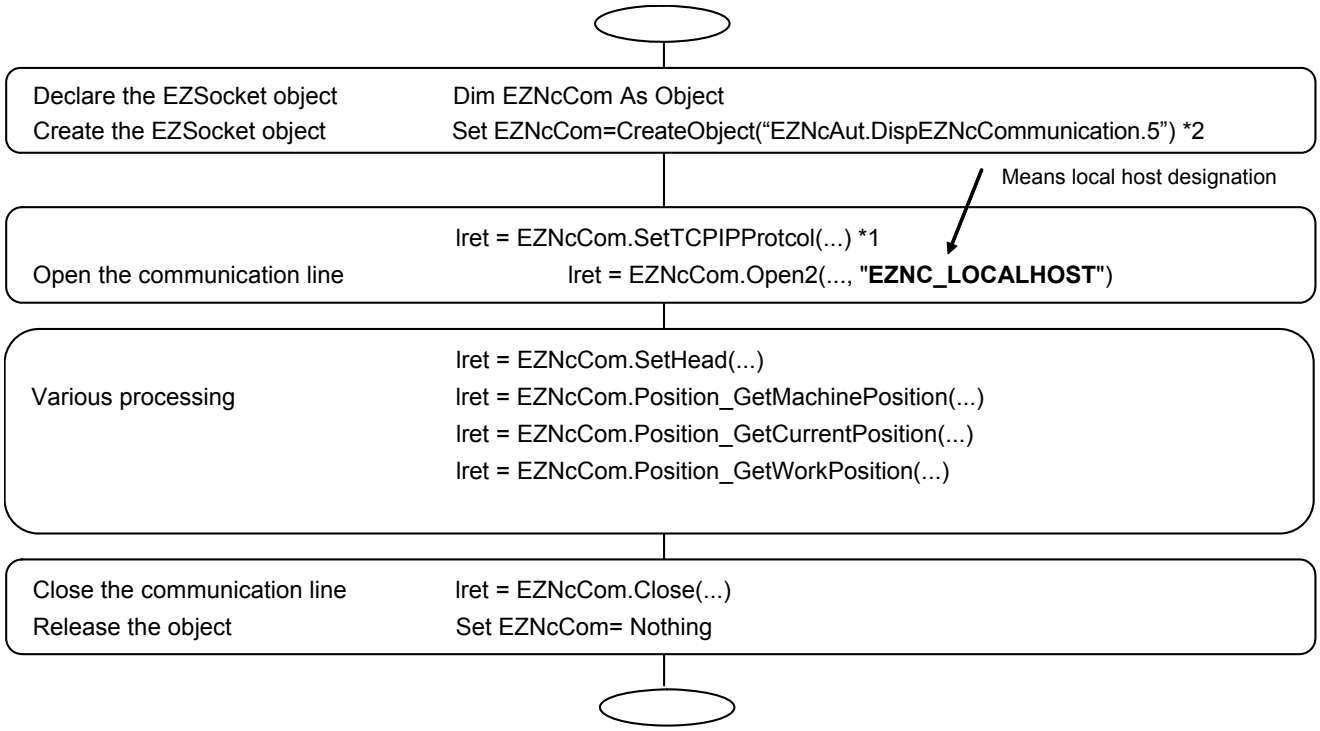
(1) How to call EZSocket by early binding

For early binging, reference setting is necessary for the type library of EZSocket automation interface in advance.



(2) How to call EZSocket by late binding

In late binding, reference setting is not necessary. However, VB's object browser function cannot be used.



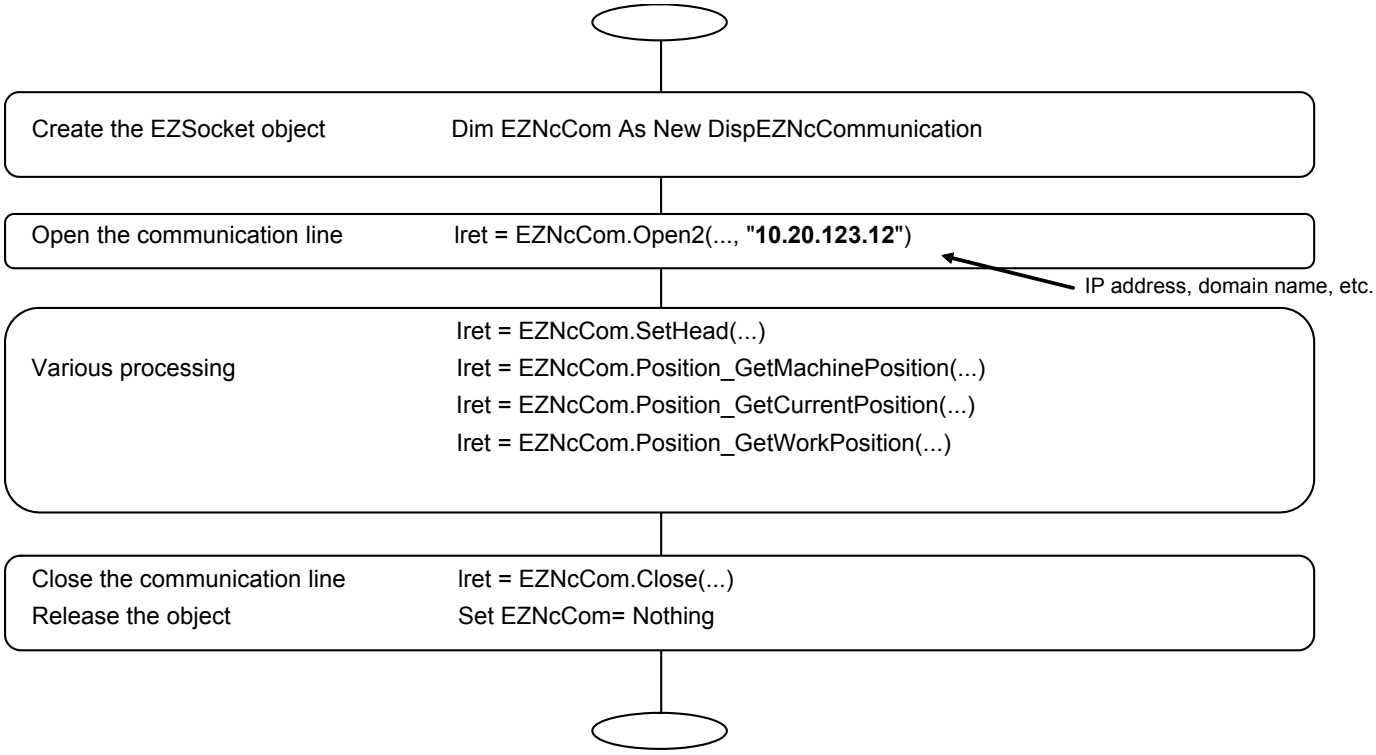
*1 Only when creating an application for CNC700 or C64 Series, call SetTCPIPProtocol before Open.
*2 Refer to *2 of 1.8.1.

1.8.4 Programming procedure with VB (2)

The following flowchart shows the programming procedure with EZSocket automation interface with VC++ in creating an application remote-controlled by a PC connected by Ethernet.

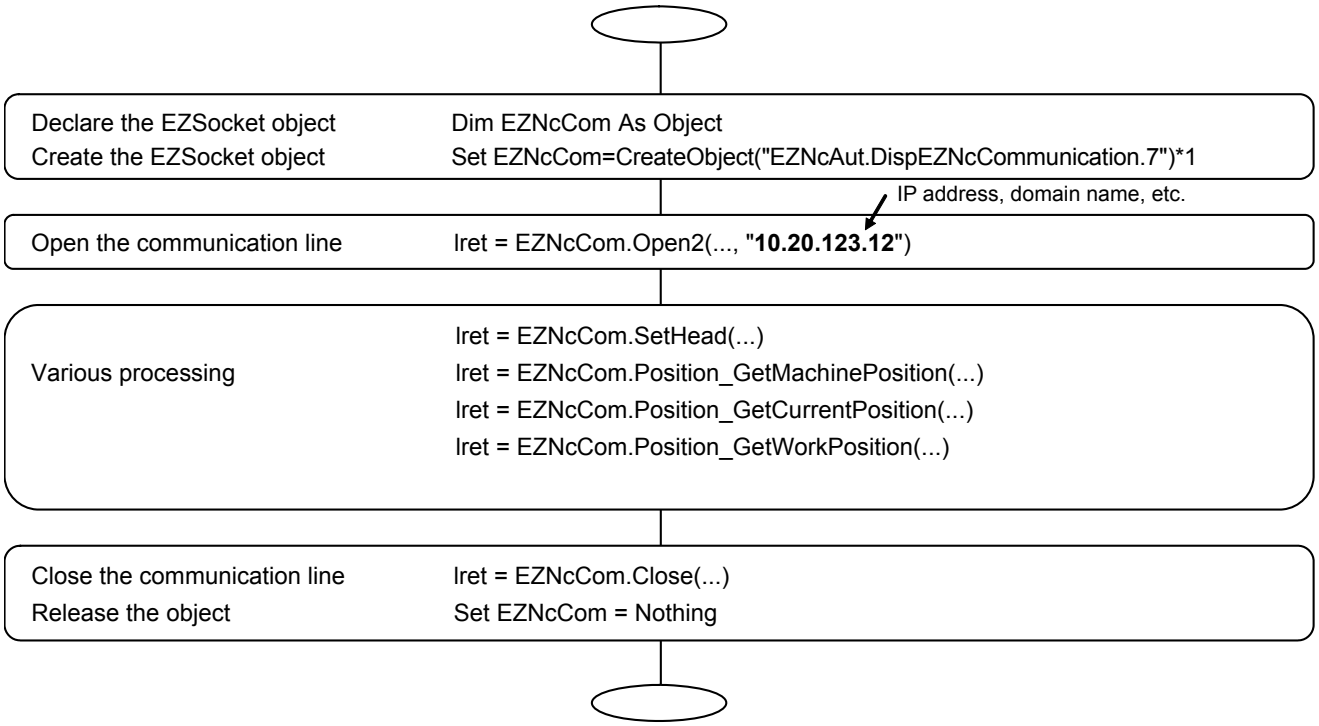
(1) How to call EZSocket by early binding

For early binging, reference setting is necessary for the type library of EZSocket automation interface in advance.



(2) How to call EZSocket by late binding

In late binding, reference setting is not necessary. However, VB's object browser function cannot be used.



*1 Refer to *2 of 1.8.1.

2. DETAILS OF I/F SPECIFICATION

2.1 Common Items

(1) Character Code

UNICODE is used for all character string parameter of EZSocket interface.

(2) How to treat character code

When using character string data, EZSocket allocates the memory when returning the character string data to an application. Make the application release the memory of disused character string data, or it causes memory leak.
When programming application by using EZSocket custom interface with VC++, release the memory of character string region explicitly with **CoTaskMemFree( ).**
When programming application by using EZSocket automation interface with VC++, release the memory of character string region explicitly with **SysFreeString( )**

(3) How to treat error code

Method returns 2 types of return value, (**S_OK**) and (**S_FALSE**).
(S_OK) is returned when exited normally.
(S_FALSE) is returned when failed.
The detailed error code is returned as argument.
The common error messages are below.

EZ_ERR_NOT_OPEN: Communication line is not opened
EZ_ERR_DOUBLE_OPEN: Double open error
EZ_ERR_DATA_TYPE: Data type of the argument is illegal
EZ_ERR_DATA_RANGE: Data range of the argument is illegal
EZ_ERR_NOT_SUPPORT: Not supported
EZ_ERR_NULLPTR: Argument is NULL pointer

(4) How to treat errors when calling the interface that is not supported by the model

When calling the interface that is not supported by the model, **EZ_ERR_NOT_SUPPORT** is output.

(5) Descriptions of section 2.3 and later

Detailed specifications of each method are shown in section 2.3 and later with following Items and contents.

"□Argument": Describes the argument specifications for the method.
"□Return value: Describes the return values to the method.
"□Functions": Describes the function outline of the method.
"□Reference": Describes the related methods.
"□Designation": Describes any required designations of part system #, PLC axis part system # or axis #.
Unless the method is used with the described designations, the operation is not guaranteed.
Required designations are described as follows.
System: Designation of part system # is required. Use **SetHead()** for the designation.
PLC axis: Designation of PLC axis part system # is required. Use **SetHead()** for the designation.
Axis: Designation of axis # is required.

Some methods, such as the one in 2.12.1, may not require the part system designation due to the argument value, although System is shown in the column. See the complements at "□Designation" column of each method.

(6) Restriction in the method that requires part system designation

The result is uncertain when the method is executed with the invalid part system designated. Please confirm that the designated part system is valid before the execution.
In addition, after opening the EZSocket line open, part system 1 is designated.

(7) Designation of a file name

EZSocket treats NC system(*1) as a drive, and treats various kinds of data (including machining programs and tool offsets) on NC system as files. Unless any exceptions are provided, EZSocket uses the following designation of a file name to access a file on NC system:

Drive name+":"+\Directory name+\File name

Always use the absolute path to designate a file name. *2

The correspondence of drive names to NC system #s is as follows.

NC system #	Drive name (NC memory)
01	M01
02	M02
03	M03
:	:
:	:
FF	MFF

*1 Described as "NC card" from next section in this EZSocket manual. "NC card" is referred to as "NC control unit" in other manuals of NC series except Magic64.

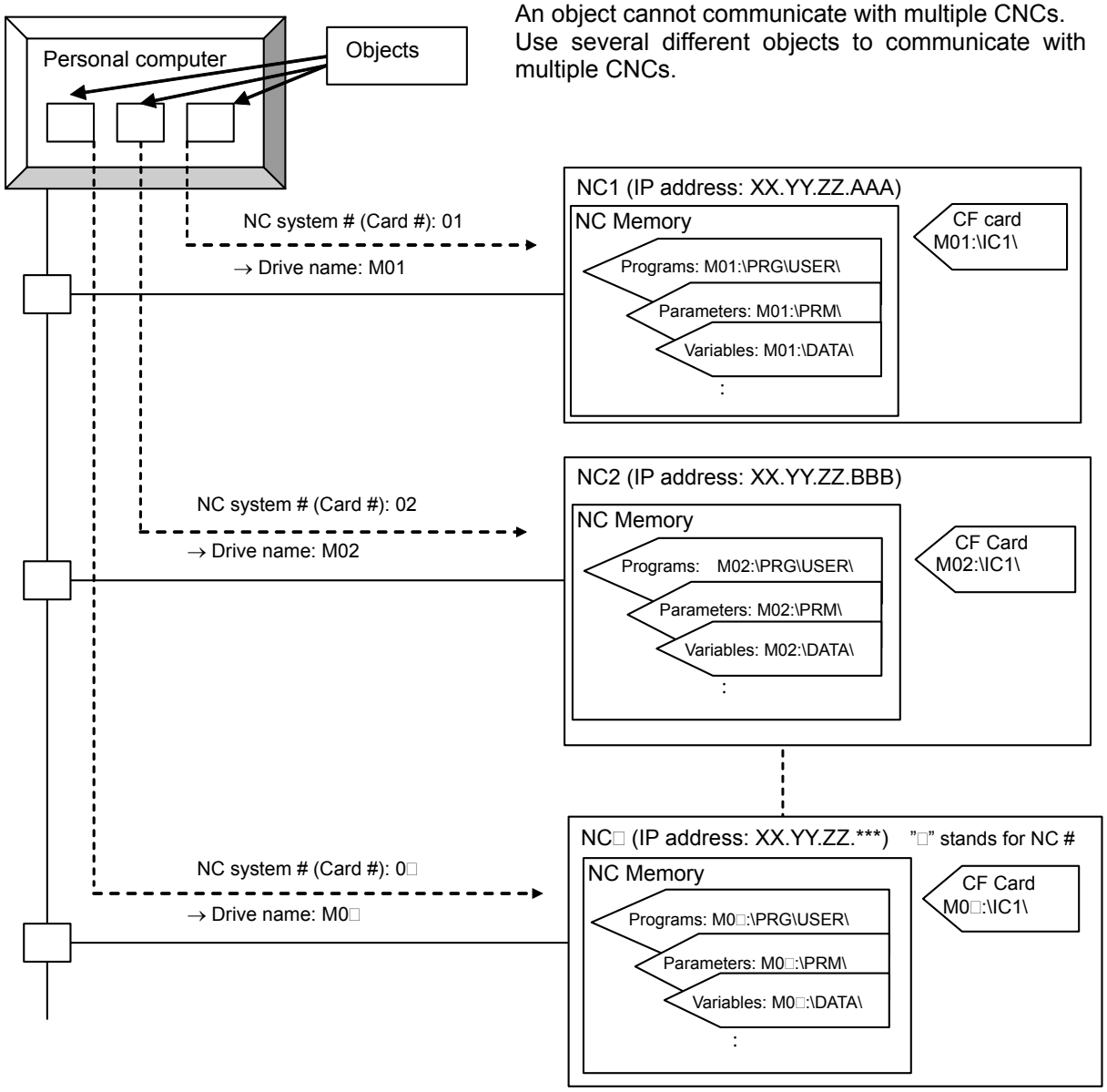
*2 Use capital letters to designate a file name.

(8) Restriction in calling from more than one thread of Mitsubishi CNC C70 Series

In the case of Mitsubishi CNC C70 Series, if the number of threads using EZSocket exceeds 1000, error codes may not be output correctly. In creating an application for Mitsubishi CNC C70 Series, pay attention to the number of threads using EZSocket.

(9) Precaution when connecting to multiple CNCs

An object cannot communicate with multiple CNCs. Use several different objects to communicate with multiple CNCs.



2.2 Method List

Table 2-1 Interface list **A: Available, L: Available with limitation, U: Unavailable**

No	Function class	Interface (Operation)	Function	Availability					
				Magic64	M6x5M	M6x5L	C64	C70*1	CNC700
2.3.1	IEZNc-Communication3 (Communication)	Open2	Opening NC card system line	A	A	A	A	A	A
2.3.2		Close	Closing NC card system line	A	A	A	A	A	A
2.3.3		SetHead	Setting part system	A	A	A	A	A	A
2.3.4		GetHead	Getting the part system	A	A	A	A	A	A
2.3.5		SetTCPIPProtocol	Setting TCPIP communication	U	U	U	A	U	A
2.3.6		SetMelsecProtocol	Setting MELSEC communication	U	U	U	U	A	U
2.3.7		SetModalCondition	Setting modal communication condition	U	U	U	A	U	U
2.3.8		GetModalCondition	Getting modal communication condition	U	U	U	A	U	U
2.4.1	IEZNcSystem (NC system)	GetVersion	Getting the system #, name and control S/W version	L	A	A	A	A	A
2.4.2		GetSystemInformation	Getting the NC system information	A	A	A	A	A	A
2.4.3		GetAlarm	Getting the alarm information	A	A	A	A	A	A
2.5.1	IEZNcPosition (Position)	GetWorkPosition	Getting the workpiece coordinate position (Skip ON supported)	A	A	A	A	A	A
2.5.2		GetMachinePosition	Getting the machine position (Skip ON supported)	A	A	A	A	A	A
2.5.3		GetCurrentPosition	Getting the current position	A	A	A	A	A	A
2.5.4		GetDistance	Getting the remaining distance (Skip ON supported)	A	A	A	A	A	A
2.5.5		GetNextDistance	Getting the next travel distance	A	A	A	A	A	A
2.5.6		GetFeedSpeed	Getting the feedrate	A	A	A	A	A	A
2.5.7		GetManualOverlap	Getting the manual interruption amount	A	A	A	A	A	A
2.5.8		GetProgramPosition	Getting the program position	U	A	A	A	A	A
2.6.1	IEZNcCommand2 (Command)	GetGCodeCommand	Getting the G-code modal command value	A	A	A	A	A	A
2.6.2		GetToolCommand	Getting the tool offset #	A	A	U	A	A	A
2.6.3		GetFeedCommand	Getting the feedrate command value	A	A	A	A	A	A
2.6.4		GetCommand2	Getting the M/S/T/B modal command value	A	A	A	A	A	A
2.6.5		SetCommand2	Setting manual command (M, S, T, B)	L	A	A	L	A	A
2.7.1	IEZNcProgram2 (Program)	CurrentBlockRead	Reading the current block	A	A	A	A	A	A
2.7.2		GetProgramNumber2	Getting the program # (main, sub)	A	A	A	A	A	A
2.7.3		GetSequenceNumber	Getting the sequence # (main, sub)	A	A	A	A	A	A
2.7.4		GetBlockNumber	Getting the block # (main, sub)	A	A	A	A	A	A
2.7.5		GetSubProLevel	Getting the sub program calling level	A	A	A	A	A	A
2.7.6		GetInformation	Getting the program information	A	A	U	A	A	A
2.8.1	IEZNcTime (Time)	GetClockData	Getting the date/time	A	A	L	A	A	A
2.8.2		SetClockData	Setting date/time	A	A	A	A	A	A
2.8.3		GetAliveTime	Getting the power ON time	A	A	A	A	A	A
2.8.4		SetAliveTime	Setting power ON time	A	A	A	A	A	A
2.8.5		GetRunTime	Getting the automatic operation time	A	A	A	A	A	A
2.8.6		SetRunTime	Setting automatic operation time	A	A	A	A	A	A
2.8.7		GetStartTime	Getting the automatic start time	A	A	A	A	A	A
2.8.8		SetStartTime	Setting automatic start time	A	A	A	A	A	A
2.8.9		GetEstimateTime	Getting the external elapsed time (1, 2)	A	A	A	A	A	A
2.8.10		SetEstimateTime	Setting external elapsed time (1, 2)	A	A	A	A	A	A

No	Function class	Interface (Operation)	Function	Availability					
				Magic64	M6x5M	M6x5L	C64	C70 *1	CNC700
2.9.1	IEZNcAxisMonitor	GetServoMonitor	Servo monitor	L	A	A	A	U	A
2.9.2	(Axis monitor)	GetServoVersion	Getting the servo information	U	A	A	A	U	A
2.9.3		GetServoDiagnosis	Getting the servo diagnosis information	U	A	A	A	U	A
2.9.4		GetPowerVersion	Getting the power supply version information	U	A	A	A	U	A
2.9.5		GetPowerDiagnosis	Getting the power supply diagnosis information	U	A	A	A	U	A
2.9.6		GetSpindleMonitor	Spindle monitor (Function limited for Magic64)	L	A	A	A	U	A
2.9.7		GetSpindleVersion	Getting the spindle version information	U	A	A	A	U	A
2.9.8		GetSpindleDiagnosis	Getting the spindle diagnosis information	U	A	A	A	U	A
2.9.9		GetAbsPositionMonitor	Getting the absolute position monitor information	U	A	A	A	U	A
2.9.10		GetAuxAxisMonitor	Getting the auxiliary axis monitor information	U	A	U	A	U	A
2.9.11		GetAuxAxisDiagnosis	Getting the auxiliary axis diagnosis information	U	A	U	A	U	A
2.9.12		GetAuxAxisVersion	Getting the auxiliary axis version information	U	A	U	A	U	A
2.9.13		GetDowelTime	Getting the remaining dwell time	A	A	A	A	U	A
2.10.1	IEZNcRunStatus	GetInvalidStatus	Getting the invalid status	A	A	A	A	A	A
2.10.2	(Status)	GetCommandStatus	Getting the operation command status	A	U	U	A	A	A
2.10.3		GetCuttingMode	Getting the cutting mode	A	A	A	A	A	A
2.10.4		GetAxisStatus	Getting the axis status	A	A	A	A	A	A
2.10.5		GetRunStatus	Getting the operation status	A	A	A	A	A	A
2.11.1	IEZNcFile5	FindDir	Finding directory	A	A	A	A	A	A
2.11.2	(File)	FindNextDir	Finding the next directory	A	A	A	A	A	A
2.11.3		ResetDir	Finishing the directory search	A	A	A	A	A	A
2.11.4		Copy2 *2	File copy	A	A	A	A	A	A
2.11.5		Delete2 *2	Deleting file	A	A	A	A	A	A
2.11.6		Rename2 *2	Renaming file	A	A	A	A	A	A
2.11.7		GetDriveInformation	Getting the drive information	A	A	A	A	A	A
2.11.8		GetDriveSize	Getting the drive free capacity	A	A	A	A	A	A
2.11.9		OpenFile3 *3	Opening file	A	A	A	A	A	A
2.11.10		CloseFile2 *2*3	Closing file	A	A	A	A	A	A
2.11.11		AbortFile2 *2	Closing file compulsorily	A	A	A	A	A	A
2.11.12		ReadFile2 *2	Reading the file	A	A	A	A	A	A
2.11.13		WriteFile	Writing file	A	A	A	A	A	A
2.11.14		OpenNCFile2 *2	Opening NC program file	A	A	A	A	U	A
2.11.15		CloseNCFile2 *2	Closing NC program file	A	A	A	A	U	A
2.11.16		AbortNCFile2 *2	Closing NC program file compulsorily	A	A	A	A	U	A
2.11.17		ReadNCFile2 *2	Writing NC program file	A	A	A	A	U	A
2.11.18		WriteNCFile	Reading the NC program file	A	A	A	A	U	A

No	Function class	Interface (Operation)	Function	Availability					
				Magic64	M6x5M	M6x5L	C64	C70*1	CNC700
2.12.1	IEZNcCommon-Variable2 (Common variable)	CommonVRead	Reading the common variables (#100, #500)	A	A	A	A	A	A
2.12.2		CommonVWrite	Writing common variables (#100, #500)	A	A	A	A	A	A
2.12.3		GetSize	Getting the the number of common variable sets (#100, #500)	A	A	A	A	A	A
2.12.4		GetName	Getting the common variable name (#500 to #519)	A	A	A	A	A	A
2.12.5		SetName	Setting common variable name (#500 to #519)	A	A	A	A	A	A
2.12.6		GetCVNullData	Getting the common variable null data	A	U	A	A	A	A
2.13.1	IEZNcLocalVar-iable2 (Local variable)	LocalVRead	Reading the local variable	A	A	A	A	A	A
2.13.2		GetMacroLevel	Getting the macro sub program execution level (Level 0 to 4)	A	A	A	A	A	A
2.13.3		GetLVNullData	Getting the local variable null data	A	U	A	A	A	A
2.14.1	IEZNcTool3 (Tool)	GetToolSetSize	Getting the the number of tool offset sets	A	A	A	A	U	A
2.14.2		GetType	Getting the tool offset type	A	A	A	A	U	A
2.14.3		GetOffset	Getting the tool offset value data	A	A	A	A	U	A
2.14.4		SetOffset	Setting tool offset value data	A	A	A	A	U	A
2.14.5		GetToolWorkOffset	Getting the workpiece coordinate system offset (#54 to 60)	A	A	A	A	U	A
2.14.6		SetToolWorkOffset	Setting workpiece coordinate system offset (#54 to 60)	A	A	A	A	U	A
2.14.7		GetSurface	Getting the reference surface level	A	A	U	A	U	A
2.14.8		SetSurface	Setting reference surface level	A	A	U	A	U	A
2.14.9		GetToolLifeType2	Getting the tool life management type 2	U	A	A	A	U	A
2.14.10		SetToolLifeType2	Setting tool life management type 2	U	A	A	A	U	A
2.14.11		GetToolLifeGroupList	Getting the tool life management group # list	U	A	U	A	U	A
2.14.12		AddToolLifeGroup	Adding tool life management group #	U	A	U	U	U	U
2.14.13		ChangeToolLifeGroup	Changing tool life management group #	U	A	U	A	U	A
2.14.14		DeleteToolLifeGroup	Deleting tool life management group #	U	A	U	A	U	A
2.14.15		GetToolLifeToolNoList	Getting the tool list in the tool life management group	U	A	U	A	U	A
2.14.16		AddToolLifeToolNo	Adding tool # in the tool life management group	U	A	U	A	U	A
2.14.17		ChangeToolLifeToolNo	Changing tool # of tool life management	U	A	U	A	U	A
2.14.18		DeleteToolLifeToolNo	Deleting tool # of tool life management	U	A	U	A	U	A
2.14.19		GetToolLifeValue	Getting the tool life management data	U	A	A	A	U	A
2.14.20		SetToolLifeValue	Respective setting of tool life management data	U	A	A	A	U	A
2.14.21		SetToolLifeValue2	Setting tool life management data	U	A	A	A	U	A
2.14.22		GetSpareTool	Getting the spare tool for tool life management	U	U	A	U	U	U
2.14.23		SetSpareTool	Setting spare tool for tool life management	U	U	A	U	U	U

No	Function class	Interface (Operation)	Function	Availability					
				Magic64	M6x5M	M6x5L	C64	C70[*1]	CNC700
2.15.1	IEZNcATC2 (ATC)	GetMGNControl	Getting the control parameter for ATC tool register	A	A	U	A	U	A
2.15.2		GetMGNSize	Getting the total number of the ATC magazine pot sets	A	A	U	A	U	A
2.15.3		GetMGNSize2	Getting the number of pots of each ATC magazine	U	A	U	A	U	A
2.15.4		GetMGNReady	Getting the ATC ready tool #	A	A	U	A	U	A
2.15.5		GetMGNPot	Getting the tool # of the ATC magazine pot	A	A	U	A	U	A
2.15.6		GetMGNPot3 [*4]	Getting the tool # of each ATC magazine pot	U	A	U	A	U	A
2.15.7		SetMGNPot	Setting tool # of the ATC magazine pot	A	A	U	A	U	A
2.15.8		SetMGNPot3 [*4]	Setting tool # of the pot of each ATC magazine	U	A	U	A	U	A
2.15.9		GetMGNPotEx	Getting the tool # of the ATC extended magazine pot	A	U	U	U	U	U
2.15.10		SetMGNPotEx	Setting tool # of the ATC extended magazine pot	A	U	U	U	U	U
2.15.11		GetMGNAux	Getting the ATC user PLC interface	A	A	U	A	U	A
2.15.12		SetMGNAux	Setting ATC user PLC interface	A	A	U	A	U	A
2.16.1	IEZNcParameter2 (Parameter)	GetParameterData	Getting the parameter	L[*5]	A	A	A	A	A
2.16.2		SetParameterData	Setting parameter	L[*5]	A	A	A	A	A
2.17.1	IEZNcOperation (Operation)	Search	Search	A	A	A	A	A	A
2.17.2		Run	Starting PLC program	A	A	A	A	A	A
2.17.3		Stop	Stopping PLC program	A	A	A	A	A	A
2.18.1	IEZNcDevice (Device)	SetDevice	Setting device	A	A	A	A	U	A
2.18.2		DeleteDeviceAll	Deleting all device setting	A	A	A	A	U	A
2.18.3		ReadDevice	Reading the device	A	A	A	A	U	A
2.18.4		WriteDevice	Writing device	A	A	A	A	U	A
2.19.1	IEZNcGeneric2 (Generic)	ReadData	Reading the generic data	A	A	A	A	U[*6]	A
2.19.2		WriteData	Writing generic data	A	A	A	A	U[*6]	A
2.19.3		SetData	Setting data	A	A	A	A	A	A
2.19.4		DeleteDataAll	Deleting all data setting	A	A	A	A	A	A
2.19.5		ReadBlockData	Batch reading of data	A	A	A	A	A	A
2.19.6		WriteBlockData	Batch writing of data	A	A	A	A	A	A
2.20.1	IEZNcSubFunction 2(Sub function)	ChangeInit	Initializing sub function	A	A	A	A	U	A
2.20.2		GetToolLifeValueOfFile	Getting the tool life management data of tool life management file	U	A	A	U	U	U
2.20.3		SetToolLifeValueOfFile	Setting the tool life management data of tool life management file	U	A	A	U	U	U
2.20.4		GetToolLifeValueOfFile2	Getting tool life management data of tool life management file	U	A	A	A	U	U
2.20.5		SetToolLifeValueOfFile2	Setting tool life management data of tool life management file	U	A	A	A	U	U
2.20.6		GetSpareToolOfFile	Getting tool exchange data of tool life management tile	U	U	A	U	U	U
2.20.7		SetSpareToolOfFile	Setting tool exchange data of tool life management tile	U	U	A	U	U	U
2.20.8		GetToolWorkOffsetOfFile	Getting the workpiece coordinate system offset data of workpiece offset file	A	A	A	A	U	A
2.20.9		SetToolWorkOffsetOfFile	Setting workpiece coordinate system offset data of workpiece offset file	A	A	A	A	U	A

*1) Mitsubishi CNC C70 Series doesn’t support the automation interface.
*2) In this method, modifications have been added to the functions supporting MELDASC6/C64 Series. Conventional method is available to get the backward compatilibity.
*3) In this method, a file overwrite function has been added for Mitsubishi CNC700 Series. Conventional method remains available to get the backward compatilibity.).
*4) In this method, modifications have been added to the functions supporting Mitsubishi CNC 700 Series. Conventional method is available to get the backward compatilibity.
*5) Unavailable with PCI NC card. ISA NC card limits the function. Contact MITSUBISHI before using the card.
*6) Mitsubishi CNC C70 Series doesn't support this method, but reading and writing general-purpose data is enabled by use of ReadBlockData and WriteBlockData.

2.3 IEZNcCommunication3 Interface

Magic64	M6x5M	M6x5L	C64	C70	CNC700

2.3.1 IEZNcCommunication3::Open2 — Opening line

☐Calling procedure (Custom interface)

```
HRESULT     Open2 (
                LONG lSystemType,      //  (I)  Setting NC system type
                LONG lMachine,         //  (I)  Setting NC card
                LONG lTimeOut          //  (I)  Setting communication timeout period
                LONG* plRet            //  (O)  Error code
                )
```

☐Calling procedure (Automation interface)

```
Open2(
    lSystemType As LONG        //  (I)  Setting NC system type
    lMachine As LONG           //  (I)  Setting NC card
    lTimeOut As LONG           //  (I)  Setting communication timeout period
    bstrHostName As STRING     //  (I)  Host name
    ) As LONG                  //  (O)  Error code
```

☐Argument

lSystemType: Set NC system type

Value	Meaning
EZNC_SYS_MELDAS6x5M	Starts connecting with M6x5M
EZNC_SYS_MELDAS6x5L	Starts connecting with M6x5L
EZNC_SYS_MAGICBOARD64	Starts connecting with Magic64
EZNC_SYS_MELDASC6C64	Starts connecting with C64
EZNC_SYS_MELDASC70	Starts connecting with C70
EZNC_SYS_MELDAS700M	Starts connecting with CNC700M
EZNC_SYS_MELDAS700L	Starts connecting with CNC700L

lMachine: Set NC card #

When connecting to more than one NC, designate different NC card No. for each NC. (Number: 1 to 255)

lTimeOut: Set communication timeout period. However, with Magic64 and C70, this value is invalid.

Magic64 communication timeout period is 180 seconds fixed.Set C70's communication timeout period with lTimeOut of SetMelsecProtocol().

Value	Meaning
1 to 3000	Timeout amount (Unit: 100ms) (With C64 or CNC700, set the timeout value to 10 or higher. If timeout error occurs, increase the value.)

bstrHostName: Set the host name of the NC system to connect. IP address setting is available. To connect to the local host, set **"EZNC_LOCALHOST"** as a character string. In the case of C64 or CNC700, always set **"EZNC_LOCALHOST"**.

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK: Normal termination
EZ_ERR_DATA_TYPE: Data type of the argument is illegal
EZNC_SYSFUNC_IOCTL_ADDR_: NC card # is illegal
EZNC_SYSFUNC_IOCTL_NOTOPEN: Device not open
EZNC_SYSFUNC_IOCTL_DATA: Open parameter data range is illegal
EZNC_COMM_NOTSETUP_PROTOCOL: TCP/IP communication has not been set (Only in the case of C64 or CNC700.)
EZNC_COMM_NOTMODULE: Submodules don't exist
EZNC_COMM_CREATEPC: Impossible to create an EZSocketPc object (Only in the case of C70)

☐ **Return value**	Return value	Meaning
	S_OK	Normal termination
	S_FALSE	Communication failure
☐ **Functions**	This starts connecting the line of **IEZNcCmmunication3** object. In the case of C64 or CNC700, make sure to perform communication setting by executing SetTCPIPProtocol() before Open. In addition, in the case of C70, before carrying out Open, make sure to execute SetMelsecProtocol() for communication setting. If not, an error occurs at executing Open2(). (Note 1) Mitsubishi CNC C70 Series doesn't support the automation interface. (Note 2) The existing interface IEZNcCommunication2::Open() continues to be available for backward compatibility. When newly introducing EZSocket, use this method.	
☐ **Reference**	**Close(),** **SetTCPIPProtocol(),** **SetMelsecProtocol()**	
☐ **Designation**		

2.3.2 IEZNcCommunication3::Close — Closing line

☐**Calling procedure (Custom interface)**

```
HRESULT        Close(
                   LONG* plRet        //   (O)   Error code
                   )
```

☐**Calling procedure (Automation interface)**

```
               Close( ) As LONG       //   (O)   Error code
```

☐**Argument**

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK : Normal termination
EZNC_SYSFUNC_IOCTL_NOTOPEN: Device not open

☐**Return value**

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐**Functions**

This closes the **IEZNcCmmunication** object line that was opened by **Open2().**
In the case of C64 or CNC700, the setting of **SetTCPIPProtocol**() can be held, so **Open2()** is possible repeatedly. In the case of C70, the setting of SetMelsecProtocol() can be held, so Open2() is possible repeatedly.

☐**Reference**

Open2(),
SetTCPIPProtocol(),
SetMelsecProtocol()

☐**Designation**

2.3.3 IEZNcCommunication3::SetHead — Setting part system

☐Calling procedure (Custom interface)

```
HRESULT        SetHead(
                   LONG lHead,        //  (I)  Setting part system
                   LONG* plRet        //  (O)  Error code
                   )
```

☐Calling procedure (Automation interface)

```
               SetHead(
                   lHead As LONG      //  (I)  Setting part system
                   )As LONG           //  (O)  Error code
```

☐Argument

lHead: Set part system or PLC axis part system. Value: The range of value depends on NC specifications (or options) and the values (for parameters) set by the machine tool builder. When the value is 0, it means that no part system is designated. To set PLC axis part system, set **EZNC_PLCAXIS**. In the case of **M6x5M/L** and **Magic64**, PLC axes are invalid.

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK : Normal termination

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This sets the part system of NC axis or PLC axis. Set part system before executing the method that requires the part system setting. The part system set here will be valid until it is changed. After opening the EZSocket line, part system 1 is designated.

(Note) Mitsubishi CNC C70 Series doesn't support the automation interface.

☐Reference

GetHead()

☐Designation

2.3.4 IEZNcCommunication3::GetHead — Getting the part system

☐Calling procedure (Custom interface)

```
HRESULT        GetHead(
                   LONG* plHead,      //  (O)  Getting the part system
                   LONG* plRet        //  (O)  Error code
                   )
```

☐Calling procedure (Automation interface)

```
               GetHead(
                   plHead As LONG*    //  (O)  Getting the part system
                   )As LONG           //  (O)  Error code
```

☐Argument

plHead: Returns the part system number or the PLC axis part system. Value: The range of the value depends on NC specifications (or options) and the values (for parameters) set by the machine tool builder. In the case of PLC axis part system, **EZNC_PLCAXIS** is returned. In the case of **M6x5M/L** and **Magic64**, PLC axes are invalid.

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK : Normal termination

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This gets the part system of NC axis or PLC axis. In the case of PLC axis part system, **EZNC_PLCAXIS** is got.

(Note) Mitsubishi CNC C70 Series doesn't support the automation interface.

☐Reference

SetHead()

☐Designation

2.3.5 IEZNcCommunication3::SetTCPIPProtocol — Setting TCP/IP communication

☐Calling procedure (Custom interface)

```
HRESULT        SetTCPIPProtocol (
                   LPCOLESTR lpcwszIPAddress,  //  (I)  IP address
                   LONG lPort,                 //  (I)  Port #
                   LONG* plRet                 //  (O)  Error code
                   )
```

☐Calling procedure (Automation interface)

```
               SetTCPIPProtocol (
                   lpcwszIPAddress As STRING   //  (I)  IP address
                   lPort As LONG               //  (I)  Port #
                   )As LONG                    //  (O)  Error code
```

☐Argument

lpcwszIPAddress: Set the IP address or host name of C64 or CNC700 to connect (E.g. "192.168.1.1", "host123")

lPort: Set the port # of C64 or CNC700 to connect. To determine the port #, check the setting of the NC system.
Port # 64758 is set for C64 and 683 is set for CNC700.

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_COMM_ALREADYOPENED: Setting impossible as the line is already open
EZ_ERR_DATA_RANGE: Data range of the argument is illegal
EZ_ERR_NOT_SUPPORT: Not supported

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This is for setting TCP/IP communication.
With C64 or CNC700, call this before **Open2()**. If not, an error occurs at executing **Open2()**.
The setting set here will be held until the object is released by **Release()**.
Once **Open2()** is executed, resetting by **SetTCPIPProtocol()** is impossible until **Close()**, an error occurs.

☐Reference

Open2(), Close()

☐Designation

2.3.6 IEZNcCommunication3::SetMelsecProtocol — Setting Melsec communication

☐Calling procedure (Custom interface)

```
HRESULT     SetMelsecProtocol (
                EZNCST_OPEN* pstOpen,     //  (I)   Sets line
                LONG* plRet               //  (O)   Error code
                )
```

☐Argument

pstOpen : Pointer for indicating the **EZNCST_OPEN** structure which specifies the opening parameters. Refer below for members of the **EZNCST_OPEN** structure.

(Note) The structure supports other communications than Ethernet communication and serial communication. For unnecessary parts in the following explanation on the structure members, set “0”.

lNetworkNumber : Set the network number on MELSECNET/H. Set "0x00" to set the host station. Set as described below for Qn multidropping (via serial communication/CC-Link unit).

Value	Meaning
0x00	Set the host network
0x01	Set the other network at multidrop destination

lStationNumber : Set the station number on MELSECNET or CC-Link. Set "0xFF" to set the host station. The station is handled as a host station when access to the CPU connected with the CPU board or AF board is made.
Set as described below for Qn multidropping (via serial communication/CC-Link unit)

Value	Meaning
0x00	Set the host network
0x01	Set the other network at multidrop destination

lUnitNumber : Set the unit number of the computer link (serial communication) unit or the station number when the target is the Qn communication series intelligent special unit. Set "0x00" to set the QnA series host station (unit loaded to the host station CPU). This number is invalid for other than computer link communication (serial communication) or when the target is not the Qn intelligent special unit. For multidrop link, set the unit number of the destination computer link (serial communication) unit station.

lConnectUnitNumber : Set the unit number of the computer link (serial communication) unit or QnA/Qn Ethernet unit. For multidrop link, set the unit number of the request source computer link (serial communication) unit station. For multidrop via the directly connected CPU, however, the unit number of the request source station is not needed (set "0x00"). Set "0x00" for other than multidrop link. Set the relay destination station number for the QnA/Qn Ethernet unit. (The number is fixed to "0x00" for access within the host network.) For access to the other network via MELSECNET/10, set the station number set in the parameter of the connected Ethernet unit.

lIONumber : Set the unit I/O number. In this parameter, set the actual I/O number (first I/O number÷16) of the destination serial communication unit or intelligent special unit for multidrop link execution or access to the intelligent special unit. **(For multidrop link, set the I/O number of the relayed station:request source station.)** Set any of 0x3F0 to 0x3FF for access to another station via the host station CPU or network.

Value	Destination
0 to 1FFh	Communication series intelligent special unit (first I/O number÷16)
200 to 3CFh	Reserved
3D0h	Control system CPU unit
3D1h	Standby system CPU unit
3D2h	System A CPU unit
3D3h	System B CPU unit
3D4h	Other System CPU unit
3D5h to 3DBh	Reserved
3DCh	System A peripheral device1
3DDh	System B peripheral device1
3DEh	System A peripheral device2
3DFh	System B peripheral device2
3E0 to 3E3h	Each CPU unit in multi-CPU (1 to 4)
3F0h	Global unit for multi-CPU
3FCh	CPU-adjacent card
3FDh	Peripheral device 2
3FEh	Peripheral device 1
3FFh	CPU unit (including LM)

lCpuType : Set the CPU to communicate with

Value	Destination	Value	Destination
CPU_A0J2HCPU	A0J2H, FR-C500	**CPU_Q02CPU**	Q02HCPU
CPU_A1FXCPU	A1FX	**CPU_Q06CPU**	Q06HCPU
CPU_A1SCPU	A1S, A1SJ	**CPU_Q12CPU**	Q12HCPU
CPU_A1SHCPU	A1SH, A1SJH	**CPU_Q25CPU**	Q25HCPU
CPU_A1NCPU	A1(N)		
CPU_A2CCPU	A2C, A2CJ	**CPU_Q00JCPU**	Q00JCPU
CPU_A2NCPU	A2(N), A2S	**CPU_Q00CPU**	Q00CPU
CPU_A2SHCPU	A2SH	**CPU_Q01CPU**	Q01CPU
CPU_A3NCPU	A3(N)	**CPU_Q12PHCPU**	Q12PHCPU
CPU_A3HCPU	A3H	**CPU_Q25PHCPU**	Q25PHCPU
CPU_A2ACPU	A2A	**CPU_Q12PRHCPU**	Q12PR H
CPU_A3ACPU	A3A	**CPU_Q25PRHCPU**	Q25PRH
CPU_A2UCPU	A2U, A2US	**CPU_Q02UCPU**	Q02UCPU
CPU_A2USHS1CPU	A2USH-S1	**CPU_Q03UDCPU**	Q03UDCPU
CPU_A3UCPU	A3U	**CPU_Q04UDHCPU**	Q04UDHCPU
CPU_A4UCPU	A4U	**CPU_Q06UDHCPU**	Q06UDHCPU
CPU_Q2ACPU	Q2A		
CPU_Q2AS1CPU	Q2A-S1	**CPU_Q17NNCCPU**	Q172NCCPU
CPU_Q3ACPU	Q3A		Q173NCCPU
CPU_Q4ACPU	Q4A, C64	**CPU_Q02CPU_A**	Q02(H) A mode
CPU_FX0CPU	FX0, FX0S	**CPU_Q06CPU_A**	Q06H A mode
CPU_FX0NCPU	FX0N	**CPU_Q12CPU_A**	Q12H A mode
CPU_FX1CPU	FX1	**CPU_Q25CPU_A**	Q25H A mode
CPU_FX2CPU	FX2、FX2C	**CPU_P25**	P25/R25
CPU_FX2NCPU	FX2N, FX2NC	**CPU_LP25**	LP25/BR15
CPU_FX1SCPU	FX1S	**CPU_QLP25**	QLP25/QBR15
CPU_FX1NCPU	FX1N, FX1NC	**CPU_QJ72LP25**	QJ72LP25/BR15
CPU_FX3UCCPU	FX3UC, FX3U	**CPU_A900GOT**	GOT900/1000
CPU_A171SHCPU	A171SH	**CPU_BOARD**	Host board
CPU_A172SHCPU	A172SH	**CPU_PC**	Personal computer (LM)
CPU_A273UHCPU	A273UH	**CPU_OTHER**	General
CPU_A173UHCPU	A173UH		

lUnitType : Set the unit connected with the physical port of the computer

Value	Meaning
UNIT_ACPU	ACPU-RS422 port direct connection
UNIT_QCPU	QnACPU-RS422 port direct connection
UNIT_QNCPU	QnCPU (Q mode) RS232C port direct connection
UNIT_QNCPU_A	QnCPU (A mode) RS232C direct connection
UNIT_QNUSB	QnCPU (Q mode) USB port direct connection
UNIT_QNUSB_A	QnCPU (A mode) USB port direct connection
UNIT_QNMOTION	Q motion-RS232C port direct connection
UNIT_QNMOTIONUSB	Q motion-USB port direct connection
UNIT_FXCPU	FXCPU-RS422 port direct connection
UNIT_C24	C24 unit direct connection for A
UNIT_UC24	UC24 unit direct connection for A
UNIT_QC24	QC24 unit direct connection for QnA
UNIT_QJ71C24	C24 unit direct connection for Q
UNIT_FXENET_ADP	Ethernet adaptor connection for FX
UNIT_FX232BD	FXCPU computer link (RS232C) connection
UNIT_FX485BD	FXCPU computer link (RS485) connection
UNIT_E71	Ethernet LAN connection for A
UNIT_QE71	Ethernet LAN connection for QnA
UNIT_QJ71E71	Ethernet LAN connection for Q
UNIT_G4ACPU	AJ65BT-G4 (-S3) unit direct connection (ACPU access)
UNIT_G4QCPU	AJ65BT-G4 (-S3) unit direct connection (QnA access)
UNIT_G4QNCPU	AJ65BT-G4-S3 unit direct connection (Qn access)
UNIT_MNET2BOARD	MNET2 board connection
UNIT_MNET10BOARD	MNET/10 board connection
UNIT_MNETHBOARD	MNET/H board connection
UNIT_MNETGBOARD	MNET/G board connection
UNIT_CCLINKBOARD	CC-Link board connection
UNIT_MSPANUBOARD	CPU board connection
UNIT_AFBOARD	AF board connection
UNIT_EMEDBOARD	EmEd board connection
UNIT_SIMULATOR	Simulator (GX Simulator) connection
UNIT_QBF	Personal computer CPU connection for Q
UNIT_SSCBOARD	SSC network board connection
UNIT_A900GOT	GOT900 series connection
UNIT_OTHER	General connection

lPacketType : Set the packet message format of computer link or Ethernet. Set either of the following formats in this parameter.

Value	Meaning
PACKET_BINARY1	Dedicated protocol format (for AJ71QC24/UC24 series)
PACKET_ASCII1	Dedicated protocol format (for AJ71(U)C24, AJ71E71/AJ71QE71 series)
PACKET_PLC1	CPU protocol format (for other than AJ71E7/AJ71QE71 series)

lProtocolType : Set the communication protocol type of the connected unit (board). When selecting communication via AJ71QC24N/QJ71C24+modem, select “Via serial port and modem”. (When directly connecting to AJ71QC24N/QJ71C24, select “Via serial port”.) Select “Via shared memory server” only when connecting a simulator.

Value	Meaning
PROTOCOL_MNET2	Via MNET II board
PROTOCOL_MNET10	Via MNET/10 board
PROTOCOL_MNETH	Via MNET/10H or MNET/25H baord
PROTOCOL_MNETG	Via MNET/G board
PROTOCOL_EMED	Via EmEd board
PROTOCOL_SERIAL	Via serial port
PROTOCOL_USB	Via USB port
PROTOCOL_TCPIP	Via TCP/IP
PROTOCOL_UDPIP	Via UDP/IP
PROTOCOL_SHAREDMEMORY	Via shared memory server
PROTOCOL_CCLINK	Via CC-Link board
PROTOCOL_MSPANU	Via CPU board
PROTOCOL_AF	Via AF board
PROTOCOL_SSC	Via SSC net
PROTOCOL_TEL	Via Q6TEL or A6TEL
PROTOCOL_SERIALMODEM	Via serial port and modem
PROTOCOL_QBF	Via Q bus
PROTOCOL_USBGOT	Via GOT1000 USB port

lPortNumber : Set the port number for connection of the physical port of the computer and the unit specified in lUnitType. For the port connectable with the connection unit, refer to the connectable ports in Remarks. For Ethernet connection, set any value as the port number of the request source (personal computer). When setting "=0" as the port number, use the automatic response system as the MNET/10 routing system. When selecting other than the automatic response system via the QE71 series or E71/QE71’s TCP/IP designation, set the fixed value of "5001".

Value	Meaning
PORT_1	Communication port 1
PORT_2	Communication port 2
PORT_3	Communication port 3
PORT_4	Communication port 4
PORT_5	Communication port 5
PORT_6	Communication port 6
PORT_7	Communication port 7
PORT_8	Communication port 8
PORT_9	Communication port 9
PORT_10	Communication port 10

For Ethernet connection, set as follows.

Model	Protocol		Port No.
QJ71E71 AJ71QE71 (UDP)	UDP	Other than automatic response system	Fixed to 5001
		Automatic response system	0: Free port in personal computer is automatically allocated
			Other than 0: Designated port is used to create socket
	TCP	-	Fixed to 0: Free port in personal computer is automatically allocated
AJ71QE71 (TCP) AJ71E71	UDP	Designate according to the port No. designated by sequence	
	TCP	When designated by sequence: Designate according to the designated port No.	
		When not designated by sequence: Designate any free port in personal conputer	

lBaudRate : Set the baudrate for serial communication. Set any of the following values in this parameter.

Value	Meaning
CBR_2400	2400bps
CBR_4800	4800bps
CBR_9600	9600bps
CBR_14400	14400bps
CBR_19200	19200bps
CBR_38400	38400bps
CBR_56000	56000bps
CBR_57600	57600bps
CBR_115200	115200bps
CBR_128000	128000bps
CBR_256000	256000bps

lDataBits : Set the bit count (6 to 8) of the byte data to be sent or received.

lParity : Set the used parity system. Set any of the following values in this parameter. This parameter is valid for serial communication only.

Value	Meaning
EVENPARITY	Even
ODDPARITY	Odd
MARKPARITY	Mark
NOPARITY	No parity

lStopBits : Set the number of stop bits used. Set any of the following values in this parameter. This parameter is valid for serial communication only.

Value	Meaning
ONESTOPBIT	1 stop bit
ONE5STOPBITS	1.5 stop bits
TWOSTOPBITS	2 stop bits

lControl : Set the control setting of the signal line. Set any of the following values in this parameter. This parameter is valid for serial communication only.

Value	Meaning
TRC_NONE	No flow control
TRC_DTR	DTR control
TRC_RTS	RTS control
TRC_DTR_OR_RTS	DTR or RTS control
TRC_DTR_CD	DTR control (with CD control)
TRC_RTS_CD	RTS control (with CD control)
TRC_DTR_OR_RTS_CD	DTR or RTS control (with CD control)

lpcwszHostAddress : Set the host name (IP address) of Ethernet as a UNICODE character string. Set NULL for other than Ethernet communication.

lCpuTimeOut : Set the CPU monitoring timer for Ethernet communication in increments of *250ms (defaults to 4).

lTimeOut : Set the time-out value of communication in increments of 1ms (defaults to 1000ms).
*Time-out is counted from when data send or receive has stopped.

lSumCheck : Set whether sumcheck will be made or not. Set either of the following values in this parameter. This parameter is valid only for communication made via the computer link unit or A series Ethernet unit (TCP/IP).

Value	Meaning
TRUE	With sumcheck
FALSE	Without sumcheck

lSourceNetworkNumber : Set the request source network number when QnA/Qn Ethernet is specified. Set the same network number (network number specified in the network parameter) as that for connected QnA/Qn Ethernet.

lSourceStationNumber : Set the request source station number (personal computer side station number) when QnA/Qn Ethernet is specified. Set the station number so that it does not overlap the station number of the QE71 set within the same Ethernet loop.

lDestinationPortNumber : Set the port number of the destination unit when Ethernet is specified. For access to another network, set the relayed port number. Set the following for other than the automatic response system/E71/QE71 (TCP/IP).

・QnA（UDP/IP） : Fixed to "5001"
・Qn（TCP/IP） : Fixed to "5002"
・Qn（UDP/IP） : Fixed to "5001"

lDestinationIONumber : Set the actual I/O number (first I/O number÷16) of the last accessed station for Qn multidropping (via serial communication/CC-Link). (When the destination is the intelligent special unit) When the destination is the CPU unit, set any of 0x3F0 to 0x3FF. (Refer to "lIONumber".)

lConnectChannelNumber : Set the connected channel number (Ch1/Ch2) when Qn serial communication unit connection is specified. As this parameter is reserved for the system, set no value in this parameter. (Set 0x00.)

lMultiDropChannelNumber : Set the multidropping channel number (Ch1/Ch2) for Qn multidropping. This parameter is invalid when any other connection is specified.

Value	Meaning
0x01	Channel 1 connection
0x02	Channel 2 connection

lThroughNetworkType : Set whether or not the MNET/10 mode is included in the relayed network for access to another station via MELSECNET/10H.

Value	Meaning
0x00	MNET/10 mode not included
0x01	MNET/10 mode included

lIntelligentPreferenceBit : Set whether or not the network at the multidrop link destination will be relayed for Qn multidropping (via serial communication/CC-Link). (This setting is made to differentiate the host network unit.)

Value	Meaning
0x00	No access to another network at multidrop destination
0x01	Access to another network at multidrop destination

lDidPropertyBit : You need not set "lUnitNumber" by making the following setting invalid for access to the Qn series host station's intelligent special unit (intelligent special unit loaded to the host station CPU). (Use only the unit I/O number "lIONumber" to specify.)

Value	Meaning
0x00	Unit number is made valid
0x01	Unit number is made invalid

lDsidPropertyBit : You need not set "lDestinationIONumber" by making the following setting invalid for Qn multidropping. Note that "lDidPropertyBit" must be made valid when the following setting is made invalid. (Use "lUnitNumber" to specify.)

Value	Meaning
0x00	I/O number of last accessed station is made valid
0x01	I/O number of last accessed station is made invalid

plRet : Returns an error code

S_OK : Normal termination
EZNC_COMM_ALREADYOPENED: Setting impossible as the line is already open
EZ_ERR_DATA_TYPE : Data type of the argument is illegal
EZ_ERR_DATA_RANGE : Data range of the argument is illegal
EZ_ERR_NOT_SUPPORT : Not supported

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This is for setting MELSEC communication.
Call this before **Open2().** If not, an error occurs at executing **Open2()**.
The setting set here will be held until the object is released by **Release()**.
Once **Open2()** is executed, resetting by **SetMelsecProcotol()** is impossible until **Close()** is executed. An eror occurs.
If an error occurs at **SetMelsecProcotol()**, setting before the error is held. Setting that caused the error is not held.
For argument of pointer designation as “lpcwszHostAddress”, set NULL if you don’t use this argument.

(Note) This method doesn’t support automation interface. It is limited to custom interface

☐Reference

Open2(), Close()

☐Designation

2.3.7 IEZNcCommunication3::SetModalCondition — Setting modal communication condition

☐Calling procedure (Custom interface)

```
HRESULT        SetModalCondition (
                   LONG lRegistTime,        // (I)Setting modal communication register time
                   LONG lCancelTime,        // (I)Setting modal communication cancel time
                   LONG lIntervalTime,      // (I)Setting modal communication interval time
                   LONG* plRet              // (O) Error code
                   )
```

☐Calling procedure (Automation interface)

```
               SetModalCondition (
                   lRegistTime As LONG      // (I)Setting modal communication register time
                   lCancelTime As LONG      // (I) Setting modal communication cancel time
                   lIntervalTime As LONG    // (I)Setting modal communication interval time
                   )As LONG                 // (O) Error code
```

☐Argument

lRegistTime: Set modal communication register time [ms] (0 to 600000 [ms])
When 0ms is set, register for the modal communication won't be performed.
Default value: 2000 [ms]

lCanceltTime: Set modal communication cancel time [ms] (0 to 600000 [ms])
When 0ms is set, the modal communication will be canceled immediately at the next communication cycle.
Default value: 4000 [ms]

lIntervalTime: Set modal communication interval time [ms] (10 to 600000 [ms])
Default value: 20 [ms]

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_SYSFUNC_IOCTL_ADDR: NC card # illegal
EZNC_SYSFUNC_IOCTL_NOTOPEN: Device not open

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This sets modal communication condition.
Modal communication is a communication method to collect data from the NC regularly. After collecting data regularly in the setting range of the modal communication setting, the normal transient communication is switched to modal communication. After switched to modal communication, the target data is taken into the PC regularly to be read. However, if you perform reading with a faster cycle than the cycle of taking data from the NC, you cannot read the newest data.

☐Reference

Close(), GetModalCondition()

☐Designation

2.3.8 IEZNcCommunication3::GetModalCondition — Getting modal communication condition

☐Calling procedure (Custom interface)

```
HRESULT        GetModalCondition (
                   LONG* plRegistTime,        // (O) Getting the modal communication register time
                   LONG* plCancelTime,        // (O) Getting the modal communication cancel time
                   LONG* plIntervalTime,      // (O) Getting the modal communication interval time
                   LONG* plRet                // (O) Error code
                   )
```

☐Calling procedure (Automation interface)

```
               GetModalCondition (
                   plRegistTime As LONG*,     // (O) Getting the modal communication register time
                   plCancelTime As LONG*,     // (O) Getting the modal communication cancel time
                   plIntervalTime As LONG*    // (O) Getting the modal communication interval time
                   )As LONG                   // (O) Error code
```

☐Argument

plRegistTime: Returns the modal communication register time [ms] (0 to 600000 [ms])
Default value: 2000 [ms]

plCanceltTime: Returns the modal communication cancel time [ms] (0 to 600000 [ms])
Default value: 4000 [ms]

plIntervalTime: Returns the modal communication interval time [ms] (10 to 600000 [ms])
Default value: 20 [ms]

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_SYSFUNC_IOCTL_ADDR: NC card # illegal
EZNC_SYSFUNC_IOCTL_NOTOPEN: Device not open

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This gets the modal communication condition that has been set.

☐Reference

Close(), SetModalCondition()

☐Designation

2.4 IEZNcSystem Interface

Magic64 Limited	M6x5M	M6x5L	C64	C70	CNC700

2.4.1 IEZNcSystem::GetVersion — Getting NC System # and name

☐Calling procedure (Custom interface)

```
HRESULT        GetVersion(
                    LONG lAxisNo,              //  (I)  Axis designation
                    LONG lIndex,               //  (I)  Parameter #
                    LPOLESTR* lppwszBuffer,    //  (O)  NC system # and system name
                    LONG* plRet                //  (O)  Error code
                    )
```

☐Calling procedure (Automation interface)

```
               System_GetVersion(
                    lAxisNo As LONG            //  (I)  Axis designation
                    lIndex As LONG             //  (I)  Parameter #
                    lppwszBuffer As STRING*    //  (O)  NC system # and system name
                    )As LONG                   //  (O)  Error code
```

☐Argument

lAxisNo: Set axis # (“1” or later) (Not used)

lIndex: Set parameter # (Refer to the table below.)

lppwszBuffer: Returns the system #, system name and control S/W version as UNICODE character strings.

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK : Normal termination
EZ_ERR_DATA_TYPE: Argument data type is illegal
EZNC_DATA_READ_DATASIZE: Too much data for the buffer prepared by the application
EZNC_DATA_READ_READ: Impossible to read the data

lIndex	Meaning	Data range	Magic64
0	System #, name, control PLC version	Depends on the system specifications	Available
1	Control unit, extended unit	Depends on the system specifications	Unavailable
2	RIO unit, terminal RIO unit Axis designation is necessary. (Excluding CNC700)	Depends on the system specifications	Unavailable

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐ **Functions**

This gets various information on each version of NC system as **UNICODE**.

0: This gets NC system #, system name, and control PLC version.
The data format of the character string is as below.
NC system number\tNC system name\tPLC system number\0
Do not fail to add TAB code between the NC system # and NC system name.
The data have to be ended with NULL code.
Output example: "BND-353W000-A0\tMELDASMAGIC64\tBND-400W000-A0"
TAB code comes next if there is no item. TAB code is followed by NULL code if there is no item for termination.
In the case of M6x5M, System model\tS/W vertion\0

1: This gets versions of the NC system's control unit and extended units.
The data format of the character string is as below.
Control unit number\tExtended unit number\0
Do not fail to add **TAB** code between the control unit # and extended unit #.
The data have to be ended with **NULL** code.
In the case of M6x5M, Control unit\tExtended unit1\tExteded unit2…\0

2: This gets versions of the NC system's RIO and terminal RIO unit.
The data format of the character string is as below.
RIO unit number\tTerminal RIO unit number\0
In the case of M6x5M or CNC700, 24 units. (RIO unit 1\tRIO unit 2\t…\0)
Do not fail to add **TAB** code between the RIO unit # and terminal RIO unit #.
The data have to be ended with **NULL** code.
For RIO unit and terminal RIO unit, axis designation is necessary.

Memory area for character strings is saved inside of EZSocket. If your client is a VC++ client, it is necessary to release the memory area explicitly by using **CoTaskMemFree()**.

With MELDASMAGIC64, only 0 is available for lIndex.
If 1 or 2 is designated, EZ_ERR_DATA_TYPE will be returned in plRet.

(Note) Mitsubishi CNC C70 Series doesn't support the automation interface.

☐ **Reference**

☐ **Designation**

2.4.2 IEZNcSystem::GetSystemInformation — Getting NC system information

☐Calling procedure (Custom interface)

```
HRESULT        GetSystemInformation(
                    LONG lType,          //  (I)  Information type
                    LONG* plSystem,      //  (O)  System information
                    LONG* plRet          //  (O)  Error code
                    )
```

☐Calling procedure (Automation interface)

```
               System_GetSystemInformation(
                    lType As LONG        //  (I)  Information type
                    plSystem As LONG*    //  (O)  System information
                    )As LONG             //  (O)  Error code
```

☐Argument

lType: Set NC system information type (Refer to the table below.)

plSystem: Returns the NC system information

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK : Normal termination
EZ_ERR_DATA_TYPE: Data type of the argument is illegal
EZNC_DATA_READ_READ: Impossible to read the data

lType	Meaning	Data range
0	Validity of the part system.	0: Invalid part system 1: Valid part system
1	The number of the axes in the part system	More than 1 (Depends on each NC system)

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This gets the information on NC part system.

(Note) Mitsubishi CNC C70 Series doesn't support the automation interface.

☐Reference

☐Designation

System

2.4.3 IEZNcSystem::GetAlarm — Getting alarm information

☐Calling procedure (Custom interface)

```
HRESULT        GetAlarm(
                    LONG lMessageNumber,        //  (I)   The number of alarm messages to get
                    LONG lAlarmType,            //  (I)   Alarm types to get
                    LPOLESTR* lppwszBuffer,     //  (O)   Message character strings
                    LONG* plRet                 //  (O)   Error code
                    )
```

☐Calling procedure (Automation interface)

```
               System_GetAlarm(
                    lMessageNumber As LONG      //  (I)   The number of alarm messages to get
                    lAlarmType As LONG          //  (I)   Alarm types to get
                    lppwszBuffer As STRING*     //  (O)   Message character strings
                    )As LONG                    //  (O)   Error code
```

☐Argument

lMessageNumber: Set the number of messages to get. Number**: 1 to 10** (Max.)

lAlarmType: Set alarm types to get

Error code	Meaning
M_ALM_NC_ALARM	NC alarm
M_ALM_STOP_CODE	Stop code (Invalid with M6x5M)
M_ALM_PLC_ALARM	PLC alarm messages
M_ALM_OPE_MSG	Operator messages
M_ALM_ALL_ALARM	All kinds of alarm (Invalid with M6x5M)
M_ALM_WARNING	Warnings (Valid only with M6x5M)

lppwszBuffer: Gets alarm messages as **UNICODE** character strings. Messages are marked off by **CR** or **LF** code. Messages are ended with **NULL**.

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK: Normal termination
EZNC_OPE_CURRALM_ADDR: Part system designation, or axis designation is illegal
EZNC_OPE_CURRALM_ALMTYPE: Alarm type is illegal
EZNC_OPE_CURRALM_DATAERR: Communication data errors between NC and PC
EZNC_OPE_CURRALM_DATASIZE: Too much data for the buffer prepared by the application
EZNC_OPE_CURRALM_NOS: The number of messages is illegal

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This gets the alarm messages currently occurring in the designated NC card. The language of alarm messages depends on the NC parameter (#1043 lang).
Messages are taken in order of importance.
Memory area for character strings is saved inside of EZSocket. If your client is a VC++ client, it is necessary to release the memory area explicitly by using **CoTaskMemFree().**

☐Reference

☐Designation

System When set to 0, all the part systems are designated.

2.5 IEZNcPosition Interface

Magic64	M6x5M	M6x5L	C64	C70	CNC700

2.5.1 IEZNcPosition::GetWorkPosition — Getting the workpiece coordinate position

☐Calling procedure (Custom interface)

```
HRESULT        GetWorkPosition(
                   LONG lAxisNo,          //  (I)  Axis designation
                   DOUBLE* pdPosition,    //  (O)  Workpiece coordinate position
                   LONG lSkipOn,          //  (I)  Skip ON flag
                   LONG* plRet            //  (O)  Error code
                   )
```

☐Calling procedure (Automation interface)

```
               Position_GetWorkPosition(
                   lAxisNo As LONG        //  (I)  Axis designation
                   pdPosition As DOUBLE*  //  (O)  Workpiece coordinate position
                   lSkipOn As LONG        //  (I)  Skip ON flag
                   )As LONG               //  (O)  Error code
```

☐Argument

lAxisNo: Set axis # (“1” or later)

pdPosition: Returns the workpiece coordinate position of the designated axes in the designated part system

Data range : -99999.9999 to 99999.9999 [mm] (other than **CNC700**) 8 digits from the leftmost are displayed for integer and decimal parts in total.

: -999999.999999 to 999999.999999 [mm] (**CNC700**) 10 digits from the leftmost are displayed for integer and decimal parts in total.

Each number of digits for each part depends on the NC specifications (or options) and the values (for parameters) set by the machine tool builder.

lSkipOn: Set skip on flag

Value	Meaning
1	At skip ON (Invalid with **M6x5L**)
0	In normal operation

plRet: Returns an error code. (When using automation interface, returns a return value instead.)

S_OK : Normal termination
EZ_ERR_DATA_TYPE: Data type of the argument is illegal
EZNC_DATA_READ_ADDR: Part system designation or axis designation is illegal
EZNC_DATA_READ_READ: Impossible to read the data

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This gets the workpiece coordinate position of the designated axis part system and axis.
When skip ON flag is 1, it gets the workpiece coordinate position at the moment that skip ON signal turns on.

(Note) Mitsubishi CNC C70 Series doesn’t support the automation interface.

☐Reference

☐Designation

System, Axis

2.5.2 IEZNcPosition::GetMachinePosition — Getting the machine position

☐Calling procedure (Custom interface)

```
HRESULT        GetMachinePosition(
                   LONG lAxisNo,            //  (I)  Axis designation
                   DOUBLE* pdPosition,      //  (O)  Machine position
                   LONG lSkipOn,            //  (I)  Skip ON flag
                   LONG* plRet              //  (O)  Error code
                   )
```

☐Calling procedure (Automation interface)

```
               Position_GetMachinePosition(
                   lAxisNo As LONG          //  (I)  Axis designation
                   pdPosition As DOUBLE*    //  (O)  Machine position
                   lSkipOn As LONG          //  (I)  Skip ON flag
                   )As LONG                 //  (O)  Error code
```

☐Argument

lAxisNo: Set axis # (“1” or later)

pdPosition: Returns the machine position of the designated axes in the designated part system
Data range: -99999.9999 to 99999.9999 [mm] (other than **CNC700**) 8 digits from the leftmost are displayed for integer and decimal parts in total.
: -999999.999999 to 999999.999999 [mm] (**CNC700**)10 digits from the leftmost are displayed for integer and decimal parts in total.
Each number of digits for each part depends on the NC specifications (or options) and the values (for parameters) set by the machine tool builder.

lSkipOn: Set the skip ON flag

Value	Meaning
1	At skip ON (Invalid with **M6x5M/L**)
0	In normal operation

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK : Normal termination
EZNC_DATA_READ_ADDR: Part system designation or axis designation is illegal
EZNC_DATA_READ_READ: Impossible to read the data

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This gets the machine coordinate position (coordinate position in the base machine coordinate system) of the designated part system and axis.
When skip ON flag is 1, it gets the machine coordinate position at the moment that skip ON signal turns on.

(Note) Mitsubishi CNC C70 Series doesn’t support the automation interface.

☐Reference

☐Designation

System, Axis

2.5.3 IEZNcPosition::GetCurrentPosition — Getting the current position

☐Calling procedure (Custom interface)

```
HRESULT        GetCurrentPosition(
                    LONG lAxisNo,          //  (I)  Axis designation
                    DOUBLE* pdPosition,    //  (O)  Current position
                    LONG* plRet            //  (O)  Error code
                    )
```

☐Calling procedure (Automation interface)

```
               Position_GetCurrentPosition(
                    lAxisNo As LONG        //  (I)  Axis designation
                    pdPosition As DOUBLE*  //  (O)  Current position
                    )As LONG               //  (O)  Error code
```

☐Argument

lAxisNo: Set axis # (“1” or later)

pdPosition: Returns the designated axis of the designated part system's relative position at the dog-type origin return complete or the relative position from preset point in G92/Origin set/Counter set

Data range: -99999.9999 to 99999.9999 [mm] (other than **CNC700**) 8 digits from the leftmost are displayed for integer and decimal parts in total.

: -999999.999999 to 999999.999999 [mm] (**CNC700**)10 digits from the leftmost are displayed for integer and decimal parts in total.

Each number of digits for each part depends on the NC specifications (or options) and the values (for parameters) set by the machine tool builder.

plRet: Returns an error code. (When using automation interface, returns a return value instead.)

S_OK: Normal termination
EZNC_DATA_READ_ADDR: Part system designation or axis designation is illegal
EZNC_DATA_READ_READ: Impossible to read the data

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This gets the current coordinate position (relative position at the dog-type origin return complete or the relative position from preset point in G92/Origin set/Counter set) for the designated part system and axis.

(Note) Mitsubishi CNC C70 Series doesn’t support the automation interface.

☐Reference

☐Designation

System, Axis

2.5.4 IEZNcPosition::GetDistance — Getting the remaining distance

☐Calling procedure (Custom interface)

```
HRESULT        GetDistance(
                    LONG lAxisNo,              //  (I)  Axis designation
                    DOUBLE* pdDistance,        //  (O)  Remaining command distance
                    LONG lSkipOn,              //  (I)  Skip ON flag
                    LONG* plRet                //  (O)  Error code
                    )
```

☐Calling procedure (Automation interface)

```
               Position_GetDistance(
                    lAxisNo As LONG            //  (I)  Axis designation
                    pdDistance As DOUBLE*      //  (O)  Remaining command distance
                    lSkipOn As LONG            //  (I)  Skip ON flag
                    )As LONG                   //  (O)  Error code
```

☐Argument

lAxisNo: Set axis # (“1” or later)

pdDistance: Returns the remaining distance of the current travel command of the designated axis in the designated part system.
Data range: -99999.9999 to 99999.9999 [mm] (other than **CNC700**) 8 digits from the leftmost are displayed for integer and decimal parts in total.
: -999999.999999 to 999999.999999 [mm] (**CNC700**) 10 digits from the leftmost are displayed for integer and decimal parts in total.
Each number of digits for each part depends on the NC specifications (or options) and the values (for parameters) set by the machine tool builder.

lSkipOn: Set the skip ON flag

Value	Meaning
1	At skip ON (Invalid with **M6x5M/L**)
0	In normal operation

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK: Normal termination
EZ_ERR_DATA_TYPE: Data type of the argument is illegal
EZNC_DATA_READ_ADDR: Part system designation or axis designation is illegal
EZNC_DATA_READ_READ: Impossible to read the data

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This gets the remaining distance of the current travel command of the designated part system and axis.
When skip ON flag is 1, it gets the machine coordinate position at the moment that skip ON signal turns on.

(Note) Mitsubishi CNC C70 Series doesn’t support the automation interface.

☐Reference

☐Designation

System, Axis

2.5.5 IEZNcPosition::GetNextDistance — Getting the next travel distance

☐Calling procedure (Custom interface)

```
HRESULT        GetNextDistance(
                    LONG lAxisNo,            //  (I)  Axis designation
                    DOUBLE* pdDistance,      //  (O)  Next travel distance
                    LONG* plRet              //  (O)  Error code
                    )
```

☐Calling procedure (Automation interface)

```
               Position_GetNextDistance(
                    lAxisNo As LONG          //  (I)  Axis designation
                    pdDistance As DOUBLE*    //  (O)  Next travel distance
                    )As LONG                 //  (O)  Error code
```

☐Argument

lAxisNo: Set axis # (“1” or later)

pdDistance: Returns the travel distance in the next block of the designated axis in the designated part system

Data range: -99999.9999 to 99999.9999 [mm] (other than **CNC700**) 8 digits from the leftmost are displayed for integer and decimal parts in total.

: -999999.999999 to 999999.999999 [mm] (**CNC700**)10 digits from the leftmost are displayed for integer and decimal parts in total.

Each number of digits for each part depends on the NC specifications (or options) and the values (for parameters) set by the machine tool builder.

plRet: Returns an error code. (When using automation interface, returns a return value instead.)

S_OK: Normal termination
EZNC_DATA_READ_ADDR: Part system designation or axis designation is illegal
EZNC_DATA_READ_READ: Impossible to read the data

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This gets the travel distance in the next block of the designated part system and axis.

(Note) Mitsubishi CNC C70 Series doesn’t support the automation interface.

☐Reference

☐Designation

System, Axis

2.5.6 IEZNcPosition::GetFeedSpeed — Getting the feedrate

☐Calling procedure (Custom interface)

```
HRESULT        GetFeedSpeed(
                    LONG lFeedType,          //  (I)  Feedrate type
                    DOUBLE* pdSpeed,         //  (O)  Feedrate
                    LONG* plRet              //  (O)  Error code
                    )
```

☐Calling procedure (Automation interface)

```
               Position_Get FeedSpeed(
                    lFeedType As LONG        //  (I)  Feedrate type
                    pdSpeed As DOUBLE*       //  (O)  Feedrate
                    )As LONG                 //  (O)  Error code
```

☐Argument

lFeedType: Set feedrate type to get
In the case of **M6x5M**, the value of synchronous federate and screw lead are the same as that of effective federate in automatic operation.

Value	Meaning
0	F programming feedrate (FA)
1	Effective feedrate in manual feed (FM)
2	Synchronous feedrate (FS)
3	Effective feedrate in automatic operation (Fc)
4	Screw lead feedrate (FE)

pdSpeed: Returns the feedrate of the designated part system

Feedrate type	Data range	
	Other than CNC C70	**CNC C70**
FA	0.000 to 240000.000 [mm/min]	0.000 to 1000000.000 [mm/min]
FM	0.000 to 240000.000 [mm/min]	0.000 to 1000000.000 [mm/min]
FS	0.000 to 240000.000 [mm/min]	0.000 to 999.999 [mm/rev]
Fc	0.000 to 240000.000 [mm/min]	0.000 to 1000000.000 [mm/min]
FE	0.000 to 240000.000 [mm] (Different unit)	0.000 to 999.999 [mm/rev]

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK : Normal termination
EZ_ERR_DATA_TYPE: Data type of the argument is illegal
EZNC_DATA_READ_ADDR: Part system designation or axis designation is illegal
EZNC_DATA_READ_READ: Impossible to read the data

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This gets the feedrate of the designated part system.

(Note) Mitsubishi CNC C70 Series doesn't support the automation interface.

☐Reference

☐Designation

System

2.5.7 IEZNcPosition::GetManualOverlap — Getting the manual interruption amount

☐Calling procedure (Custom interface)

```
HRESULT        GetManualOverlap(
                    LONG lAxisNo,          //  (I)  Axis designation
                    LONG lType,            //  (I)  Type
                    DOUBLE* pdLength,      //  (O)  Manual interruption amount
                    LONG* plRet            //  (O)  Error code
                    )
```

☐Calling procedure (Automation interface)

```
               Position_GetManualOverlap (
                    lAxisNo As LONG        //  (I)  Axis designation
                    lType As LONG          //  (I)  Type
                    pdLength As DOUBLE*    //  (O)  Manual interruption amount.
                    )As LONG               //  (O)  Error code
```

☐Argument

lAxisNo: Set axis # (“1” or later)

lType: Set the type of the manual interruption amount

Value	Meaning
0	To get the manual interruption amount while manual ABS switch is OFF
1	To get the manual interruption amount while manual ABS switch is ON (Invalid with **M6x5M/L**)

pdLength: Returns the manual interruption amount of the designated axis of the designated part system.
With **M6x5M**, accumulation of the manual interruption amount
With **M6x5L**, the interruption amount of the current status of the manual ABS switch
Data range: -99999.999 to 99999.999 [mm]

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK: Normal termination
EZ_ERR_DATA_TYPE: Data type of the argument is illegal
EZNC_DATA_READ_ADDR: Part system designation or axis designation is illegal
EZNC_DATA_READ_READ: Impossible to read the data

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure.

☐Functions

This returns the manual interruption amount of the designated axis of the designated part system and axis.

(Note) Mitsubishi CNC C70 Series doesn’t support the automation interface.

☐Reference

☐Designation

System, Axis

2.5.8 IEZNcPosition::GetProgramPosition — Getting the program position

☐Calling procedure (Custom interface)

```
HRESULT        GetProgramPosition(
                    LONG lAxisNo,              //  (I)  Axis designation
                    DOUBLE* pdPosition,        //  (O)  Program position
                    LONG* plRet                //  (O)  Error code
                    )
```

☐Calling procedure (Automation interface)

```
               Position_GetProgramPosition(
                    lAxisNo As LONG            //  (I)  Axis designation
                    pdPosition As DOUBLE*      //  (O)  Program position
                    )As LONG                   //  (O)  Error code
```

☐Argument

lAxisNo: Set axis # (“1” or later)

pdPosition: Returns the program position
Data range: -99999.999 to 99999.999 [mm]

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK : Normal termination
EZNC_DATA_READ_ADDR: Part system designation or axis designation is illegal
EZNC_DATA_READ_READ: Impossible to read the data

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This gets the program position. Invalid with **MELDASMAGIC64**.

(Note 1) CNC C70 doesn’t support PLC axis sytem’s program position.
(Note 2) Mitsubishi CNC C70 Series doesn’t support the automation interface.

☐Reference

☐Designation

System, Axis

2.6 IEZNcCommand2 Interface

Magic64	M6x5M	M6x5L	C64	C70	CNC700

2.6.1 IEZNcCommand2::GetGCodeCommand — Getting the G-code modal command value

☐Calling procedure (Custom interface)

```
HRESULT        GetGCodeCommand(
                    LONG lType,              //  (I)  Type
                    DOUBLE* pdValue,         //  (O)  Command value
                    LONG* plRet              //  (O)  Error code
                    )
```

☐Calling procedure (Automation interface)

```
               Command_GetGCodeCommand(
                    lType As LONG            //  (I)  Type
                    pdValue As DOUBLE*       //  (O)  Command value
                    ) As LONG                //  (O)  Error code
```

☐Argument

lType: Set the type of G-code modal command value to get

Value	Meaning
1	Group 1 (Interpolation mode). G00, G01 G02, G03, and G33 command modal.
2	Group 2 (Plane selection). G17, G18, and G19 command modal.
3	Group3 (Absolute) G90 and (Incremental) G91 command modal.
4	Group 4 (Chuck barrier) G22 and G23 command modal.
5	Group 5 (Feed mode) G94 and G95 command modal.
6	Group 6 (Inch) G20 and (mm) G21 command modal.
7	Group 7 (Tool nose R compensation mode) G40, G41, and G42 command modal.
8	Group 8 (Tool length compensation mode) G43, G44 and G49 command modal.
9	Group 9 (Fixed cycle mode) G70, G71, G72, G73, G74, G75, G76, G77, G78, G79, G80, G81, G82, G83, G84, G85, G86, G87, G88, and G89 command modal.
10	Group 10 (Initial return) G98, (Reference point return) G99 command modal.
11	Group 11 G50.2 and G51.2 command modal. (Invalid with **M6x5L** and **C64**)
12	Group 12 (Workpiece coordinate system modal) G54, G55, G56, G57, G58, and G59 command modal.
13	Group 13 (Cutting mode) G61, G62, G63, and G64 command modal.
14	Group 14 (Modal call) G66, G66.1, and G67 command modal.
15	Group 15 (Normal line control) G40.1, G41.1 and G42.1 command modal. (Valid only with **M6x5M** or **CNC700M**) (Mirror image for facing tool posts) G68 and G69 modal (For **C64L**)
16	Group 16 (Coordinate system rotation) G68 and G69 (Valid only with **CNC700M**)
17	Group 17 (Constant peripheral speed control) G96 and G97 command modal.
18	Group 18 (Balance cut/polar coordinate command) G14, G15 and G16 command modal.
19	Group 19 (G command mirror image) G50.1 and G51.1 command modal.
20	Group 20 (Spindle selection) G43.1, G44.1 and G47.1 command modal. (Invalid with **M6x5M/L**)
21	Group 21(Cylindrical interpolation/polar coordinate interpolation) G07.1, G107, G12.1, G112, G13.1, G113 (Valid only with **CNC700M**)

pdValue: Returns the current G-code modal command value of the designated part system

Value (Example)	Meaning
2	G02
17	G17
50.2	G50.2

☐**Argument** *plRet*: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK: Normal termination
EZ_ERR_DATA_TYPE: Data type of the argument is illegal
EZNC_DATA_READ_ADDR: Part system designation is illegal
EZNC_DATA_READ_READ: Impossible to read the data

☐**Return value**

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐**Functions** This gets the G-code modal value of the designated part system.
For the lists of each model's G command and group, refer to each model's Programming manual. By using G code macro call, operation different from the one specified with the original G command may be incorporated in the machine. Refer to the manual issued by the machine tool builder.

(Note) Mitsubishi CNC C70 Series doesn't support the automation interface.

☐**Reference**

☐**Designation** System

2.6.2 IEZNcCommand2::GetToolCommand — Getting the tool offset

☐Calling procedure (Custom interface)

```
HRESULT        GetToolCommand(
                    LONG lAxisNo,          //  (I)  Axis designation
                    LONG lType,            //  (I)  Type
                    LONG* plValue,         //  (O)  Tool offset #
                    LONG* plRet            //  (O)  Error code
                    )
```

☐Calling procedure (Automation interface)

```
               Command_GetToolCommand(
                    lAxisNo As LONG        //  (I)  Axis designation
                    lType As LONG          //  (I)  Type
                    plValue As LONG*       //  (O)  Tool offset #
                    ) As LONG              //  (O)  Error code
```

☐Argument

lAxisNo: Set axis in getting the tool length offset # (Axis #1 or later)

lType: Set type of tool offset to get

Value	Meaning
0	D value of the tool shape offset #
1	D value of the tool wear offset #
2	H value of the tool length offset # (Axis designation required.)

plValue: Returns the tool shape and offset # of the designated part system, and the tool length offset # of the designated axis of the designated part system
Data range: 1 to 200 (The range depends on the number of the tool offset sets.)
Meaning of value: 1=D1, 1=H1

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK: Normal termination
EZ_ERR_DATA_TYPE: Data type of the argument is illegal
EZNC_DATA_READ_ADDR: Part system designation and axis designation is illegal
EZNC_DATA_READ_READ: Impossible to read the data

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This gets the tool shape and offset # of the designated part system, and the tool length offset # of the designated axis of the designated part system.

(Note) Mitsubishi CNC C70 Series doesn't support the automation interface.

☐Reference

☐Designation

System, Axis

2.6.3 IEZNcCommand2::GetFeedCommand — Getting the feedrate command value

☐Calling procedure (Custom interface)

```
HRESULT        GetFeedCommand(
                    LONG lType,              //  (I)  Type
                    DOUBLE* pdValue,         //  (O)  Command value
                    LONG* plRet              //  (O)  Error code
                    )
```

☐Calling procedure (Automation interface)

```
               Command_GetFeedCommand(
                    lType As LONG            //  (I)  Type
                    pdValue As DOUBLE*       //  (O)  Command value
                    ) As LONG                //  (O)  Error code
```

☐Argument

lType: Set command value types to get

Value	Meaning M6x5M	M6x5L
0	F programming feedrate (FA)	F programming feedrate (FA)
1	Synchronous feedrate (FS)	Synchronous feedrate (FS)
2	Screw lead feedrate (FE)	Screw lead feedrate (FE)
3	Peripheral speed	-
4	-	-

Value	Meaning **MELDASMAGIC64, MELDASC6/C64, CNC C70, CNC 700M/L**
0	F programming feedrate (FA)
1	Effective feedrate in manual feed (FM)
2	Synchronous feedrate(FS)
3	Effective feedrate in automatic operation(FC)
4	Screw lead feedrate(FE)

pdValue: Returns the current feedrate of the designated part system

Feedrate type	Data range	
	Other than CNC C70	**CNC C70**
FA	0.000 to 240000.000 [mm/min]	0.000 to 1000000.000 [mm/min]
FM	0.000 to 240000.000 [mm/min]	0.000 to 1000000.000 [mm/min]
FS	0.000 to 240000.000 [mm/min]	0.000 to 999.999 [mm/rev]
Fc	0.000 to 240000.000 [mm/min]	0.000 to 1000000.000 [mm/min]
FE	0.000 to 240000.000 [mm] (Different unit)	0.000 to 999.999 [mm/rev]

plRet: Returns an error code. (When using automation interface, returns a return value instead.)

S_OK: Normal termination
EZ_ERR_DATA_TYPE: Data type of the argument is illegal
EZNC_DATA_READ_ADDR: Part system designation is illegal
EZNC_DATA_READ_READ: Impossible to read the data

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This gets the current feedrate of the designated part system.

(Note) Mitsubishi CNC C70 Series doesn't support the automation interface.

☐Reference

☐Designation System

2.6.4 IEZNcCommand2::GetCommand2 — Getting the M/S/T/B command modal value

☐Calling procedure (Custom interface)

```
HRESULT        GetCommand2(
                    LONG lType,          //  (I)  Command type
                    LONG lIndex          //  (I)  Command No.
                    LONG* plValue,       //  (O)  Command value
                    LONG* plRet          //  (O)  Error code
                    )
```

☐Calling procedure (Automation interface)

```
               Command_GetCommand2(
                    lType As LONG        //  (I)  Command type
                    lIndex As LONG       //  (I)  Command No.
                    plValue As LONG*     //  (O)  Command value
                    ) As LONG            //  (O)  Error code
```

☐Argument

lType: Set command value types to get

Value	Meaning
EZNC_M	M programming 1 (Miscellaneous M value)
EZNC_S	S programming (Spindle speed function S value)
EZNC_T	T programming (Tool exchange T value)
EZNC_B	B programming (Second auxiliary function for indexing table position, etc.)

lIndex: Set command No.
Example) When lType=EZNC_M, lIndex=1, M command will be 1.

Command \ Model	Magic64	M6x5M	M6x5L	C64	CNC C70	CNC700M/L
M	1	1 to 4	1	1 to 4	1 to 4	1 to 4
S	1	1 to 4	1	1 to 7	1 to 7	1 to 4
T	1	1 to 2	1	1 to 4	1 to 4	1
B	1	1 to 4	1	1 to 4	1 to 4	1 to 4

plValue: Returns the current command value of the designated part system.
Data range: 0 to 99999999 (Maximum)

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK : Normal termination
EZ_ERR_DATA_TYPE: Data type of the argument is illegal
EZNC_DATA_READ_READ: Impossible to read the data
EZNC_DATA_READ_ADDR: Part system designation is illegal

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This gets the current M/S/T/B command modal value of the designated part system.

(Note) Mitsubishi CNC C70 Series doesn't support the automation interface.

☐Reference

SetCommand2()

☐Designation

System (PLC axis part system cannot be designated.)

2.6.5 IEZNcCommand2::SetCommand2 — Setting manual command (M/S/T/B)

☐Calling procedure (Custom interface)

```
HRESULT        SetCommand2(
                   LONG lType,          //  (I)  Type
                   LONG lIndex          //  (I)  Command No.
                   LONG lValue,         //  (I)  Command value
                   LONG* plRet          //  (O)  Error code
                   )
```

☐Calling procedure (Automation interface)

```
               Command_SetCommand2(
                   lType As LONG        //  (I)  Type
                   lIndex As LONG       //  (I)  Command No.
                   lValue As LONG       //  (I)  Command value
                   ) As LONG            //  (O)  Error code
```

☐Argument

lType: Set command value types to get

Value	Meaning
EZNC_M	M programming (Miscellaneous M value)
EZNC_S	S programming (Spindle speed function S value)
EZNC_T	T programming (Tool exchange T value)
EZNC_B	B programming (Second auxiliary function value for indexing table position, etc.)

lIndex: Specified No. is specified.
EX) M command 1 when lType=EZNC_M, lIndex=1

Model / *Command*	Magic64	M6x5M	M6x5L	C64	CNC C70	CNC700M/L
M	1	1 to 4	1	1 to 4	1	1 to 4
S	1	1 to 4	1	1 to 7	1 to 7	1 to 4
T	1	1 to 2	1	1 to 4	1	1
B	1	1 to 4	1	1 to 4	1	1 to 4

lValue: Set command value of the designated axis or the designated part system
Data range: 0 to 99999999 (Maximum value)
0 to 9999 (Maximum value) (CNC C70 M command)

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK: Normal termination
EZ_ERR_DATA_TYPE: Data type of the argument is illegal
EZNC_DATA_WRITE_WRITE: Impossible to write the data
EZNC_DATA_WRITE_ADDR: Part system designation is illegal

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This sets the manual command value of M/S/T/B function of the designated axis or designated part system.
The target command of M6x5M/L is only the first command.

(Note) Mitsubishi CNC C70 Series doesn't support the automation interface.

☐Reference

GetCommand2()

☐Designation

System (PLC axis part system cannot be designated.)

2.7 IEZNcProgram2 Interface

Magic64	M6x5M	M6x5L	C64	C70	CNC700

2.7.1 IEZNcProgram2::CurrentBlockRead — Reading the current block

☐Calling procedure (Custom interface)

```
HRESULT        CurrentBlockRead(
                    LONG lBlockNumber,                  //  (I)  The number of blocks
                    LPOLESTR* lppwszProgramData,        //  (O)  Storing program
                    LONG* plCurrentBlockNo,             //  (O)  Block # being executed
                    LONG* plRet                         //  (O)  Error code
                    )
```

☐Calling procedure (Automation interface)

```
               Program_CurrentBlockRead(
                    lBlockNumber As LONG                //  (I)  The number of blocks
                    lppwszProgramData As STRING*        //  (O)  Storing program
                    plCurrentBlockNo As LONG*           //  (O)  Block # being executed
                    ) As LONG                           //  (O)  Error code
```

☐Argument

lBlockNumber: Set the number of the blocks to get. Value: 1 to 10

lppwszProgramData: Gets the program blocks as **UNICODE** character strings. Programs are marked off by **CR** or **LF** code. Programs are ended with **NULL**.

plCurrentBlockNo: Returns the block # currently executed

Value	Meaning
0	Out of operation
1	1st block
2	2nd block

plRet: Returns an error code. (When using automation interface, returns a return value instead.)

S_OK : Normal termination
EZNC_OPE_GETPRGBLK_ADDR: Part system designation is illegal
EZNC_OPE_GETPRGBLK_DATAERR: Communication data errors between NC and PC
EZNC_OPE_GETPRGBLK_DATASIZE: Too much data for the buffer prepared by the application
EZNC_OPE_GETPRGBLK_NOS: The designation of the number of blocks illegal

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This gets the program which is after the operation search execution or is currently executed. This reads the program after the operation search execution or the current program block in the designated part system.
When out of operation search, it shows
lppwszProgramData="\0"
plCurrentBlockNo=0
Memory area for character strings is saved inside of EZSocket. Memory area for character strings is saved inside of EZSocket. If your client is a VC++ client, it is necessary to release the memory area explicitly by using **CoTaskMemFree()**.
Even though it is out of operation search, it is necessary to release the memory area.

(Note) Mitsubishi CNC C70 Series doesn't support the automation interface.

☐Reference

IEZNcOperation::Search( )

☐Designation

System

2.7.2 IEZNcProgram2::GetProgramNumber2 — Getting the program

☐Calling procedure (Custom interface)

```
HRESULT        GetProgramNumber2(
                    LONG lProgramType,              //  (I)  Program type
                    LPOLESTR* lppwszProgramNo,      //  (O)  Program #
                    LONG* plRet                     //  (O)  Error code
                    )
```

☐Calling procedure (Automation interface)

```
               Program_GetProgramNumber2(
                    lProgramType As LONG            //  (I)  Program type
                    lppwszProgramNo As STRING*      //  (O)  Program #
                    ) As LONG                       //  (O)  Error code
```

☐Argument

lProgramType: Set program type

Value	Meaning
EZNC_MAINPRG	Main program
EZNC_SUBPRG	Sub program

lppwszProgramNo: Returns the program # which has already finished being searched, or which is automatically operated as **UNICODE** character strings. With CNC700, program # is gotton as the program file name.

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK : Normal termination
EZ_ERR_DATA_TYPE: Data type of the argument is illegal
EZNC_DATA_READ_ADDR: Part system designation is illegal
EZNC_DATA_READ_READ: Impossible to read the data

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This returns the program # which has already finished being searched, or which is automatically operated.
Memory area for character strings is saved inside of EZSocket. If your client is a VC++ client, it is necessary to release the memory area explicitly by using **CoTaskMemFree()**.

(Note 1) Mitsubishi CNC C70 Series doesn't support the automation interface.
(Note 2) The existing interface IEZNcProgram::GetProgramNumber() continues to be available for backward compatibility. When newly introducing EZSocket, use this method.

☐Reference

GetSequenceNumber(), GetBlockNumber(), GetSubProLevel()

☐Designation

System

2.7.3 IEZNcProgram2::GetSequenceNumber — Reading the sequence

☐Calling procedure (Custom interface)

```
HRESULT        GetSequenceNumber(
                    LONG lProgramType,          //  (I)  Program type
                    LONG* plSequenceNo,         //  (O)  Sequence #
                    LONG* plRet                 //  (O)  Error code
                    )
```

☐Calling procedure (Automation interface)

```
               Program_GetSequenceNumber(
                    lProgramType As LONG        //  (I)  Program type
                    plSequenceNo As LONG*       //  (O)  Sequence #
                    ) As LONG                   //  (O)  Error code
```

☐Argument

lProgramType: Set program type

Value	Meaning
EZNC_MAINPRG	Main program
EZNC_SUBPRG	Sub program

plSequenceNo: Returns the sequence # which has already finished being searched, or which is automatically operated
Data range: 0 to 99999 (other than **CNC700**)
0 to 999999 (**CNC700**)

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK: Normal termination
EZ_ERR_DATA_TYPE: Data type of the argument is illegal
EZNC_DATA_READ_ADDR: Part system designation is illegal
EZNC_DATA_READ_READ: Impossible to read the data

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This returns the sequence # which has already finished being searched, or which is automatically operated.

(Note) Mitsubishi CNC C70 Series doesn't support the automation interface.

☐Reference

GetProgramNumber2(), GetBlockNumber(), GetSubProLevel()

☐Designation

System

2.7.4 IEZNcProgram2::GetBlockNumber — Reading the block

☐Calling procedure (Custom interface)

```
HRESULT        GetBlockNumber(
                    LONG lProgramType,        //  (I)  Program type
                    LONG* plBlockNo,          //  (O)  Block #
                    LONG* plRet               //  (O)  Error code
                    )
```

☐Calling procedure (Automation interface)

```
               Program_GetBlockNumber(
                    lProgramType As LONG      //  (I)  Program type
                    plBlockNo As LONG*        //  (O)  Block #
                    ) As LONG                 //  (O)  Error code
```

☐Argument

lProgramType: Set program type

Value	Meaning
EZNC_MAINPRG	Main program
EZNC_SUBPRG	Sub program

plBlockNo: Returns the block # which has already finished being searched, or which is automatically operated
Data range: 0 to 99 (other than **CNC700**)
0 to 99999 (**CNC700**)

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK: Normal termination
EZ_ERR_DATA_TYPE: Data type of the argument is illegal
EZNC_DATA_READ_ADDR: Part system designation is illegal
EZNC_DATA_READ_READ: Impossible to read the data

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This returns the block # which has already finished being searched, or which is automatically operated.

(Note) Mitsubishi CNC C70 Series doesn't support the automation interface.

☐Reference

GetSequenceNumber2(), GetSequenceNumber(), GetSubProLevel()

☐Designation

System

2.7.5 IEZNcProgram2::GetSubProLevel — Getting the subprogram level

☐Calling procedure (Custom interface)

```
HRESULT        GetSubProLevel(
                    LONG* plLevel,          //  (O)  Level
                    LONG* plRet             //  (O)  Error code
                    )
```

☐Calling procedure (Automation interface)

```
               Program_GetSubProLevel(
                    plLevel As LONG*        //  (O)  Level
                    ) As LONG               //  (O)  Error code
```

☐Argument

plLevel: Returns the subprogram calling level
Value: 0 to 8

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_READ_ADDR: Part system designation is illegal
EZNC_DATA_READ_READ: Impossible to read the data

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This gets the subprogram level of the designated part system.

(Note) Mitsubishi CNC C70 Series doesn't support the automation interface.

☐Reference

GetProgramNumber2, GetSequenceNumber()

☐Designation

System

2.7.6 IEZNcProgram2::GetInformation — Getting the program information

☐Calling procedure (Custom interface)

```
HRESULT        GetInformation(
                    LONG lInfoType,          //  (I)  Information type
                    LONG* plInfoData,        //  (O)  User machining program information
                    LONG* plRet              //  (O)  Error code
                    )
```

☐Calling procedure (Automation interface)

```
               Program_GetInformation(
                    lInfoType As LONG        //  (I)  Information type
                    plInfoData As LONG*      //  (O)  User machining program information
                    ) As LONG                //  (O)  Error code
```

☐Argument

lInfoType: Set information type to get

Value	Meaning
EZNC_PRG_MAXNUM	The maximum number of pieces of information to register
EZNC_PRG_CURNUM	The number of the current registrations
EZNC_PRG_RESTNUM	How many more pieces of information can be registered
EZNC_PRG_CHARNUM	The number of the registered characters
EZNC_PRG_RESTCHARNUM	How many more characters can be registered (unit: 250 characters)

plInfoData: Returns the program information designated by IInfoType
When designating **EZNC_PRG_MAXNUM**, *lpInfoData* shows
1: 200 pieces of information. The data range and the value depend on the NC card's specifications.

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK: Normal termination
EZ_ERR_DATA_TYPE: Data type of the argument is illegal
EZNC_DATA_READ_READ: Impossible to read the data
EZ_ERR_NOT_SUPPORT: Not suported

☐Return value

Return Value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This gets the program information.

(Note) Mitsubishi CNC C70 Series doesn't support the automation interface.

☐Reference

☐Designation

2.8 IEZNcTime Interface

Magic64	M6x5M	M6x5L Limited	C64	C70	CNC700

2.8.1 IEZNcTime::GetClockData — Getting the date/time

☐Calling procedure (Custom interface)

```
HRESULT        GetClockData(
                    LONG* plDate,        //  (O)  Year, month and day
                    LONG* plTime,        //  (O)  Hour, minute and second
                    LONG* plRet          //  (O)  Error code
                    )
```

☐Calling procedure (Automation interface)

```
               Time_GetClockData(
                    plDate As LONG*      //  (O)  Year, month and day
                    plTime As LONG*      //  (O)  Hour, minute and second
                    ) As LONG            //  (O)  Error code
```

☐Argument

plDate: Returns the date; year, month, and day
Example: 1998/12/25=19981205

plTime: Returns the hour, minute and second of the NC internal clock
Value: 0 to 235959
Example: 23 : 59 : 59 = 235959

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_READ_READ: Impossible to read the data.

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This gets the date and time from the NC internal clock.

(Note) Mitsubishi CNC C70 Series doesn't support the automation interface.

☐Reference

SetClockData()

☐Designation

2.8.2 IEZNcTime::SetClockData — Setting date and time

☐Calling procedure (Custom interface)

```
HRESULT        SetClockData(
                   LONG lDate,        //  (I)  Year, month and day
                   LONG lTime,        //  (I)  Hour, minute and second
                   LONG* plRet        //  (O)  Error code
                   )
```

☐Calling procedure (Automation interface)

```
               Time_SetClockData(
                   lDate As LONG      //  (I)  Year, month and day
                   lTime As LONG      //  (I)  Hour, minute and second
                   ) As LONG          //  (O)  Error code
```

☐Argument

lDate: Set date; year, month, and day
Example: 1998/12/25=19981205

lTime: Set hour, minute and second of the NC internal clock
Value: 0 to 235959
Example: 23 : 59 : 59 = 235959

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_WRITE_WRITE: Impossible to write the data

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

For setting the date and time of the NC internal clock.

(Note) Mitsubishi CNC C70 Series doesn't support the automation interface.

☐Reference

GetClockData()

☐Designation

2.8.3 IEZNcTime::GetAliveTime — Getting the power ON time

☐Calling procedure (Custom interface)

```
HRESULT        GetAliveTime(
                    LONG* plTime,        //  (O)  Power ON time
                    LONG* plRet          //  (O)  Error code
                    )
```

☐Calling procedure (Automation interface)

```
               Time_GetAliveTime(
                    plTime As LONG*      //  (O)  Power ON time
                    ) As LONG            //  (O)  Error code
```

☐Argument

plTime: This gets the total power ON time (in hour, minute and second format) from the power ON to OFF of the control unit.
Value: 0 to 99995959 (0 to 599995959 when **CNC700** is used)
Exmple: 9999 : 59 : 59 = 99995959 (59999 : 59 : 59 = 599995959 when **CNC700** is used)

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_READ_READ: Impossible to read the data

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This gets the total power ON time (in hour, minute and second format) from the power ON to OFF of the control unit. When the total time reaches the maximum, the system stops counting and holds the maximum amount of time.

(Note) Mitsubishi CNC C70 Series doesn't support the automation interface.

☐Reference

SetAliveTime()

☐Designation

2.8.4 IEZNcTime::SetAliveTime — Setting power ON time

☐Calling procedure (Custom interface)

```
HRESULT        SetAliveTime(
                    LONG lTime,          //  (I)  Power ON time
                    LONG* plRet          //  (O)  Error code
                    )
```

☐Calling procedure (Automation interface)

```
               Time_SetAliveTime(
                    lTime As LONG        //  (I)  Power ON time
                    ) As LONG            //  (O)  Error code
```

☐Argument

lTime: Set the total power ON time (in hour, minute and second format) from the power ON to OFF of the control unit.
Value: 0 to 99995959 (0 to 599995959 when **CNC700** is used)
Example: 9999 : 59 : 59 = 99995959 (59999 : 59 : 59 = 599995959 when **CNC700** is used)

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_WRITE_WRITE: Impossible to write the data

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This compulsory sets the total power ON time (in hour, minute and second format) from the power ON to OFF of the control unit.

(Note) Mitsubishi CNC C70 Series doesn't support the automation interface.

☐Reference

GetAliveTime()

☐Designation

2.8.5 IEZNcTime::GetRunTime — Getting the automatic operation time

☐Calling procedure (Custom interface)

```
HRESULT        GetRunTime(
                   LONG* plTime,        //  (O)  Automatic operation time
                   LONG* plRet          //  (O)  Error code
                   )
```

☐Calling procedure (Automation interface)

```
               Time_GetRunTime(
                   plTime As LONG*      //  (O)  Automatic operation time
                   ) As LONG            //  (O)  Error code
```

☐Argument

plTime: Returns the total machining time (in hour, minute and second format) from the automatic operation is started in the memory (tape) or MDI mode until it is stopped by M02/M30 or reset.
Value: 0 to 99995959 (0 to 599995959 when **CNC700** is used)
Example: 9999 : 59 : 59 = 99995959 (59999 : 59 : 59 = 599995959 when **CNC700** is used)

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_READ_READ: Impossible to read the data

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This gets the total machining time (in hour, minute and second format) from the automatic operation is started in the memory (tape) or MDI mode until it is stopped by M02/M30 or reset. When the total time reaches the maximum, the system stops counting and holds the maximum amount of time.

(Note) Mitsubishi CNC C70 Series doesn't support the automation interface.

☐Reference

SetRunTime()

☐Designation

2.8.6 IEZNcTime::SetRunTime — Setting automatic operation time

☐Calling procedure (Custom interface)

```
HRESULT        SetRunTime(
                   LONG lTime,        //  (I)  Automatic operation time
                   LONG* plRet        //  (O)  Error code
                   )
```

☐Calling procedure (Automation interface)

```
               Time_SetRunTime(
                   lTime As LONG      //  (I)  Automatic operation time
                   ) As LONG          //  (O)  Error code
```

☐Argument

lTime: Set the total machining time (in hour, minute and second format) from the automatic operation is started in the memory (tape) or MDI mode until it is stopped by M02/M30 or reset.
Value: 0 to 99995959 (0 to 599995959 when **CNC700** is used)
Example: 9999 : 59 : 59 = 99995959 (59999 : 59 : 59 = 599995959 when **CNC700** is used)

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_WRITE_WRITE: Impossible to write the data

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This compulsory sets the total machining time (in hour, minute and second format) from the automatic operation is started in the memory (tape) or MDI mode until it is stopped by M02/M30 or reset.

(Note) Mitsubishi CNC C70 Series doesn't support the automation interface.

☐Reference

GetRunTime()

☐Designation

2.8.7 IEZNcTime::GetStartTime — Getting the automatic start time

☐Calling procedure (Custom interface)

```
HRESULT        GetStartTime(
                    LONG* plTime,        //  (O)  Automatic start time
                    LONG* plRet          //  (O)  Error code
                    )
```

☐Calling procedure (Automation interface)

```
               Time_GetStartTime(
                    plTime As LONG*      //  (O)  Automatic start time
                    ) As LONG            //  (O)  Error code
```

☐Argument

plTime: Returns the total automatic start time (in hour, minute and second format) from the automatic operation is started in the memory (tape) or MDI mode until it is stopped by feed hold, block stop, or reset.
Value: 0 to 99995959 (0 to 599995959 when **CNC700** is used)
Exmple: 9999 : 59 : 59 = 99995959 (59999 : 59 : 59 = 599995959 when **CNC700** is used)

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_READ_READ: Impossible to read the data.

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This gets the total automatic start time (in hour, minute and second format) from the automatic operation is started in the memory (tape) or MDI mode until it is stopped by feed hold, block stop, or reset.

(Note) Mitsubishi CNC C70 Series doesn't support the automation interface.

☐Reference

SetStartTime()

☐Designation

2.8.8 IEZNcTime::SetStartTime — Setting the automatic start time

☐Calling procedure (Custom interface)

```
HRESULT        SetStartTime(
                    LONG lTime,          //  (I)  Automatic start time
                    LONG* plRet          //  (O)  Error code
                    )
```

☐Calling procedure (Automation interface)

```
               Time_SetStartTime(
                    lTime As LONG        //  (I)  Automatic start time
                    ) As LONG            //  (O)  Error code
```

☐Argument

lTime: Set the total automatic start time (in hour, minute and second format) from the automatic operation is started in the memory (tape) or MDI mode until it is stopped by feed hold, block stop, or reset.
Value: 0 to 99995959 (0 to 599995959 when **CNC700** is used)
Exmple: 9999 : 59 : 59 = 99995959 (59999 : 59 : 59 = 599995959 when **CNC700** is used)

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_WRITE_WRITE: Impossible to write the data

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This sets the total automatic start time (in hour, minute and second format) from the automatic operation is started in the memory (tape) or MDI mode until it is stopped by feed hold, block stop, or reset.

(Note) Mitsubishi CNC C70 Series doesn't support the automation interface.

☐Reference

GetStartTime()

☐Designation

2.8.9 IEZNcTime::GetEstimateTime — Getting the external elapsed time

☐Calling procedure (Custom interface)

```
HRESULT        GetEstimateTime(
                    LONG lKind,          //  (I)  Type of external elapsed time
                    LONG* plTime,        //  (O)  External elapsed time
                    LONG* plRet          //  (O)  Error code
                    )
```

☐Calling procedure (Automation interface)

```
               Time_GetEstimateTime(
                    lKind As LONG        //  (I)  Type of external elapsed time
                    plTime As LONG*      //  (O)  External elapsed time
                    ) As LONG            //  (O)  Error code
```

☐Argument

lKind: Set type of external elapsed time

Value	Meaning
0	Counted when PLC device Y234 is ON (Y344 for **M6x5M**; Y704 for **CNC700**)
1	Counted when PLC device Y235 is ON (Y345 for **M6x5M**; Y705 for **CNC700**)

plTime: Returns the time managed by PLC. It is expressed with hour, minute, and second. When the elapsed time gets to the maximum, 9999:59:59, it stops counting. The display keeps showing the maximum time.
Value: 0 to 99995959
Example: 9999 : 59 : 59 = 99995959

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_READ_READ: Impossible to read the data

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This returns the time managed by PLC. It is expressed with hour, minute, and second. It starts counting when user PLC device turns ON. Device # differs according to the model. Check the device # with the respective PLC interface manual.

(Note) Mitsubishi CNC C70 Series doesn't support the automation interface.

☐Reference

SetEstimateTime()

☐Designation

2.8.10 IEZNcTime::SetEstimateTime — Setting external elapsed time

☐Calling procedure (Custom interface)

```
HRESULT        SetEstimateTime(
                    LONG lKind,        //  (I)  Type of external elapsed time
                    LONG lTime,        //  (I)  External elapsed time
                    LONG* plRet        //  (O)  Error code
                    )
```

☐Calling procedure (Automation interface)

```
               Time_SetEstimateTime(
                    lKind As LONG      //  (I)  Type of external elapsed time
                    lTime As LONG      //  (I)  External elapsed time
                    ) As LONG          //  (O)  Error code
```

☐Argument

lKind: Set type of external elapsed time

Value	Meaning
0	Counted when PLC device Y234 is ON (Y344 for **M6x5M**; Y704 for **CNC700**)
1	Counted when PLC device Y235 is ON (Y345 for **M6x5M**; Y705 for **CNC700**)

lTime: Returns the time managed by PLC. It is expressed with hour, minute, and second.
When the elapsed time gets to the maximum, 9999:59:59, it stops counting. The display keeps showing the maximum time.
Value: 0 to 99995959
Example: 9999 : 59 : 59 = 99995959

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_WRITE_WRITE: Impossible to write the data

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This sets the time managed by PLC. It is expressed with hour, minute, and second. It starts counting when user PLC device turns ON. Device # differs according to the model. Check the device # with the respective PLC interface manual.

(Note) Mitsubishi CNC C70 Series doesn't support the automation interface.

☐Reference

GetEstimateTime()

☐Designation

2.9 IEZNcAxisMonitor Interface

Magic64 Limited	M6x5M	M6x5L	C64		CNC700

2.9.1 IEZNcAxisMonitor::GetServoMinitor — Servo monitor

☐Calling procedure (Custom interface)

```
HRESULT        GetServoMonitor(
                    LONG lAxisNo,          //  (I)  Axis designation
                    LONG lIndex,           //  (I)  Designation of monitor data
                    LONG* plData,          //  (O)  Monitor data
                    LPOLESTR* lppwszBuffer, //  (O)  Monitor data character strings
                    LONG* plRet            //  (O)  Error code
                    )
```

☐Calling procedure (Automation interface)

```
               Monitor_GetServoMonitor(
                    lAxisNo As LONG        //  (I)  Axis designation
                    lIndex As LONG         //  (I)  Designation of monitor data
                    plData As LONG*        //  (O)  Monitor data
                    lppwszBuffer As STRING* //  (O)  Monitor data character strings
                    ) As LONG              //  (O)  Error code
```

☐Argument

lAxisNo: Set axis # (“1” or later)

lIndex: Set parameter # of the designated axis of the designated part system
(With M6x5L, remaining command, manual interpolation amount and alarm 4 are not supported.)

plData: Returns the axis condition

lppwszBuffer: Outputs the data (return value) with UNICODE character strings to the designated value 100 to 104 in lIndex.
With CNC700, outputs the data (return value) with UNICODE character strings to the designated value 11 to 15, 18 to 20 or 100 to 104 in lIndex.

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_READ_READ: Impossible to read the data
EZNC_DATA_READ_DATASIZE: Too much data for the buffer prepared by the application
EZNC_DATA_READ_DATATYPE: Data type (parameter #) illegal
EZNC_DATA_READ_ADDR: Part system designation or axis designation is illegal

lIndex	Description	Data range	Magic64	Remark
0	GAIN. Status of position loop gain	Unit: 1/sec	Unavailable	
1	DROOP. (Tracking delay)	Unit: i	Unavailable	
2	SPEED. The actual speed of revolutions.	0 and above [rpm]	Available	
3	CURRENT. Load current. Current of motors. (When stalling, displays the result of consecutive current conversion.)	0 and above [%]	Available	
4	MAXCUR1. MAX current I.	Unit: %	Unavailable	
5	MAXCUR2. MAX current II.	Unit: %	Unavailable	
6	OVER LOAD. Overload.	Unit: %	Unavailable	
7	OVER REG. Regenerative load.	Unit: %	Unavailable	
10	CYC CNT. Cycle counter.	Unit: Pulse	Unavailable	

☐Argument

11	GRIDSP. Grid interval (space).	Unit: Command unit	Unavailable	
12	GRID. Grid amount.	Unit: Command unit	Unavailable	
13	MACPOS. Machine position.	Unit: Command unit	Unavailable	
14	MOT POS. Motor end FB.	Unit: Command unit	Unavailable	
15	SCA POS. Machine end FB.	Unit: Command unit	Unavailable	
16	FB ERROR. FB error.	Unit: i	Unavailable	
17	DFB COMP. DFB compensation amount.		Unavailable	
18	Remaining command	Unit: Command unit	Unavailable	
19	Current value	Unit: Command unit	Unavailable	
20	Manual interruption amount	Unit: Command unit	Unavailable	
100	AMP DISP. Amplifier (drive unit) display. 7 segment LED display.	"00\0" to "FF\0" This is output as 3 character string.	Available	
101	Alarm 1	This is output as 3 character string.	Unavailable	
102	Alarm 2	Ditto	Unavailable	
103	Alarm 3	Ditto	Unavailable	
104	Alarm 4	Ditto	Unavailable	

*In the case of CNC700, gets the value converted according to the command unit (actual value) as character string.

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This gets the servo monitor information of the designated axis of the designated part system.

When the data range in the IIndex table is "Unit: Command unit", it is necessary to convert the values which are got according to the command unit set in the NC. For command units, check the specifications of the NC.

<In the case of linear axis>

	Metric system	Conversion rate (LONG 1=)
10 micron spec	-999999.99 to 999999.99	1=1/200(mm)
1 micron spec	-99999.999 to 99999.999	1=1/2000(mm)
0.1 micron spec	-9999.9999 to 9999.9999	1=1/20000(mm)

	Inch system	Conversion rate (LONG 1=)
10 micron spec	-99999.999 to 99999.999	1=1/2000(inch)
1 micron spec	-9999.9999 to 9999.9999	1=1/20000(inch)
0.1 micron spec	-999.99999 to 999.99999	1=1/200000(inch)

<In the case of rotary axis>

	Metric system	Conversion rate (LONG 1=)
10 micron spec	-999999.99 to 999999.99	1=1/200(mm)
1 micron spec	-99999.999 to 99999.999	1=1/2000(mm)
0.1 micron spec	-9999.9999 to 9999.9999	1=1/20000(mm)

	Inch system	Conversion rate (LONG 1=)
10 micron spec	-999999.99 to 999999.99	1=1/200(inch)
1 micron spec	-99999.999 to 99999.999	1=1/2000(inch)
0.1 micron spec	-9999.9999 to 9999.9999	1=1/20000(Inch)

☐**Functions** E.g. of conversion) In the case of a linear axis, 1 micron spec and the metric system. If the LONG value you got is 710001,

710001/2000 = 355.0005, then rounded down to 355.000.

Data dealt with a 0.5 micron (1/2000mm) increment are rounded down to the minus side to display.

Thus, +0.5 miscon is rounded downt to 0, -0.5 micron is rounded down to –1 micron.

Memory area for character strings is saved inside of EZSocket. If your client is a VC++ client, it is necessary to release the memory area explicitly by using **CoTaskMemFree()**.

☐**Reference** **GetServoVersion(), GetServoDiagnosis()**

☐**Designation** System, PLC axis, Axis

2.9.2 IEZNcAxisMonitor::GetServoVersion — Getting the servo information

☐Calling procedure (Custom interface)

```
HRESULT        GetServoVersion(
                    LONG lAxisNo,              //  (I)  Axis designation
                    LONG lIndex,               //  (I)  Designation of servo information
                    LPOLESTR* lppwszBuffer,    //  (O)  Servo information
                    LONG* plRet                //  (O)  Error code
                    )
```

☐Calling procedure (Automation interface)

```
               Monitor_GetServoVersion(
                    lAxisNo As LONG            //  (I)  Axis designation
                    lIndex As LONG             //  (I)  Designation of servo information
                    lppwszBuffer As STRING*    //  (O)  Servo information
                    ) As LONG                  //  (O)  Error code
```

☐Argument

lAxisNo: Set axis # (“1” or later)

lIndex: Set Servo information. Refer to the table below.

lppwszBuffer: Gets the servo information as **UNICODE** character strings

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_READ_READ: Impossible to read the data
EZNC_DATA_READ_ADDR: Part system designation or the axis designation is illegal

lIndex	Description	Data range
0	Drive unit model name	Character string within 17 letters
1	Drive unit serial #	Character string within 9 letters
2	S/W version	Character string within 17 letters
3	Control method	Character string within 7 letters
4	Motor end detector	Character string within 9 letters
5	Machine end detector	Character string within 9 letters
6	Motor model name	Character string within 9 letters

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This gets the servo’s version information.
Memory area for character strings is saved inside of EZSocket. If your client is a VC++ client, it is necessary to release the memory area explicitly by using **CoTaskMemFree()**.

☐Reference

GetServoMonitor(), GetServoDiagnosis()

☐Designation

System, PLC axis, Axis

2.9.3 IEZNcAxisMonitor::GetServoDiagnosis — Getting the servo diagnosis information

☐Calling procedure (Custom interface)

```
HRESULT        GetServoDiagnosis(
                    LONG lAxisNo,          //  (I)  Axis designation
                    LONG lIndex,           //  (I)  Designation of the diagnosis information
                    LONG* plData,          //  (O)  Diagnosis information value
                    LPOLESTR*              //  (O)  Diagnosis information character strings
                    lppwszBuffer,          //  (O)  Error code
                    LONG* plRet
                    )
```

☐Calling procedure (Automation interface)

```
Monitor_GetServoDiagnosis(
                    lAxisNo As LONG        //  (I)  Axis designation
                    lIndex As LONG         //  (I)  Designation of the diagnosis information
                    plData As LONG*        //  (O)  Diagnosis information value
                    lppwszBuffer As STRING*//  (O)  Diagnosis information character strings
                    ) As LONG              //  (O)  Error code
```

☐Argument

lAxisNo: Set axis # (“1” or later)

lIndex: Set the servo diagnosis information. Refer to the table below.

plData: Returns the value of the servo diagnosis information

lppwszBuffer: Gets the diagnosis information as **UNICODE** character strings. Outputs character strings to the designated value 21 to 30 in lIndex.

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_READ_READ: Impossible to read the data
EZNC_DATA_READ_ADDR: Part system designation or the axis designation is illegal

lIndex	Description	Data range
0	Accumulated time of Ready ON	
1	Alarm history 1 (#)	Servo alarm # that occurred in the past
2	Alarm history 2 (#)	Ditto
3	Alarm history 3 (#)	Ditto
4	Alarm history 4 (#)	Ditto
5	Alarm history 5 (#)	Ditto
6	Alarm history 6 (#)	Ditto
7	Alarm history 7 (#)	Ditto
8	Alarm history 8 (#)	Ditto
11	Alarm history 1 (Time)	Servo alarm time that occurred in the past
12	Alarm history 2 (Time)	Ditto
13	Alarm history 3 (Time)	Ditto
14	Alarm history 4 (Time)	Ditto
15	Alarm history 5 (Time)	Ditto
16	Alarm history 6 (Time)	Ditto
17	Alarm history 7 (Time)	Ditto
18	Alarm history 8 (Time)	Ditto

☐Argument

21	MNT (1)	Character string within 3 letters
22	MNT (2)	Ditto
23	MNT (3)	Ditto
24	MNT (4)	Ditto
30	SYS	Character string within 2 letters

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This gets the servo diagnosis information.
Memory area for character strings is saved inside of EZSocket. If your client is a VC++ client, it is necessary to release the memory area explicitly by using **CoTaskMemFree()**.

☐Reference

GetServoMonitor(), GetServoVersion()

☐Designation

System, PLC axis, Axis

2.9.4 IEZNcAxisMonitor::GetPowerVersion — Getting the power supply version information

☐Calling procedure (Custom interface)

```
HRESULT        GetPowerVersion(
                    LONG lAxisNo,              //  (I)  Axis designation
                    LONG lIndex,               //  (I)  Designation of the version information
                    LPOLESTR* lppwszBuffer,    //  (O)  Version information character string
                    LONG* plRet                //  (O)  Error code
                    )
```

☐Calling procedure (Automation interface)

```
               Monitor_GetPowerVersion(
                    lAxisNo As LONG            //  (I)  Axis designation
                    lIndex As LONG             //  (I)  Designation of the version information
                    lppwszBuffer As STRING*    //  (O)  Version information character string
                    ) As LONG                  //  (O)  Error code
```

☐Argument

lAxisNo: Set axis # (“1” or later)

lIndex: Set the version information. Refer to the table below.

lppwszBuffer: Gets the version information as **UNICODE** character strings

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_READ_READ: Impossible to read the data
EZNC_DATA_READ_ADDR: Part system designation or the axis designation is illegal

lIndex	Description	Data range
0	Unit model name	Character string within 17 letters
1	Unit serial #	Character string within 9 letters
2	S/W version	Character string within 17 letters
3	Connection drive	1 letter

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This gets the power supply’s version information.
Memory area for character strings is saved inside of EZSocket. If your client is a VC++ client, it is necessary to release the memory area explicitly by using **CoTaskMemFree()**.

☐Reference

GetPowerDiagnosis()

☐Designation

Axis

2.9.5 IEZNcAxisMonitor::GetPowerDiagnosis — Getting the power supply diagnosis information

☐Calling procedure (Custom interface)

```
HRESULT        GetPowerDiagnosis(
                    LONG lAxisNo,          //  (I)  Axis designation
                    LONG lIndex,           //  (I)  Designation of the diagnosis information
                    LONG* plData,          //  (O)  Diagnosis information value
                    LPOLESTR*              //  (O)  Diagnosis information character strings
                    lppwszBuffer,          //  (O)  Error code
                    LONG* plRet
                    )
```

☐Calling procedure (Automation interface)

```
               Monitor_GetPowerDiagnosis(
                    lAxisNo As LONG        //  (I)  Axis designation
                    lIndex As LONG         //  (I)  Designation of the diagnosis information
                    plData As LONG*        //  (O)  Diagnosis information value
                    lppwszBuffer As STRING*//  (O)  Diagnosis information character strings
                    ) As LONG              //  (O)  Error code
```

☐Argument

lAxisNo: Set axis # (“1” or later)

lIndex: Set the diagnosis information. Refer to the table below.

plData: Returns the value of the diagnosis information

lppwszBuffer: Gets the diagnosis information as **UNICODE** character strings

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_READ_READ: Impossible to read the data
EZNC_DATA_READ_ADDR: Part system designation or the axis designation is illegal

lIndex	Description	Data range
0	Accumulated time of Ready ON	
1	Alarm history 1 (#)	Power supply alarm # that occurred in the past
2	Alarm history 2 (#)	Ditto
3	Alarm history 3 (#)	Ditto
4	Alarm history 4 (#)	Ditto
5	Alarm history 5 (#)	Ditto
6	Alarm history 6 (#)	Ditto
7	Alarm history 7 (#)	Ditto
8	Alarm history 8 (#)	Ditto
11	Alarm history 1 (Time)	Power supply alarm time that occurred in the past
12	Alarm history 2 (Time)	Ditto
13	Alarm history 3 (Time)	Ditto
14	Alarm history 4 (Time)	Ditto
15	Alarm history 5 (Time)	Ditto
16	Alarm history 6 (Time)	Ditto
17	Alarm history 7 (Time)	Ditto
18	Alarm history 8 (Time)	Ditto

☐Argument

21	MNT (1)	Character string within 3 letters
22	MNT (2)	Ditto
23	MNT (3)	Ditto
24	MNT (4)	Ditto
30	SYS	Character string within 2 letters

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This gets the power supply diagnosis information.
Memory area for character strings is saved inside of EZSocket. If your client is a VC++ client, it is necessary to release the memory area explicitly by using **CoTaskMemFree()**.

☐Reference

GetPowerVersion()

☐Designation

Axis

2.9.6 IEZNcAxisMonitor::GetSpindleMonitor — Spindle monitor

☐Calling procedure (Custom interface)

```
HRESULT        GetSpindleMonitor(
                    LONG lIndex,              //  (I)  Designation of monitor data
                    LONG lSpindle,            //  (I)  Spindle #
                    LONG* plData,             //  (O)  Monitor data value
                    LPOLESTR* lppwszBuffer,   //  (O)  Monitor data character strings
                    LONG* plRet               //  (O)  Error code
                    )
```

☐Calling procedure (Automation interface)

```
Monitor_GetSpindleMonitor(
                    lIndex As LONG            //  (I)  Designation of monitor data
                    lSpindle As LONG          //  (I)  Spindle #
                    plData As LONG*           //  (O)  Monitor data value
                    lppwszBuffer As STRING*   //  (O)  Monitor data character strings
                    ) As LONG                 //  (O)  Error code
```

☐Argument

lIndex: Set parameter # of the designated spindle

lSpindle: Set spindle #

plData: Returns the spindle status

lppwszBuffer: Outputs the spindle information as **UNICODE** character strings

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_READ_READ: Impossible to read the data
EZNC_DATA_READ_ADDR: Part system designation or the axis designation is illegal

lIndex	Description	Data range	Magic64	Remark
0	Gain. Spindle position loop gain.	Unit: 1/sec	Unavailable	
1	Droop. Position error amount.	Unit: i	Unavailable	
2	Spindle speed (SR, SF). The actual speed of the spindle motor. Override included.	0 and above [rpm]	Available	
3	Load. Load of the spindle motor.	0 and above [%]	Available	
4	Amplifier (drive unit) display. 7 segment LED display.	"00\0" to "FF\0" This is output as 3 character string.	Available	
5	Alarm 1	Character string within 3 letters	Unavailable	
6	Alarm 2	Ditto	Unavailable	
7	Alarm 3	Ditto	Unavailable	
10	Cycle counter		Unavailable	
11	Control input 1		Unavailable	
12	Control input 2		Unavailable	
13	Control input 3		Unavailable	
14	Control input 4		Unavailable	
15	Control output 1		Unavailable	
16	Control output 2		Unavailable	
17	Control output 3		Unavailable	
18	Control output 4		Unavailable	

☐**Return value**	Return value	Meaning
	S_OK	Normal termination
	S_FALSE	Communication failure
☐**Functions**	This gets the status of the designated spindle.	
☐**Reference**	**GetSpindleVersion(), GetSpindleDiagnosis()**	
☐**Designation**		

2.9.7 IEZNcAxisMonitor::GetSpindleVersion — Getting the spindle version information

☐Calling procedure (Custom interface)

```
HRESULT        GetSpindleVersion(
                    LONG lAxisNo,              //  (I)   Axis designation
                    LONG lIndex,               //  (I)   Designation of the version information
                    LPOLESTR* lppwszBuffer,    //  (O)   Version information character string
                    LONG* plRet                //  (O)   Error code
                    )
```

☐Calling procedure (Automation interface)

```
               Monitor_GetSpindleVersion(
                    lAxisNo As LONG            //  (I)   Axis designation
                    lIndex As LONG             //  (I)   Designation of the version information
                    lppwszBuffer As STRING*    //  (O)   Version information character string
                    ) As LONG                  //  (O)   Error code
```

☐Argument

lAxisNo: Set axis # (“1” or later)

lIndex: Set the version information. Refer to the table below.

lppwszBuffer: Gets the version information as **UNICODE** character strings

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_READ_READ: Impossible to read the data
EZNC_DATA_READ_ADDR: Part system designation or the axis designation is illegal

lIndex	Description	Data range
0	Drive unit model name	Character string within 17 letters
1	Drive unit serial #	Character string within 9 letters
2	S/W version	Character string within 17 letters

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This gets the spindle’s version information.
Memory area for character strings is saved inside of EZSocket. If your client is a VC++ client, it is necessary to release the memory area explicitly by using **CoTaskMemFree()**.

☐Reference

GetSpindleMonitor(), GetSpindleDiagnosis()

☐Designation

Axis

2.9.8 IEZNcAxisMonitor::GetSpindleDiagnosis — Getting the spindle diagnosis information

☐Calling procedure (Custom interface)

```
HRESULT        GetSpindleDiagnosis(
                    LONG lAxisNo,              // (I)  Axis designation
                    LONG lIndex,               // (I)  Designation of the diagnosis information
                    LONG* plData,              // (O) Diagnosis information value
                    LPOLESTR* lppwszBuffer,    // (O) Diagnosis information character strings
                    LONG* plRet                // (O) Error code
                    )
```

☐Calling procedure (Automation interface)

```
Monitor_GetSpindleDiagnosis(
                    lAxisNo As LONG            // (I)  Axis designation
                    lIndex As LONG             // (I)  Designation of the diagnosis information
                    plData As LONG*            // (O) Diagnosis information value
                    lppwszBuffer As STRING*    // (O) Diagnosis information character strings
                    ) As LONG                  // (O) Error code
```

☐Argument

lAxisNo: Set axis # (“1” or later)

lIndex: Set the diagnosis information

plData: Returns the value of the diagnosis information

lppwszBuffer: Gets the diagnosis information as **UNICODE** character strings

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_READ_READ: Impossible to read the data
EZNC_DATA_READ_ADDR: Part system designation or the axis designation is illegal

lIndex	Description	Data range
0	Accumulated time of Ready ON	
1	Alarm history 1 (#)	Spindle alarm # that occurred in the past
2	Alarm history 2 (#)	Ditto
3	Alarm history 3 (#)	Ditto
4	Alarm history 4 (#)	Ditto
5	Alarm history 5 (#)	Ditto
6	Alarm history 6 (#)	Ditto
7	Alarm history 7 (#)	Ditto
8	Alarm history 8 (#)	Ditto
11	Alarm history 1 (Time)	Spindle alarm time that occurred in the past
12	Alarm history 2 (Time)	Ditto
13	Alarm history 3 (Time)	Ditto
14	Alarm history 4 (Time)	Ditto
15	Alarm history 5 (Time)	Ditto
16	Alarm history 6 (Time)	Ditto
17	Alarm history 7 (Time)	Ditto
18	Alarm history 8 (Time)	Ditto
21	MNT (1)	Character string within 3 letters
22	MNT (2)	Ditto
23	MNT (3)	Ditto
24	MNT (4)	Ditto
30	SYS	Character string within 2 letters

☐**Return value**	Return value	Meaning
	S_OK	Normal termination
	S_FALSE	Communication failure
☐**Functions**	This gets the spindle diagnosis information. Memory area for character strings is saved inside of EZSocket. If your client is a VC++ client, it is necessary to release the memory area explicitly by using **CoTaskMemFree()**.	
☐**Reference**	**GetSpindleMonitor(), GetSpindleVersion()**	
☐**Designation**	Axis	

2.9.9 IEZNcAxisMonitor::GetAbsPositionMonitor — Getting the absolute position monitor information

☐Calling procedure (Custom interface)

```
HRESULT        GetAbsPositionMonitor(
                    LONG lAxisNo,                  //  (I)  Axis designation
                    LONG lIndex,                   //  (I)  Designation of monitor information
                    LONG* plData,                  //  (O)  Monitor information value
                    LPOLESTR* lppwszBuffer,        //  (O)  Absolute position monitor
                                                           information character strings
                    LONG* plRet                    //  (O)  Error code
                    )
```

☐Calling procedure (Automation interface)

```
Monitor_ GetAbsPositionMonitor (
                    lAxisNo As LONG                //  (I)  Axis designation
                    lIndex As LONG                 //  (I)  Designation of monitor information
                    plData As LONG*                //  (O)  Monitor information value
                    lppwszBuffer As STRING*        //  (O)  Absolute position monitor
                                                           information character strings
                    ) As LONG                      //  (O)  Error code
```

☐Argument

lAxisNo: Set axis # (“1” or later)

lIndex: Set the monitor information.

plData: Returns the value of the monitor information.

lppwszBuffer: Gets the absolute position monitor information as **UNICODE** character strings *lindex* outputs character string to 0.
With **CNC700**, outputs the result of **0 to 3** of *lIndex* as **UNICODE** character strings.

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_READ_READ: Impossible to read the data
EZNC_DATA_READ_ADDR: Part system designation or the axis designation is illegal

lIndex	Description	Data range	Part system	Axis	PLC axis
0	ABS SYS. Detection system.	0: Semi-closed encoder (ES) 1: Semi-closed high speed serial encoder (ESS) 2: Incremental(INC)	Available	Available	Available
1	POF POS. Power OFF position.	Unit: Command unit	Available	Available	Available
2	PON POS. Power ON position.	Unit: Command unit	Available	Available	Available
3	MAC POS. Current position.	Unit: Command unit	Available	Available	Available
4	R0		Available	Available	Available
5	P0		Available	Available	Available
6	E0		Available	Available	Available
7	Rn		Available	Available	Available
8	Pn		Available	Available	Available
9	En		Available	Available	Available
10	ABSn		Available	Available	Available
11	ABS0		Available	Available	Available

*In the case of CNC700, gets the value converted according to the command unit (actual value) as character string. For the command unit conversion, refer to ☐ Function of IEZNcAxisMonitor::GetServoMonitor().

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This gets the absolute position monitor information. ABS0 is valid only with **CNC700**.
Memory area for character strings is saved inside of EZSocket. If your client is a VC++ client, it is necessary to release the memory area explicitly by using **CoTaskMemFree()**.

☐Reference

☐Designation

System, PLC axis, Axis

2.9.10 IEZNcAxisMonitor::GetAuxAxisMonitor — Getting the auxiliary axis monitor information

☐Calling procedure (Custom interface)

```
HRESULT        GetAuxAxisMonitor(
                    LONG lAxisNo,              //  (I)  Axis designation
                    LONG lIndex,               //  (I)  Auxiliary axis information type
                    LONG* plData,              //  (O)  Auxiliary axis information value
                    LPOLESTR* lppwszBuffer,    //  (O)  Auxiliary axis monitor information
                                                        character strings
                    LONG* plRet                //  (O)  Error code
                    )
```

☐Calling procedure (Automation interface)

```
Monitor_GetAuxAxisMonitor (
                    lAxisNo As LONG            //  (I)  Axis designation
                    lIndex As LONG             //  (I)  Auxiliary axis information type
                    plData As LONG*            //  (O)  Auxiliary axis information value
                    lppwszBuffer As STRING*    //  (O)  Auxiliary axis monitor information
                                                        character strings
                    ) As LONG                  //  (O)  Error code
```

☐Argument

lAxisNo: Set axis # (“1” or later)

lIndex: Set the type of the auxiliary axis information. Refer to the table below.

plData: Returns the value of the auxiliary axis information

lppwszBuffer: Gets the monitor information as **UNICODE** character strings

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_READ_READ: Impossible to read the data

lIndex	Description	Data range
0	Droop	-999 to 999
1	Rotation speed	0 to 9999 [rpm]
2	Load current	-999 to 999 [%]
3	MAX current 1	-999 to 999 [%]
4	MAX current 2	-999 to 999 [%]
5	Load ratio	-999 to 999 [%]
6	Regenerative load	-999 to 999 [%]
7	Current st_No	1 to 360
8	Current position	-99999.999 to 99999.999
9	Target st_No	1 to 360
10	Command position	-99999.999 to 99999.999
11	Position control gain 1	0 to 999
12	Speed control gain 1	0 to 999
13	Position control gain 2	0 to 999
14	Speed control gain 2	0 to 999
15	Speed integral compensation	0 to 999
16	Load inertia ratio	0 to 999.9

☐**Return value**	Return value	Meaning
	S_OK	Normal termination
	S_FALSE	Communication failure
☐**Functions**	This gets the auxiliary axis monitor information. Memory area for character strings is saved inside of EZSocket. If your client is a VC++ client, it is necessary to release the memory area explicitly by using **CoTaskMemFree()**.	
☐**Reference**		
☐**Designation**	Axis	

M6x5M C64 CNC700

2.9.11 IEZNcAxisMonitor::GetAuxAxisDiagnosis — Getting the auxiliary axis diagnosis information

☐Calling procedure (Custom interface)

```
HRESULT        GetAuxAxisDiagnosis(
                    LONG lAxisNo,                  //  (I)  Axis designation
                    LONG lIndex,                   //  (I)  Auxiliary axis diagnosis
                    LPOLESTR* lppwszBuffer,                information type
                                                   //  (O)  Auxiliary axis diagnosis
                    LONG* plRet                            information character strings
                                                   //  (O)  Error code
                    )
```

☐Calling procedure (Automation interface)

```
               Monitor_GetAuxAxisDiagnosis(
                    lAxisNo As LONG                //  (I)  Axis designation
                    lIndex As LONG                 //  (I)  Auxiliary axis diagnosis
                    lppwszBuffer As STRING*                information type
                                                   //  (O)  Auxiliary axis diagnosis
                    ) As LONG                              information character strings
                                                   //  (O)  Error code
```

☐Argument

lAxisNo: Set axis # (“1” or later)

lIndex: Set the type of the auxiliary axis diagnosis information. Refer to the table below.

lppwszBuffer: Gets the auxiliary axis diagnosis information as **UNICODE** character strings

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_READ_READ: Impossible to read the data
EZNC_DATA_READ_ADDR: Part system designation or the axis designation is illegal

lIndex	Description	Data range
0	Alarm history 1	Character string within 9 letters "Alarm information by type"
1	Alarm history 2	Character string within 9 letters "Alarm information by type"
2	Alarm history 3	Character string within 9 letters "Alarm information by type"
3	Alarm history 4	Character string within 9 letters "Alarm information by type"
4	Alarm history 5	Character string within 9 letters "Alarm information by type"
5	Alarm history 6	Character string within 9 letters "Alarm information by type"

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This gets the auxiliary axis diagnosis information.
Memory area for character strings is saved inside of EZSocket. If your client is a VC++ client, it is necessary to release the memory area explicitly by using **CoTaskMemFree()**.

☐Reference

☐Designation

Axis

2.9.12 IEZNcAxisMonitor::GetAuxAxisVersion — Getting the auxiliary axis version information

☐Calling procedure (Custom interface)

```
HRESULT        GetAuxAxisVersion(
                    LONG lAxisNo,              //  (I)  Axis designation
                    LONG lIndex,               //  (I)  Auxiliary axis version information
                    LPOLESTR* lppwszBuffer,          type
                    LONG* plRet                //  (O)  Version information character string
                                               //  (O)  Error code
                    )
```

☐Calling procedure (Automation interface)

```
               Monitor_ GetAuxAxisVersion (
                    lAxisNo As LONG            //  (I)  Axis designation
                    lIndex As LONG             //  (I)  Auxiliary axis version information
                    lppwszBuffer As STRING*          type
                    ) As LONG                  //  (O)  Version information character string
                                               //  (O)  Error code
```

☐Argument

lAxisNo: Set axis # (“1” or later)

lIndex: Set the type of the auxiliary axis version information. Refer to the table below.

lppwszBuffer: Gets the version information as **UNICODE** character strings

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_READ_READ: Impossible to read the data
EZNC_DATA_READ_ADDR: Part system designation or the axis designation is illegal

lIndex	Description	Data range
0	Unit model name	Character string within 9 letters
1	S/W version	Character string within 16 letters
2	Motor model name	Character string within 9 letters

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This gets the auxiliary axis’s version information.
Memory area for character strings is saved inside of EZSocket. If your client is a VC++ client, it is necessary to release the memory area explicitly by using **CoTaskMemFree()**.

☐Reference

☐Designation

Axis

2.9.13 IEZNcAxisMonitor::GetDowelTime — Getting the remaining dwell time

☐Calling procedure (Custom interface)

```
HRESULT        GetDowelTime(
                   DOUBLE* pdTime,      //  (O)  Remaining dwell time
                   LONG* plRet          //  (O)  Error code
                   )
```

☐Calling procedure (Automation interface)

```
               Monitor_GetDowelTime(
                   pdTime As DOUBLE*    //  (O)  Remaining dwell time
                   ) As LONG            //  (O)  Error code
```

☐Argument

pdTime: Returns the remaining time of dowel (G04)
Unit: second
Value: 0.000 to 99999.999 (sec)

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_READ_READ: Impossible to read the data
EZNC_DATA_READ_ADDR: Part system designation or the axis designation is illegal

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This returns the remaining time of dowel (G04). Unit: second.

☐Reference

☐Designation

System

2.10 IEZNcRunStatus Interface

Magic64	M6x5M	M6x5L	C64	C70	CNC700

2.10.1 IEZNcRunStatus::GetInvalidStatus — Getting the invalid status

☐Calling procedure (Custom interface)

```
HRESULT        GetInvalidStatus(
                    LONG* plStatus,        //  (O)  Invalid status flag
                    LONG* plRet            //  (O)  Error code
                    )
```

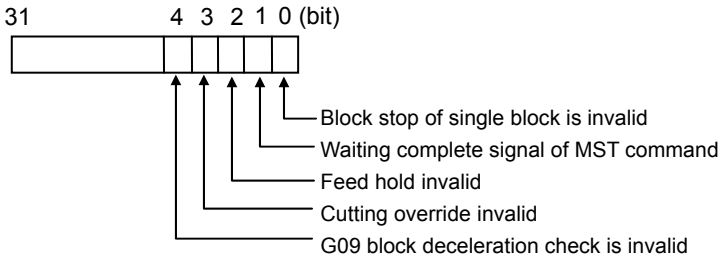
☐Calling procedure (Automation interface)

```
               Status_GetInvalidStatus(
                    plStatus As LONG*      //  (O)  Invalid status flag
                    ) As LONG              //  (O)  Error code
```

☐Argument

plStatus: Returns the bit flag of invalid status

Value	Meaning
0	OFF
1	ON



plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_READ_ADDR: Part system designation is illegal
EZNC_DATA_READ_READ: Impossible to read the data

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This returns the invalid status flag.

Invalid status:
- Block stop of single block is invalid
- Waiting complete signal of MST command
- Feed hold invalid
- Cutting override invalid
- G09 block deceleration check is invalid

(Note) Mitsubishi CNC C70 Series doesn't support the automation interface.

☐Reference

☐Designation

System

2.10.2 IEZNcRunStatus::GetCommandStatus — Getting the operation command status

☐Calling procedure (Custom interface)

```
HRESULT        GetCommandStatus(
                    LONG* plStatus,        //  (O)  Operation command status
                    LONG* plRet            //  (O)  Error code
                    )
```

☐Calling procedure (Automation interface)

```
               Status_GetCommandStatus(
                    plStatus As LONG       //  (O)  Operation command status
                    ) As LONG              //  (O)  Error code
```

☐Argument

plStatus: This returns the operation command status by the numbers below

Value	Meaning	Value	Meaning
0	Positioning (Independent axes)	**15**	The 3rd reference position check
1	Positioning (Linear)	**16**	The 4th reference position check
2	Linear interpolation	**17**	Automatic reference position return
3	Circular interpolation (CW)	**18**	Return from automatic reference position
4	Circular interpolation (CCW)	**19**	The second reference position return
5	Helical interpolation (CW)	**20**	The third reference position return
6	Helical interpolation (CCW)	**21**	The fourth reference position return
7	Reservation	**22**	Skip function
8	Reservation	**23**	Multi-step skip function 1
9	Reservation	**24**	Multi-step skip function 2
10	Reservation	**25**	Multi-step skip function 3
11	Time command dwell	**26**	Thread cutting
12	Reservation	**27**	Reservation
13	The 1st reference position check	**28**	Reservation
14	The 2nd reference position check	**29**	Setting coordinate system

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_READ_ADDR: Part system designation is illegal
EZNC_DATA_READ_READ: Impossible to read the data

☐Return value

Return Value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This gets operation command status.

(Note) Mitsubishi CNC C70 Series doesn't support the automation interface.

☐Reference

☐Designation

System

2.10.3 IEZNcRunStatus::GetCuttingMode — Getting the cutting mode

☐Calling procedure (Custom interface)

```
HRESULT        GetCuttingMode(
                   LONG* plMode,        //  (O)  Cutting mode
                   LONG* plRet          //  (O)  Error code
                   )
```

☐Calling procedure (Automation interface)

```
               Status_GetCuttingMode(
                   plMode As LONG       //  (O)  Cutting mode
                   ) As LONG            //  (O)  Error code
```

☐Argument

plMode: Returns the cutting mode

Value	Meaning
1	G01, G02, G03, G31, G33, G34, G35 mode
0	Other than the above

plRet: Returns an error code. (When using automation interface, returns a return value instead.)

S_OK: Normal termination
EZNC_DATA_READ_ADDR: Part system designation is illegal
EZNC_DATA_READ_READ: Impossible to read the data

☐Return value

Return Value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This gets the cutting mode.

(Note) Mitsubishi CNC C70 Series doesn't support the automation interface.

☐Reference

☐Designation

System

2.10.4 IEZNcRunStatus::GetAxisStatus — Getting the servo axis status

☐Calling procedure (Custom interface)

```
HRESULT        GetAxisStatus(
                    LONG lAxisNo,          //  (I)  Axis designation
                    LONG lType,            //  (I)  Status type
                    LONG* plStatus,        //  (O)  Servo axis status
                    LONG* plRet            //  (O)  Error code
                    )
```

☐Calling procedure (Automation interface)

```
               Status_GetAxisStatus(
                    lAxisNo As LONG        //  (I)  Axis designation
                    lType As LONG          //  (I)  Status type
                    plStatus As LONG*      //  (O)  Servo axis status
                    ) As LONG              //  (O)  Error code
```

☐Argument

lAxisNo: Set axis # ("1" or later) When *lType*=4, valid. When *lType* is other than 4, gets the information of all the axes for the designated part system.

lType: Set status type.

plStatus: Returns the servo axis status

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK: Normal termination
EZ_ERR_DATA_TYPE: Data type of the argument is illegal
EZNC_DATA_READ_ADDR: Part system designation is illegal
EZNC_DATA_READ_READ: Impossible to read the data

lType	Meaning	Data range
0	Completion of the 1st reference position return	The bit corresponding to the returned axis gets 1. E.g.) 00000101=The 1st and 3rd axes completed the return
1	Completion of the 2nd reference position return	The bit corresponding to the returned axis gets 1. E.g.) 00000010=The 2nd axis completed the return
2	Completion of the 3rd reference position return	The bit corresponding to the returned axis gets 1. E.g.) 00001010=The 2nd and 4th axes completed the return
3	Completion of the 4th reference position return	The bit corresponding to the returned axis gets 1. E.g.) 00001000=The 4th axes completed the return
4	Axis status (While removing the axis) Axis designation is required.	0: The axis is not being removed 1: The axis is being removed
5	Axis status (Servo OFF)	The servo OFF axis The bit corresponding to the axis gets 1.
6	Axis status (Mirror image)	Mirror image axis. The bit corresponding to the axis designated as mirror image axis gets 1.

☐Return value

Return Value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐**Functions**	This gets the servo axis status. (Note) Mitsubishi CNC C70 Series doesn't support the automation interface.
☐**Reference**	
☐**Designation**	System (System designation is required only in the case of CNC700 and IType 4, 6.), Axis

2.10.5 IEZNcRunStatus::GetRunStatus — Getting the operation status

☐Calling procedure (Custom interface)

```
HRESULT        GetRunStatus(
                    LONG lIndex,          //  (I)  Operation type
                    LONG* plStatus,       //  (O)  Operation status
                    LONG* plRet           //  (O)  Error code
                    )
```

☐Calling procedure (Automation interface)

```
               Status_GetRunStatus(
                    lIndex As LONG        //  (I)  Operation type
                    plStatus As LONG*     //  (O)  Operation status
                    ) As LONG             //  (O)  Error code
```

☐Argument

lIndex: Set status #

plStatus: Returns the status of the designated operation

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK: Normal termination
EZ_ERR_DATA_TYPE: Data type of the argument is illegal
EZNC_DATA_READ_ADDR: Part system designation is illegal
EZNC_DATA_READ_READ: Impossible to read the data

lIndex	Meaning	Data range
0	Tool length measurement	0: Not executing tool length measurement 1: Executing tool length measurement
1	Automatic operation	0: Not in automatic operation 1: In automatic operation
2	Automatic operation start-up	0: Not executing automatic operation start-up 1: Executing automatic operation start-up

☐Return value

Return Value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This gets the operation status.

(Note) Mitsubishi CNC C70 Series doesn't support the automation interface.

☐Reference

☐Designation

System

2.11 IEZNcFile5 Interface

Magic64	M6x5M	M6x5L	C64	C70	CNC700

2.11.1 IEZNcFile5::FindDir — Finding directory

☐Calling procedure (Custom interface)

```
HRESULT     FindDir(
            LPCOLESTR                   //  (I)  Directory name
            lIpcwszDirectryName,        //  (I)  Setting type and format of the data to read
            LONG lFileType,             //  (O)  File information character strings
            LPOLESTR* lppwszFileInfo,   //  (O)  Error code
            LONG* plRet
            )
```

☐Calling procedure (Automation interface)

```
            File_FindDir(
            lpcwszDirectryName As STRING //  (I)  Directory name
            lFileType As LONG            //  (I)  Setting type and format of the data to read
            lppwszFileInfo As STRING*    //  (O)  File information character strings
            )As LONG                     //  (O)  Error code
```

☐Argument

lpcwszDirectryName: Set directory name as **UNICODE** character strings
Set the directory by absolute path as follows,

Drive name +":"+\Directory name\File name --- Gets the information of the designated file name.(Note1)
Drive name +":"+\Directory name --- Gets the information of the designated directory name. (Note1)
Drive name +":"+\Directory name\ --- Gets the information under the designated directory.
(Note1) This designation is for CNC700.

lFileType: Set type and format of the data to read
It is possible to designate the following with pipes (|). When setting **NULL**, the file information is read.

Value	Meaning
EZNC_DISK_DIRTYPE	Reading directory information
EZNC_DISK_COMMENT	Reading comment information (Only the NC side)
EZNC_DISK_DATE	Reading date information (Only the PC side)
EZNC_DISK_SIZE	Reading size information

lppwszFileInfo: Gets the file information as **UNICODE** character strings
The format of file information is as below.

File name\tSize\tDate\tComment\0

Do not fail to add the **TAB** code between file name and size, size and date, date and comment.
The data have to be ended with the **NULL** code.

plRet: Returns whether any file information is provided for the read data, and the error code if provided. (When using automation interface, it returns a return value instead.)

0: No file information
More than 1: File information provided
EZNC_FILE_DIR_DATASIZE: Data size over
EZNC_FILE_DIR_NOTOPEN: Not opened
EZNC_FILE_DIR_READ: File information reading error
EZNC_FILE_DIR_ALREADYOPENED: Another directory is already open
EZNC_FILE_DIR_NODRIVE: Drive doesn't exist
EZNC_FILE_DIR_NODIR: Directory doesn't exist

(Note2) When an error occurs to the PC, the error code **EZNC_PCFILE_**... is shown instead of **EZNC_FILE_**....

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This searches directory.

FindDir() reads one file, and by calling **FindDir()** repeatedly, the file name list is got from the designated directory. The format of the file information that is saved in the area shown by *lpszFileInfo* is as follows.

File name\tSize\tDate\tComment\0

Do not fail to add the **TAB** code between file name and size, size and date, date and comment.
Data have to be ended with the **NULL** code.
The information after the file name is saved as long as the reading type is designated.
When designating "**EZNC_DISK_COMMENT|EZNC_DISK_DATE**", the format is

File name\tDate\tComment\0

In the case that the comment is not found in the file,
"**EZNC_DISK_SIZE|ESNC_DISK_COMMENT**" is invalidated. The format is

File name\tSize\t\0

Memory area for character strings is saved inside of EZSocket. If your client is a VC++ client, it is necessary to release the memory area explicitly by using **CoTaskMemFree()**.

(Note 1) With **M6x5M**, other than M01:\PRG cannot be specified. (An error occurs.)
(Note 2) Reading of the directory size information on the NC-side compact flash (**CNC700M/L**) is not corresponded. The read directory size information is invalid.

☐Restriction

If you use **FindDir()** with **MELDAS600L**, **MELDASMAGIC64**, **MELDASC6/C64**, **CNC700M/L**, or **CNC C70**, **FindDir()**, **OpenFile3()** and **OpenNCFile2()** cannot be executed until **ResetDir()** is executed.
If you try to execute it, an error "**EZNC_FILE_DIR_ALREADYOPENED (**0x80030101) Another directory is already open" will occur. If you wish to execute it, execute **ResetDir()** immediately after executing **FindDir()**.

☐Reference

FindNextDir(), ResetDir()

☐Designation

2.11.2 IEZNcFile5::FindNextDir — Finding the next directory

☐Calling procedure (Custom interface)

```
HRESULT     FindNextDir(
            LPOLESTR* lppwszFileInfo,   //  (O)  File information character strings
            LONG* plRet                 //  (O)  Error code
            )
```

☐Calling procedure (Automation interface)

```
            File_FindNextDir(
            lppwszFileInfo As STRING*   //  (O)  File information character strings
            )As LONG                    //  (O)  Error code
```

☐Argument

lpcwszFileInfo: Set file information as UNICODE character strings
The file information format is as follows.
File name\tSize\tDate\tComment\0
Do not fail to add the **TAB** code between file name and size, size and date, date and comment.
The data have to be ended with the **NULL** code.

plRet: Returns whether any file information is provided for the read data, and the error code if provided. (When using automation interface, it returns a return value instead.)
0: No file information
More than 1: File information provided
EZNC_FILE_DIR_DATASIZE: Data size over
EZNC_FILE_DIR_NOTOPEN: Not opened
EZNC_FILE_DIR_READ: File information reading error
EZNC_FILE_DIR_NODRIVE: Drive doesn't exist

(Note) When an error occurs to the PC, the error code **EZNC_PCFILE_**... is shown instead of **EZNC_FILE_**....

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This continues the directory search.
To get directory information continuously after executing **Find Dir()**, call **FindNextDir()** repeatedly. File name list in the designated directory can be got. The format of the file information that is saved in the area shown by *lpszFileInfo* is the same as **FindDir().**
Memory area for character strings is saved inside of EZSocket. If your client is a VC++ client, it is necessary to release the memory area explicitly by using **CoTaskMemFree()**.

☐Reference

FindDir(), ResetDir()

☐Designation

2.11.3 IEZNcFile5::ResetDir — Finishing the directory search

☐Calling procedure (Custom interface)

```
HRESULT     ResetDir(
            LONG* plRet                 //  (O)  Error code
            )
```

☐Calling procedure (Automation interface)

```
            File_ResetDir( ) As LONG    //  (O)  Error code
```

☐Argument

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)

EZNC_FILE_DIR_DATASIZE: Data size over
EZNC_FILE_DIR_NOTOPEN: Not opened
EZNC_FILE_DIR_READ: File information reading error

(Note) When an error occurs to the PC, the error code **EZNC_PCFILE_**... is shown instead of **EZNC_FILE_**....

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This finishes the directory search.
To search directory again, execute **FindDir()**.

☐Reference

FindDir()

☐Designation

2.11.4 IEZNcFile5::Copy2 — File Copy

☐Calling procedure (Custom interface)

```
HRESULT     Copy2(
            LPCOLESTR lpcwszSrcFileName,   //  (I)  Source file name
            LPCOLESTR lpcwszDstFileName,   //  (I)  Target file name
            LONG* plRet                    //  (O)  Error code
            )
```

☐Calling procedure (Automation interface)

```
            File_Copy2(
            lpcwszSrcFileName As STRING    //  (I)  Source file name
            lpcwszDstFileName As STRING    //  (I)  Target file name
            ) As LONG                      //  (O)  Error code
```

☐Argument

lpcwszSrcFileName: Set source file name by **UNICODE** character strings

lpcwszDstFileName: Set target file name by **UNICODE** character strings

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_FILE_COPY_BUSY: Impossible to make copies (In operation)
EZNC_FILE_COPY_ENTRYOVER: Entry number over
EZNC_FILE_COPY_FILEEXIST: Target file already exists
EZNC_FILE_COPY_FILESYSTEM: File system is abnormal
EZNC_FILE_COPY_ILLEGALNAME: File name format is illegal
EZNC_FILE_COPY_MEMORYOVER: Memory over
EZNC_FILE_COPY_NODIR: Directory doesn't exist
EZNC_FILE_COPY_NODRIVE: Drive doesn't exist
EZNC_FILE_COPY_NOFILE: File doesn't exist
EZNC_FILE_COPY_PLCRUN: Impossible to make copies (PLC is running)
EZNC_FILE_COPY_READ: Impossible to read the source file
EZNC_FILE_COPY_WRITE: Impossible to write in the target file
EZNC_FILE_COPY_PROTECT: Impossible to copy the file due to data protection (Protected)
EZNC_PCFILE_COPY_CREATE: Impossible to create files (PC)
EZNC_PCFILE_COPY_OPEN: Impossible to open files (PC)
(Note)When an error occurs to the PC, the error code **EZNC_PCFILE_**... is shown instead of **EZNC_FILE_**....

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐**Functions**	This copies the file designated by *lpcwszSrcFileName* to *lpcwszDstFileName*. The file name has to be designated by the absolute path. Drive name + " : " + \Directory name\File name In the case of M6x5M, M01:\PRG\USR\File name. (Directory other than M01:\PRG\USR\File name is invalid.) *lpcwszDustName* has to be a new name. (Impossible to use the file name that already exists.) The target directory has to exist. This method doesn't check if the designated directory and fire name are correct or not. We would recommend to check file names and directory in irregular operating, forwarding different kinds of files, or copying files to the different kinds of directory, and so on. E.g.: Overwriting the parameter files (PARAMET.BIN) onto program (\PGR\USER\(program name).PRG) **(Note)** While the NC is in automatic operation, do not execute this function. (Except for **CNC700** series)
☐**Reference**	**Delete2(), Rename2()**
☐**Designation**	

2.11.5 IEZNcFile5::Delete2 — Deleting file

☐Calling procedure (Custom interface)

```
HRESULT     Delete2(
            LPCOLESTR lpcwszFileName,   //  (I)  File name
            LONG* plRet                 //  (O)  Error code
            )
```

☐Calling procedure (Automation interface)

```
            File_Delete2(
            lpcwszFileName As STRING    //  (I)  File name
            ) As LONG                   //  (O)  Error code
```

☐Argument

lpcwszFileName: This designates the file name by **UNICODE**

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_FILE_DEL_BUSY: Impossible to delete (In operation)
EZNC_FILE_DEL_FILESYSTEM: File system is abnormal
EZNC_FILE_DEL_ILLEGALNAME: File name format is illegal
EZNC_FILE_DEL_PROTECT: Impossible to delete the file due to data protection (Protected)
EZNC_FILE_DEL_NODIR: Directory doesn't exist
EZNC_FILE_DEL_NODRIVE: Drive doesn't exist
EZNC_FILE_DEL_NOFILE: The file doesn't exist
EZNC_PCFILE_DEL_NOTDELETE: Impossible to delete the file
(Note) When an error occurs to the PC, the error code **EZNC_PCFILE_**... is shown instead of **EZNC_FILE_**....

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This deletes the file designated by *lpcwszFileName*.
The file name has to be designated by the absolute path.
Drive name + " : " + \Directory name\File name
(Note) While the NC is in automatic operation, do not execute this function. (except for **CNC700** series). Note that, however, with CNC700, it is possible as long as the target file is not under automatic operation.

☐Reference

Copy2(), Rename2()

☐Designation

2.11.6 IEZNcFile5::Rename2 — Renaming file

☐Calling procedure (Custom interface)

```
HRESULT     Rename2(
            LPCOLESTR lpcwszFileName,      //  (I)  Current file name
            LPCOLESTR lpcwszDstFileName,   //  (I)  New file name
            LONG* plRet                    //  (O)  Error code
            )
```

☐Calling procedure (Automation interface)

```
            File_Rename2(
            lpcwszSrcFileName As STRING    //  (I)  Current file name
            lpcwszDstFileName As STRING    //  (I)  New file name
            ) As LONG                      //  (O)  Error code
```

☐Argument

lpcwszSrcFileName: Set the current file name

lpcwszDstFileName: Set new file name

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_FILE_REN_BUSY: Impossible to rename (In operation)
EZNC_FILE_REN_FILEEXIST: New file name already exists
EZNC_FILE_REN_FILESYSTEM: File system is abnormal
EZNC_FILE_REN_ILLEGALNAME: File name format is illegal
EZNC_FILE_REN_PROTECT: Impossible to rename the file due to data protection (Protected)
EZNC_FILE_REN_NODIR: Directory doesn't exist
EZNC_FILE_REN_NODRIVE: Drive doesn't exist
EZNC_FILE_REN_NOFILE: The file doesn't exist
EZNC_PCFILE_REN_NOTRENAME: Impossible to rename files
EZNC_PCFILE_REN_SAMENAME: New file name is the same as the former name
(Note)When an error occurs to the PC, the error code **EZNC_PCFILE_**... is shown instead of **EZNC_FILE_**....

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This renames the file name designated by *lpcwszSrcFileName* to the one designated by *lpcwszFileName*.
The file name, *lpszSrcFileName*, has to be designated by the absolute path.
Drive name + " : " + \Directory name\File name
Designate only the file name, which does not include a drive name nor directory name, for *lpcwszDstFileName*.
Do not designate an existing file name for *lpcwszDstFileName*.
(Note) While the NC is in automatic operation, do not execute this function. (Except for CNC700)
Note that, however, with CNC700, it is possible even while the NC is in automatic operation, as long as the target file is not under automatic operation.

☐Reference

Copy2(), Delete2()

☐Designation

2.11.7 IEZNcFile5::GetDriveInformation — Getting the drive information

☐Calling procedure (Custom interface)

```
HRESULT     GetDriveInformation(
            LPOLESTR* lppwszDriveInfo,   //  (O)  Drive information character strings
            LONG* plRet                  //  (O)  Error code
            )
```

☐Calling procedure (Automation interface)

```
            File_GetDriveInformation(
            lppwszDriveInfo As STRING*   //  (O)  Drive information character strings
            ) As LONG                    //  (O)  Error code
```

☐Argument

lppwszDriveInfo: Gets the drive information as **UNICODE** character strings
The format of drive information is as below.

Drive name: **CRLF** drive name: **CRLF**... drive name: **CRLF**\0

Do not fail to add **CR, LF** code between drive names. The data has to be ended with the **CR, LF** and **NULL** code. The data has to be ended with the **NULL** code.

plRet: Returns the drive information data size or error code. (When using automation interface, it returns a return value instead.)

0: No drive
More than 1: Drive information data size. (Byte)
EZNC_FILE_DRVLIST_READ: Drive information read error
EZNC_FILE_DIR_NODRIVE: Drive doesn't exist

(Note)When an error occurs to the PC, the error code **EZNC_PCFILE_**... is shown instead of **EZNC_FILE_**....

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This reads the drive data of the currently connected NC card.
The format of drive information is as follows,

Drive name: **CRLF** Drive name: **CRLF**...Drive name: **CRLF**\0

Do not fail to add **CR** or **LF** code between drive names. The data has to be ended with the **CR**, **LF** and **NULL** code. The data has to be ended with the **NULL** code.
Memory area for character strings is saved inside of EZSocket. If your client is a VC++ client, it is necessary to release the memory area explicitly by using **CoTaskMemFree()**.
PC drive information won't be read.

☐Reference

GetDriveSize()

☐Designation

2.11.8 IEZNcFile5::GetDriveSize — Getting the drive free capacity

☐Calling procedure (Custom interface)

```
HRESULT     GetDriveSize(
            LPOLESTR lpcwszDirectryName,   //  (I)  Directory name
            LONG* plDriveSize              //  (O)  Free space
            LONG* plRet                    //  (O)  Error code
            )
```

☐Calling procedure (Automation interface)

```
            File_GetDriveSize(
            lpcwszDirectryName As STRING*  //  (I)  Directory name
            plDriveSize As LONG*           //  (O)  Free space
            ) As LONG                      //  (O)  Error code
```

☐Argument

lppwszDirectryName: Set directory name as **UNICODE** character strings
The directory has to be designated by absolute path.
Drive name + " : " + \Directory name\
In the case of M6x5M, always M01:\PRG\USR. (This argument is ignored.)

plDriveSize: Gets the free capacity size of the designated directory. (Unit: Byte)

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
EZNC_FILE_REN_FILESYSTEM: File system is abnormal
EZNC_FILE_DIR_NODRIVE: Drive doesn't exist
EZNC_FILE_DISKFREE_NODIR: Directory doesn't exist
EZNC_FILE_DISKFREE_NODRIVE: Drive doesn't exist
(Note)When an error occurs to the PC, the error code **EZNC_PCFILE_**... is shown instead of **EZNC_FILE_**....

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This returns the free capacity size of directory designated by *lpcwszDirectoryName*.
Unit to express the capacity size is byte.
The directory has to be designated by absolute path as follows,
Drive name+ " : " +\Directory name\
When designating the PC drive for drive name, the directory designation is ignored and drive capacity size is returned instead.
When designating the NC card for drive name, capacity size of the designated directory is returned. In the case that sub directory exists in the designated directory, memory size used for sub directory is not included in drive capacity size.
(Note 1) When the directory name corresponding to the NC side compact flash is specified as a directory name, the error "**EZNC_FILE_DISKFREE_NODIR**(0x80030691) Directory doesn't exist" occurs.
(Note 2) Mitsubishi CNC C70 Series doesn't support the automation interface.

☐Reference

GetDriveInformation()

☐Designation

2.11.9 IEZNcFile5::OpenFile3 — Opening file

☐Calling procedure (Custom interface)

```
HRESULT     OpenFile3(
            LPOLESTR lpcwszFileName,    //  (I)  File name with path
            LONG lMode,                 //  (I)  Open mode
            LONG* plRet                 //  (O)  Error code
            )
```

☐Calling procedure (Automation interface)

```
            File_OpenFile3(
            bstrFileName As STRING      //  (I)  File name with path
            lMode As LONG               //  (I)  Open mode
            ) As LONG                   //  (O)  Error code
```

☐Argument

lppwszFileName: Set directory name (including path) as **UNICODE** character strings
The directory has to be designated by absolute path.
Drive name + " : " + \Directory name\File name
For the lists of accessible files in the NC, refer to Table 2-2, 2-3, 2-4, 2-5, 2-6 and 2-7. All the files in the NC except for machining programs can be backed-up, but cannot be edited.

bstrFileName: Refer to the explanation of *lpcwszFileName*.
lMode: Set open mode.

Value	Meaning
EZNC_FILE_READ	Reading mode
EZNC_FILE_WRITE	Writing mode
EZNC_FILE_OVERWRITE	Overwriting mode (Overwrites the data even if the designated file already exists.)

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_FILE_OPEN_OPEN: Impossible to open the file
EZNC_FILE_OPEN_ALREADYOPENED: File already open
EZNC_FILE_OPEN_FILEEXIST: Target file already exists (In the writing mode)
EZNC_FILE_OPEN_FILENOEXIST: The file doesn’t exist (In the reading mode)
EZNC_FILE_OPEN_MODE: Open mode illegal
EZNC_FILE_OPEN_NOTOPEN: Impossible to open the file
EZNC_FILE_OPEN_CREATE: Impossible to create temporary files (In the writing mode)
EZNC_FILE_READFILE_CREATE: Impossible to create temporary files (In the reading mode)
EZNC_FILE_DIR_NODRIVE: Drive doesn't exist
EZNC_FILE_DIR_ALREADYOPENED: Another directory already open
EZNC_FILE_OPEN_ILLEGALPATH: Illegal path

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐**Functions**	This opens file in the designated mode. The directory to create the temporary file is created in the priority order below. • Directory designated by environment variable TMP • Directory in which EZSocket is installed The temporary file name is "MELDASn". A figure comes to the position of "n". **(Note1) EZNC_FILE_OVERWRITE** for *lMode* is available with **CNC 700M/L** and **CNC700**. **M6x5L**, **M6x5M**, **Magic64** and **C64** do not allow writing (overwriting) data into an existing file. They return **EZNC_FILE_OPEN_MODE** to *plRet* instead. **(Note2)** Make sure to execute **CloseFile2()** (or **AbortFile2()**) to close the file opened by thismethod. Unless **CloseFile2()** is executed, a temporary file remains. **(Note3)** While the NC is in automatic operation, do not execute writing/overwriting. (Except for CNC700) Note that, however, with CNC700, writing/overwriting is possible even while the NC is in automatic operation, as long as the file to be written/overwritten is not under automatic operation. Reading while NC is in automatic operation is possible. (Note 4) Mitsubishi CNC C70 Series doesn't support the automation interface.
☐**Reference**	**CloseFile2(), AborFile2(), ReadFile2(), WriteFile()** (Note 5) The existing interface IEZNcFile4:CloseFile() canot be used with this metod. In order to close a file opened by this method, use **CloseFile2().**
☐**Designation**	

Table2-2 List of accessible files of Magic64 (Ver. B6 and later)

File description	Directory	File name	Remarks
Machining program	M01:\PRG\USER\	Program #.PRG	Refer to the Instruction Manual of MELDASMAGIC64.
Fixed cycle program	M01:\PRG\FIX\	Program #.PRG	
MDI program	M01:\PRG\MDI\	MDI.PRG	
Parameter (User area, machine area)	M01:\PRM\	ALL.PRM	
Parameter	M01:\PRM\	PARAMET.BIN	
NC file system data	M01:\PRM\	FILESYS.BIN	
User PLC program	M01:\LAD\	USERPLC.LAD	
Workpiece offset	M01:\DAT\	WORK.OFS	
Tool offset data	M01:\DAT\	TOOL.OFS	
Common variable data	M01:\DAT\	COMMON.VAR	
C register	M01:\REG\	CREG.REG	
R register	M01:\REG\	RREG.REG	
T register	M01:\REG\	TREG.REG	

Table2-3 List of accessible files of M6x5M (Ver. E7 and later)

File description	Directory	File name	Remarks
Machining program	M01:/PRG/USR/	Program #	Refer to the Instruction Manual of MELDAS600M series.
User macro program	M01:/PRG/USR/	Program #	
Machine builder macro program	M01:/PRG/MAC/	Program #	Area setting or parameter setting necessary
Fixed cycle program	M01:/PRG/FIX/	Program #	Parameter setting necessary
MDI program	M01:/PRG/MDI/	100000000	
System parameter	M02:/PRM/	SYSTEM.PRM	Password necessary for input
Parameter (User area, machine area)	M02:/PRM/	USER.PRM	
Machine parameter	M02:/PRM/	MACHINE.PRM	For maintenance
User PLC program	M02:/MEM/	LAD.DAT	Open for writing impossible (Writing prohibited)
User PLC program (Built-in F-ROM)	M02:/FRA/	LAD.DAT	On-board Flash-ROM
User PLC program (Extended F-ROM)	M02:/FRB/	LAD.DAT	Extension Flash-ROM necessary
Tool offset data	M02:/TOL/	OFFSET.TOL	
Common variable data	M02:/COM/	COMMON.VAR	
SRAM data	M02:/MEM/	SRAM.DAT	For maintenance
Sampling data	M02:/SMP/	SAMPLE.DAT	For maintenance. Sampling setting necessary (Writing prohibited)
Sampling data (Complete round)	M02:/SMP/	SAMPLING.DAT	For maintenance. Sampling setting necessary (Writing prohibited)
Tool life data	M02:/TLF/	TOOLLIFE.TLF	
Tracking data	M02:/LOG/	TRACK.MNT	Extension RAM necessary. Reading possible after tracking (Writing prohibited)
Auxiliary axis absolute position data	M02:/AUX/	AUXABS.AUX	

Table2-4 List of accessible files of M6x5L (Ver. A2B and later)

File description	Directory	File name	Remarks
Machining program	M01:\PRG\USER\	Program #.PRG	Refer to the Instruction Manual of MELDAS600L series.
User macro program	M01:\PRG\UMACRO\	Program #.PRG	Program # is 8000 - 8999
Machine builder macro program	M01:\PRG\MMACRO\	Program #.PRG	Program # is 9000 – 9999, option
Fixed cycle program	M01:\PRG\FIX\	Program #.PRG	
MDI program	M01:\PRG\MDI\	MDI.PRG	
System parameter	M01:\PRM\	SYSCFG.BIN	System parameter input mode setting necessary
Parameter (User area, machine area)	M01:\PRM\	ALL.PRM	
Machine parameter	M01:\PRM\	PARAMET.BIN	Possible to input only to the same control unit
User PLC program	M01:\LAD\	USERPLC.LAD	Machine parameter input mode, and also PLC is not running
User PLC program (Built-in F-ROM)	M01:\LAD\	FROM-A.LAD	
User PLC program (Extended F-ROM)	M01:\LAD\	FROM-B.LAD	Extension cassette necessary
Wokpiece offset	M01:\DAT\	WORK.OFS	
Tool offset data	M01:\DAT\	TOOL.OFS	
Common variable data	M01:\DAT\	COMMON.VAR	
Custom variable data	M01:\DAT\	CUSTOM.VAR	Custom variable option necessary
SRAM data	M01:\DAT\	SRAM.BIN	System parameter is input mode, and wiring is possible only to the same control unit
Tool life data	M01:\DAT\	TOOLLIFE.TLF	Tool life management option necessary
Auxiliary axis absolute position data	M01:\DAT\	AUXABS.AUX	Only in the case auxiliary axis is connected
R register	M01:\REG\	RREG.REG	

Table2-5 List of accessible files of C64 (Ver. C0 and later)

File description	Directory	File name	Remarks
Machining program	M01:\PRG\USER\	Program #.PRG	
MDI program	M01:\PRG\MDI\	MDI.PRG	
NC file system data	M01:\PRM\	FILESYS.BIN	
MR-J2-CT parameter	M01:\PRM\	MRJ2CT.PRA	
User PLC program	M01:\LAD\	USERPLC.LAD	
Tool life data	M01:\DAT\	****.TL?	“****” is the group #, “?” is the part system #
Workpiece offset	M01:\DAT\	WORK.OFS	
Tool offset data	M01:\DAT\	TOOL.OFS	
Common variable data	M01:\DAT\	COMMON.VAR	
C register	M01:\REG\	CREG.REG	
R register	M01:\REG\	RREG.REG	
T register	M01:\REG\	TREG.REG	
Illegal process history data	M01:\LOG\	ILLEGAL.ERR	
Operation history data	M01:\LOG\	TRACE.TRC	

Table2-6 List of accessible files of CNC700 (Ver. A0 and later)

File description	Directory	File name	Remarks
Machining program	M01:\PRG\USER\	Program #	
User macro program	M01:\PRG\UMACRO\	Program #	Program # is 8000-8999
Machine builder macro program	M01:\PRG\MMACRO\	Program #	Program # is 9000-9999
Fixed cycle program	M01:\PRG\FIX\	Program #	
MDI program	M01:\PRG\MDI\	MDI.PRG	
System parameter	M01:\PRM\	SYSCFG.BIN	
Parameter (User parameter area, machine parameter area)	M01:\PRM\	ALL.PRM	
Parameter file (Binary format)	M01:\PRM\	PARAMET.BIN	
Option parameter	M01:\PRM\	SYSTEM.PRM	
User PLC program	M01:\LAD\	USERPLC.LAD	
Workpiece offset	M01:\DAT\	WORK.OFS	
Tool offset data	M01:\DAT\	TOOL.OFS	
Common variable data	M01:\DAT\	COMMON.VAR	
Custom variable data	M01:\DAT\	CUSTOM.VAR	
SRAM data (Binary format)	M01:\DAT\	SRAM.BIN	
Sampling data file (Binary format)	M01:\DAT\	SAMPLE.BIN	
MELDAS-NET data file	M01:\DAT\	TRACK.MNT	
Extended SRAM data (Binary format)	M01:\DAT\	EXTSRAM.BIN	
R register	M01:\REG\	RREG.REG	
T register	M01:\REG\	TREG.REG	
C register	M01:\REG\	CREG.REG	
History data file	M01:\LOG\	ILLEGLOGERR	
All history	M01:\LOG\	ALLLOGLOG	Key history, alarm history, PLC I/O signal history, AC input power error history
Key history	M01:\LOG\	KEYLOGLOG	
NC side compact flash	M01:\IC1\	Any arbitrary file name	NC side compact flash（Hereinafter referred to as NC side CF card）is identified as DS（data server）from NC unit and can be used for data backup or for saving large capacity program, etc.

Table2-7 List of accessible files of C70

File description	Directory	File name	Remarks
Machining program	M01:\PRG\USER\	Program #.PRG	
Fixed cycle program	M01:\PRG\FIX\	Program #.PRG	
MDI program	M01:\PRG\MDI\	MDI.PRG	
Parameter (User parameter area, machine parameter area)	M01:\PRM\	ALL.PRM	
Parameter file (Binary format)	M01:\PRM\	PARAMET.BIN	For maintenance
User PLC program	M01:\LAD\	USERPLC.LAD	
Workpiece offset	M01:\DAT\	WORK.OFS	
Tool offset data	M01:\DAT\	TOOL.OFS	
Common variable data	M01:\DAT\	COMMON.VAR	
SRAM data (Binary format)	M01:\DAT\	SRAM.BIN	For maintenance
C register	M01:\REG\	CREG.REG	
R register	M01:\REG\	RREG.REG	
T register	M01:\REG\	TREG.REG	
Sampling data	M01:\LOG\	NCSAMP.CSV	For maintenance
Operation history data	M01:\LOG\	TRACE.TRC	For maintenance

DANGER

Precaution for writing files

Before writing in the file in the NC, pay full attention to the selection of the file to write in. If you write in a wrong file, an unexpected operation may occur, which may result a serious incident.

2.11.10 IEZNcFile5::CloseFile2 — Closing file

☐Calling procedure (Custom interface)

```
HRESULT        CloseFile2(
                   LONG* plRet        //   (O)   Error code
                   )
```

☐Calling procedure (Automation interface)

```
               File_CloseFile2(       //   (O)   Error code
                   ) As LONG
```

☐Argument

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK: Normal termination
EZNC_FILE_WRITEFILE_WRITE: Impossible to write the data

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This closes file. Make sure to execute **CloseFile2()** (or **AbortFile2()**) to the file opened by **OpenFile3().**

(Note 1) While the NC is in automatic operation, do not execute this function. (Except for CNC700M/L)
Note that, however, with CNC700, it is possible even while the NC is in automatic operation, as long as the target file is not under automatic operation.
(Note 2) The existing interface IEZNcFile4::CloseFile() continues to be available for backward compatibility. When newly introducing EZSocket, use this method.
(Note 3) Mitsubishi CNC C70 Series doesn't support the automation interface.

☐Reference

OpenFile3(), AbortFile2(), ReadFile2(), WriteFile()
(Note 4) The existing interface IEZNcFile4:OpenFile2() canot be used with this metod.

☐Designation

2.11.11 IEZNcFile5::AbortFile2 — Closing file compulsorily

☐Calling procedure (Custom interface)

```
HRESULT        AbortFile2(
                   LONG* plRet        //   (O)   Error code
                   )
```

☐Calling procedure (Automation interface)

```
               File_AbortFile2(       //   (O)   Error code
                   ) As LONG
```

☐Argument

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK : Normal termination

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This closes the file opening compulsorily. Use this to abort writing. If writing is aborted, the file being created is deleted. The different point from **CloseFile2()** is that error won't be output.
(Note 1) The existing interface IEZNcFile4::AbortFile() continues to be available for backward compatibility. When newly introducing EZSocket, use this method.
(Note 2) Mitsubishi CNC C70 Series doesn't support the automation interface.

☐Reference

OpenFile3(), CloseFile2(), ReadFile2(), WriteFile()

☐Designation

2.11.12 IEZNcFile5::ReadFile2 — Reading file

☐Calling procedure (Custom interface)

```
HRESULT    ReadFile2(
           DWORD dwLength,          //  (I)  Size of the data to be read
           BYTE** ppbData,          //  (O)  Data that was read
           DWORD* pdwNumRead,       //  (O)  Size of the data that was been read
           LONG* plRet              //  (O)  Error code
           )
```

☐Calling procedure (Automation interface)

```
             File_ReadFile2(
           lLength As LONG          //  (I)  Size of the data to be read
           pvData As VARIANT*       //  (O)  Data that was read
           ) As LONG                //  (O)  Error code
```

☐Argument

dwLength: Set data size to read at one execution by the number of bytes

ppbData: Returns the pointer to the array of the byte data to have been read. The data area that was red is saved inside of **EZSocket**, so the client is required to release the memory area explicitly by using **CoTaskMemFree()**.

pdwNumRead: Returns the number of bytes that was actually read. In the automation calling, the number of bytes is included in the **VARIANT** data.

Automation argument:
lLength: Refer to the explanation of *dwLength.*
pvData: Returns the array of the byte data to have been read as **VARIANT**.

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_FILE_READFILE_NOTOPEN: File is not open in the reading mode
EZNC_FILE_READFILE_READ: Impossible to read the data
EZNC_FILE_READFILE_CREATE: Impossible to create temporary file

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

Data is read from files opened by **OpenFile3()**. The data to be read returns the array of the byte data and its number of bytes. File end is judged when *pdwNumRead* is smaller than *dwLength*.
For the size of the data to be read, set the data size to read at one execution. In reading data of large size, read can be executed by dividing the data by more than once. Until **CloseFile2()** is executed, read can be repeated.
(Note 1) The existing interface IEZNcFile4::ReadFile() continues to be available for backward compatibility. When newly introducing EZSocket, use this method.
(Note 2) Mitsubishi CNC C70 Series doesn't support the automation interface.

☐Reference

OpenFile3(), CloseFile2(), AbortFile2(), WriteFile()

☐Designation

2.11.13 IEZNcFile5::WriteFile — Writing file

☐Calling procedure (Custom interface)

```
HRESULT     WriteFile(
            DWORD dwLength,          //  (I)  Size of the data to write
            BYTE* pbData,            //  (I)  Data to write
            LONG* plRet              //  (O)  Error code
            )
```

☐Calling procedure (Automation interface)

```
            File_WriteFile(
            vData As VARIANT         //  (I)  Data to write
            ) As LONG                //  (O)  Error code
```

☐Argument

dwLength: Set data size to write at one execution by the number of bytes

pbData: Set data to write as byte array.

Automation argument:
vData: Create the data to be written in a byte alignment and substitute the data for *vData* (VARIANT) to specify as shown in the example below.

```
Example)  Dim vWriteFile As Variant
          Dim byteWrite() As Byte
          vWriteFile = byteWrite
```

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_FILE_WRITEFILE_NOTOPEN: File is not open in the writing mode
EZNC_FILE_ WRITE FILE_ WRITE: Impossible to write the data

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

Data is written in files opened by **OpenFile3()**. The data to be written is byte array data.
By the size of the data to write, set the data size to write at one execution. In writing data of large size, write can be executed by dividing the data by more than once. Until **CloseFile2()** is executed, write can be repeated.
(Note1) If you change files (except for machining programs) in the NC, the NC may not work normally. Backup the data in the case you have to recover the data.
(Note 2) Mitsubishi CNC C70 Series doesn't support the automation interface.

☐Reference

OpenFile3(), CloseFile2(), AbortFile2(), ReadFile2()

☐Designation

DANGER

Precaution for writing files

If wrong files are written in CNC device, unexpected operation may occur. As it can be a cause of a serious incident, pay full attention in writing.

2.11.14 IEZNcFile5::OpenNCFile2 Opening machining program

☐Calling procedure (Custom interface)

```
HRESULT        OpenNCFile2(
               LPCOLESTR lpcwszFileName,   //  (I)  File name with path
               LONG lMode,                 //  (I)  Open mode
               LONG* plRet                 //  (O)  Error code
               )
```

☐Calling procedure (Automation interface)

```
               File_ OpenNCFile2 (
               bstrFileName As STRING      //  (I)  File name with path
               lMode As LONG               //  (I)  Open mode
               )As LONG                    //  (O)  Error code
```

☐Argument

lpcwszFileName: Set file name (including path) as **UNICODE** character strings
Set the directory by absolute path as follows,
Drive name + " : " + \Directory name \File name
Paths other than below cannot be used.

Model	Machining program
M6x5M	M01:\PRG\USR\Machining program # M01:\PRG\MAC\ Machining program # M01:\PRG\FIX\ Machining program # M01:\PRG\MDI\ Machining program #
M6x5L	M01:\PRG\USER\ Machining program #.prg M01:\PRG\UMACRO\ Machining program #.prg M01:\PRG\MMACRO\ Machining program #.prg M01:\PRG\FIX\ Machining program #.prg M01:\PRG\MDI\ Machining program #.prg
Magic64	M01:\PRG\USER\ Machining program #.prg M01:\PRG\FIX\ Machining program #.prg M01:\PRG\MDI\MDI.prg
MELDASC6/C64	M01:\PRG\USER\Machining program #.prg M01:\PRG\MDI\MDI.prg
CNC700	M01:\PRG\USER\Machining program # M01:\PRG\UMACRO\Machining program # M01:\PRG\MMACRO\Machining program # M01:\PRG\FIX\Machining program # M01:\PRG\MDI\Machining program #

bstrFileName: Refer to the explanation of *lpcwszFileName*.

lMode: Set open mode

Value	Meaning
EZNC_FILE_READ	Reading mode
EZNC_FILE_WRITE	Writing mode
EZNC_FILE_OVERWRITE	Overwriting mode (Writing is executed even if the designated file already exists.)

plRet: Returns the data size of file information or an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_FILE_OPEN_ALREADYOPENED: File already open
EZNC_FILE_OPEN_FILEEXIST: Target file already exists (In the writing mode)
EZNC_FILE_OPEN_MODE: Opening mode illegal
EZNC_FILE_OPEN_NOTOPEN: Impossible to open the file
EZNC_FILE_OPEN_CREATE: Impossible to create the file (In the writing mode)
EZNC_FILE_OPEN_ILLEGALPATH: Illegal path
EZNC_FILE_OPEN_FILENOTEXIST: File doesn't exist
EZNC_FILE_OPEN_OPEN: Impossible to open the file
EZNC_FILE_DIR_ALREADYOPENED: Another directory already open
EZ_ERR_NOT_SUPPORT: Not supported

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This opens machining program files in the designated mode. The directory to create the temporary file is created in the priority order below.

- Directory designated by environment variable TMP
- Directory in which EZSocket is installed

The temporary file name is "MELDASn". A figure comes to the position of "n".

Not available with **OpenFile3()** at the same time.

(Note1) M6x5, **MELDASMAGIC64** and **MELDASC6/C64** delete the designated file when **OpenNCFile2()** is executed in overwriting mode.

(Note2) While the NC is in automatic operation, do not execute writing/overwriting. (Except for CNC700)
Note that, however, with CNC700, writing/overwriting is possible even while the NC is in automatic operation, as long as the file to be written/overwritten is not under automatic operation. Reading while NC is in automatic operation is possible.

(Note 3) The existing interface IEZNcFile4::OpenNCFile() continues to be available for backward compatibility. When newly introducing EZSocket, use this method.

☐Reference

CloseNCFile2(), AbortNCFile2(), ReadNCFile2(), WriteNCFile()

☐Designation

2.11.15 IEZNcFile5::CloseNCFile2 Closing machining program

☐Calling procedure (Custom interface)

```
HRESULT        CloseNCFile2(
                   LONG* plRet        //   (O)   Error code
                   )
```

☐Calling procedure (Automation interface)

```
               File_CloseNCFile2(     //   (O)   Error code
                   ) As LONG
```

☐Argument

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK: Normal termination
EZNC_FILE_WRITEFILE_WRITE: Impossible to write the data

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This closes machining file.

(Note1) While the NC is in automatic operation, do not execute this function. (Except for CNC700)
Note that, however, with CNC700, it is possible even while the NC is in automatic operation, as long as the target file is not under automatic operation.

(Note2) Make sure to execute **CloseNCFile2()** (or **AbortNCFile2()**) to close the file opened by **OpenNCFile2()**.

☐Reference

OpenNCFile2(), AbortNCFile2(), ReadNCFile2(), WriteNCFile()

☐Designation

2.11.16 IEZNcFile5::AbortNCFile2 Closing machining program compulsorily

☐Calling procedure (Custom interface)

```
HRESULT        AbortNCFile2(
                   LONG* plRet        //   (O)   Error code
                   )
```

☐Calling procedure (Automation interface)

```
               File_AbortNCFile2(     //   (O)   Error code
                   ) As LONG
```

☐Argument

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK : Normal termination

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This closes the file opening compulsorily. Use this to abort writing. If writing is aborted, the file being created is deleted. The different point from **CloseNCFile2()** is that error won't be output.

☐Reference

OpenNCFile2(), CloseNCFile2(), ReadNCFile2(), WriteNCFile()

☐Designation

2.11.17 IEZNcFile5::ReadNCFile2 — Reading machining program

☐Calling procedure (Custom interface)

```
HRESULT     ReadNCFile2(
            DWORD dwLength,          //  (I)  Size of the data to be read
            BYTE** ppbData,          //  (O)  Data that was read
            DWORD* pdwNumRead,       //  (O)  Size of the data that was read
            LONG* plRet              //  (O)  Error code
            )
```

☐Calling procedure (Automation interface)

```
            File_ReadNCFile2
            lLength As LONG          //  (I)  Size of the data to be read
            pvData As VARIANT*       //  (O)  Data that was read
            ) As LONG                //  (O)  Error code
```

☐Argument

dwLength: Set data size to read at one execution by the number of bytes

ppbData: Returns the pointer to the array of the byte data to have been read. The data area that was red is saved inside of **EZSocket**, so the client is required to release the memory area explicitly by using **CoTaskMemFree()**.

pdwNumRead: Returns the number of bytes that was actually read. In the automation calling, the number of bytes is include in the **VARIANT** data.

Automation argument:
lLength: Refer to the explanation of *dwLength.*
pvData: Returns the array of the byte data to have been read as **VARIANT**.

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_FILE_READFILE_NOTOPEN: File is not open in the reading mode
EZNC_FILE_READFILE_READ: Impossible to read the data
EZNC_FILE_READFILE_CREATE: Impossible to create temporary file

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

Data is read from machining program files opened by **OpenNCFile2()**. The data to be read returns the array of the byte data and its number of bytes. File end is judged when *pdwNumRead* is smaller than *dwLength*.
For the size of the data to be read, set the data size to read at one execution. In reading data of large size, read can be executed by dividing the data by more than once. Until **CloseNCFile2()** is executed, read can be repeated.

☐Reference

OpenNCFile2(), CloseNCFile2(), AbortNCFile2(), WriteNCFile()

☐Designation

2.11.18 IEZNcFile5::WriteNCFile — Writing machining program

☐Calling procedure (Custom interface)

```
HRESULT     WriteFile(
            DWORD dwLength,          //  (I)  Size of the data to write
            BYTE* pbData,            //  (I)  Data to write
            LONG* plRet              //  (O)  Error code
            )
```

☐Calling procedure (Automation interface)

```
            File_WriteNCFile(
            vData As VARIANT         //  (I)  Data to write
            ) As LONG                //  (O)  Error code
```

☐Argument

dwLength: Set data size to write at one execution by the number of bytes

pbData: Set data to write as byte array. In the automation calling, the number of bytes is included in *vData*.

Automation argument:
vData: Set the data to write in a byte alignment and substitute the data for *vData* (VARIANT) to specify.

Example)
```
Dim vWriteFile As Variant
Dim byteWrite() As Byte
vWriteFile = byteWrite
```

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_FILE_WRITEFILE_NOTOPEN: File is not open in the writing mode
EZNC_FILE_ WRITE FILE_ WRITE: Impossible to write the data

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

Data is written in machining program files opened by **OpenNCFile2()**. The data to be written is byte array data.
For the size of the data to write, set the data size to write at one execution. In writing data of large size, write can be executed by dividing the data by more than once. Until **CloseNCFile2()** is executed, write can be repeated.
(Note) With **CNC700**, when the parameter "#8105 Edit lock B" is "1", program 8000 to 9999 cannot be written. When the parameter "#1121 Edit lock C" is "1", program 9000 to 9999 cannot be written.

☐Reference **OpenNCFile2(), CloseNCFile2(), AbortNCFile2(), ReadNCFile2()**

☐Designation

DANGER

Precaution for writing files

If wrong files are written in CNC device, unexpected operation may occur. As it can be a cause of a serious incident, pay full attention in writing.

2.12 IEZNcCommonVariable2 Interface

Magic64	M6x5M	M6x5L	C64	C70	CNC700

2.12.1 IEZNcCommonVariable2::CommonVRead — Reading common variable

☐Calling procedure (Custom interface)

```
HRESULT        CommonVRead(
                 LONG lIndex,          //  (I)  Variable #
                 DOUBLE* pdData,       //  (O)  Variable value
                 LONG* plType          //  (O)  Type
                 LONG* plRet           //  (O)  Error code
                 )
```

☐Calling procedure (Automation interface)

```
                 CommonVariable_Read2(
                 lIndex As LONG        //  (I)  Variable #
                 pdData As DOUBLE*     //  (O)  Variable value
                 plType As LONG*       //  (O)  Type
                 ) As LONG             //  (O)  Error code
```

☐Argument

lIndex: Set common variable #s to read
Value: 100 to 199, 500 to 999
Value range depends on the model and option specifications.

pdData: Returns the common variable value from the designated common variable #

plType: Return type of variable value. (Available with **M6x5M** or **CNC700**.)

Value	Meaning
1	Numerical figure
0	No setting

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_READ_ADDR: Part system designation illegal
EZNC_DATA_READ_READ: Impossible to read information

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This reads common variable. The common variable number that is possible to treat here is limited depending on its specified number of sets. When treating the common variable #100 to #199, it is necessary to designate part system.

(Note) Mitsubishi CNC C70 Series doesn't support the automation interface.

☐Reference

CommonVWrite(), GetSize()

☐Designation

(System Common variables only #100 to #199)

2.12.2 IEZNcCommonVariable2::CommonVWrite — Writing common variable

☐Calling procedure (Custom interface)

```
HRESULT        CommonVWrite(
                LONG lIndex,          //  (I)  Variable #
                DOUBLE pdData,        //  (I)  Variable value
                LONG lType            //  (I)  Type
                LONG* plRet           //  (O)  Error code
                )
```

☐Calling procedure (Automation interface)

```
                CommonVariable_Write2(
                lIndex As LONG        //  (I)  Variable #
                dData As DOUBLE       //  (I)  Variable value
                lType As LONG         //  (I)  Type
                ) As LONG             //  (O)  Error code
```

☐Argument

lIndex: This designates the common variable to write
Value: 100 to 199, 500 to 999
Value range depends on the model and option specifications.

dData: Set common variable value to write in the designated common variable #

lType: Set type of variable value. (Available with **M6x5M** or **CNC700**.)

Value	Meaning
1	Numerical figure
0	No setting

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination.
EZNC_DATA_WRITE_ADDR: Part system designation illegal
EZNC_DATA_WRITE_WRITE: Impossible to write the data

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Function

This writes common variable. The common variable number that is possible to treat here is limited depending on its specified number of sets. When treating the common variable #100 to #199, it is necessary to designate part system.

(Note) Mitsubishi CNC C70 Series doesn't support the automation interface.

☐Reference

CommonVRead(), GetSize()

☐Designation

(System Common variables only #100 to #199)

2.12.3 IEZNcCommonVariable2::GetSize — Getting the number of common variable sets

☐Calling procedure (Custom interface)

```
HRESULT        Get Size(
                LONG lType,                  //  (I)  Common variable type
                LONG* plData,                //  (O)  The number of sets
                LONG* plRet                  //  (O)  Error code
                )
```

☐Calling procedure (Automation interface)

```
                 CommonVariable_GetSize(
                lType As LONG                //  (I)  Common variable type
                plData As LONG*              //  (O)  The number of sets
                ) As LONG                    //  (O)  Error code
```

☐Argument

lType: Set common variable type to read

Value	Meaning
0	To get the number of common variable sets after #100
1	To get the number of common variable sets after #500

pdData: Returns the number of common variable sets
E.g.: 40 = 40 sets

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_READ_ADDR: System designation illegal
EZNC_DATA_READ_READ: Impossible to read the data

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Function

This reads the number of common variable sets. When treating the common variable #100 to #199, it is necessary to designate part system.
In the case of **M6x5M**, regardless which type you set for *ltype*, the returned value will be the sum of #100 and #500.

(Note) Mitsubishi CNC C70 Series doesn't support the automation interface.

☐Reference

CommonVRead(), CommonVWrite()

☐Designation

(System Common variable #100 to #199)

2.12.4 IEZNcCommonVariable2::GetName — Getting common variable name

☐Calling procedure (Custom interface)

```
HRESULT        GetName(
                 LONG lIndex,                  //  (I)   Common variable #
                 LPOLESTR* lppwszName,         //  (O)   Common variable name character strings
                 LONG* plRet                   //  (O)   Error code
                 )
```

☐Calling procedure (Automation interface)

```
                 CommonVariable_GetName(
                 lIndex As LONG                //  (I)   Common variable #
                 lppwszName As STRING*         //  (O)   Common variable name character strings
                 ) As LONG                     //  (O)   Error code
```

☐Argument

lIndex: Set the common variable #s to read
Value: 500 to 519

lppwszName: Returns the common variable name as **UNICODE** character string
Names have to consist of 7 letters starting with an alphabet and ending with the **NULL** code.

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_READ_READ: Impossible to read information
EZNC_DATA_READ_DATASIZE: Data size over

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Function

This reads name of the common variable. Names have to consist of 7 letters starting with an alphabet and ending with the **NULL** code.
Memory area for character strings is saved inside of EZSocket. If your client is a VC++ client, it is necessary to release the memory area explicitly by using **CoTaskMemFree()**.

(Note) Mitsubishi CNC C70 Series doesn't support the automation interface.

☐Reference

CommonVRead(), CommonVWrite()

☐Designation

2.12.5 IEZNcCommonVariable2::SetName — Setting common variable name

☐Calling procedure (Custom interface)

```
HRESULT        SetName(
                 LONG lIndex,                  //  (I)   Common variable #
                 LPCOLESTR lpcwszName,         //  (I)   Common variable name character strings
                 LONG* plRet                   //  (O)   Error code
                 )
```

☐Calling procedure (Automation interface)

```
                 CommonVariable_SetName(
                 lIndex As LONG                //  (I)   Common variable #
                 lpcwszName As SRTING*         //  (I)   Common variable name character strings
                 ) As LONG                     //  (O)   Error code
```

☐Argument

lIndex: Set the common variable #s to write
Value: 500 to 519

lpcwszName: Returns the common variable name as **UNICODE** character string
Names have to consist of 7 letters starting with an alphabet and ending with the **NULL** code.

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_WRITE_WRITE: Impossible to write the data

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Function

This writes name of the common variable. Names have to consist of 7 letters starting with an alphabet and ending with the **NULL** code.

(Note) Mitsubishi CNC C70 Series doesn't support the automation interface.

☐Reference

GetName()

☐Designation

Magic64	M6x5L	C64	C70	CNC700

2.12.6 IEZNcCommonVariable2::GetCVNullData — Getting common variable null data

☐Calling procedure (Custom interface)

```
HRESULT        GetCVNullData(
                 DOUBLE* pdData,        //  (O)  Common variable null data
                 LONG* plRet            //  (O)  Error code
                 )
```

☐Calling procedure (Automation interface)

```
               CommonVariable_GetNullData(
                 pdData As DOUBLE*      //  (O)  Common variable null data
                 ) As LONG              //  (O)  Error code
```

☐Argument

pdData: Returns the null data value

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_READ_READ: Impossible to read the data

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Function

This gets the null data of the common variable (#100 to #199 and #500 to 519).

☐Reference

☐Designation

2.13 IEZNcLocalVariable2 Interface

Magic64	M6x5M	M6x5L	C64	C70	CNC700

2.13.1 IEZNcLocalVariable2::LocalVRead — Reading the local variable

☐Calling procedure (Custom interface)

```
HRESULT        LocalVRead(
                 LONG lIndex,          //  (I)  Variable #
                 LONG lLevel,          //  (I)  Level
                 DOUBLE* pdData,       //  (O)  Variable value
                 LONG* plType          //  (O)  Type
                 LONG* plRet           //  (O)  Error code
                 )
```

☐Calling procedure (Automation interface)

```
                 LocalVariable_Read2 (
                 lIndex As LONG        //  (I)  Variable #
                 lLevel As LONG        //  (I)  Level
                 pdData As DOUBLE*     //  (O)  Variable value
                 plType  As LONG*      //  (O)  Type
                 )As LONG              //  (O)  Error code
```

☐Argument

lIndex: Set local variable # to read
Value: 1 to 33

lLevel: Set macro sub program execution level
Value: 0 to 4

pdData: Returns the local variable value from the designated local variable #

plType: Return type of variable value. (Available only with M6x5M.)

Value	Meaning
1	Numerical figure
0	No setting

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination.
EZNC_DATA_READ_ADDR: Part system designation is illegal
EZNC_DATA_READ_READ: Impossible to read the data

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This reads the local variable value in the designated part system.

(Note) Mitsubishi CNC C70 Series doesn't support the automation interface.

☐Reference

GetMacroLevel()

☐Designation

System

2.13.2 IEZNcLocalVariable2::GetMacroLevel — Getting the macro sub program calling level

☐Calling procedure (Custom interface)

```
HRESULT        GetMacroLevel(
                LONG* plData,          //  (O)  Level
                LONG* plRet            //  (O)  Error code
                )
```

☐Calling procedure (Automation interface)

```
                LocalVariable_GetMacroLevel (
                plData As LONG         //  (O)  Level
                )As LONG               //  (O)  Error code
```

☐Argument

plData: Returns the macro sub program call data
Value: 0 to 3

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_READ_ADDR: Part system designation is illegal
EZNC_DATA_READ_READ: Impossible to read the data

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This gets the macro sub program call level.

(Note) Mitsubishi CNC C70 Series doesn't support the automation interface.

☐Reference

☐Designation

System

Magic64	M6x5L	C64	C70	CNC700

2.13.3 IEZNcLocalVariable2::GetLVNullData — Getting the local variable null data

☐Calling procedure (Custom interface)

```
HRESULT        GetLVNullData(
                DOUBLE* pdData,          //  (O)  Null data value
                LONG* plRet              //  (O)  Error code
                )
```

☐Calling procedure (Automation interface)

```
                 LocalVariable_GetNullData (
                plData As DOUBLE*        //  (O)  Null data value
                )As LONG                 //  (O)  Error code
```

☐Argument

pdData: Returns the null data value

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_READ_READ: Impossible to read the data

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This gets the null data value of the variables (#1 to #33).

☐Reference

☐Designation

2.14 IEZNcTool3 Interface

Magic64	M6x5M	M6x5L	C64	CNC700

2.14.1 IEZNcTool3::GetToolSetSize — Getting the number of tool offset sets

☐Calling procedure (Custom interface)

```
HRESULT        GetToolSetSize(
                LONG* plSize,          //  (O)  The number of sets
                LONG* plRet            //  (O)  Error code
                )
```

☐Calling procedure (Automation interface)

```
                Tool_GetToolSetSize (
                plSize As LONG*        //  (O)  The number of sets
                )As LONG               //  (O)  Error code
```

☐Argument

plSize: Returns the number of tool offset memory sets of the designated part system.
The number of sets depends on the NC specification.
E.g.: 200=200 sets

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_READ_READ: Impossible to read the data

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This gets the number of tool offset sets of the designated part system. The number of sets depends on the NC specification.

☐Reference

GetType()

☐Designation

System

2.14.2 IEZNcTool3::GetType — Getting the tool offset type

☐Calling procedure (Custom interface)

```
HRESULT     GetType(
             LONG* plType,          //  (O)  Type
             LONG* plRet            //  (O)  Error code
             )
```

☐Calling procedure (Automation interface)

```
             Tool_GetType (
            plType As LONG*         //  (O)  Type
            )As LONG                //  (O)  Error code
```

☐Argument

plType: Returns the tool offset type of the designated part system

Value	Meaning	
1	M system type-I	: 1-axis compensation amount
4	M system type-II	: 1-axis compensation amount with wear compensation amount
6	L system type	: 2-axis compensation amount

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_READ_READ: Impossible to read the data

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This gets the tool offset type of the designated part system.

☐Reference

GetToolSetSize()

☐Designation

System

2.14.3 IEZNcTool3::GetOffset — Getting the tool offset value

☐Calling procedure (Custom interface)

```
HRESULT        GetOffset(
                 LONG lType,                  //  (I)  Tool offset type
                 LONG lKind,                  //  (I)  Kinds of tool offset values
                 LONG lToolSetNo,             //  (I)  Tool set #
                 DOUBLE* pdOffset,            //  (O)  Offset value
                 LONG* plNo,                  //  (O)  Hypothetical tooltip #
                 LONG* plRet                  //  (O)  Error code
                 )
```

☐Calling procedure (Automation interface)

```
                 Tool_GetOffset (
                 lType As LONG                //  (I)  Tool offset type
                 lKind As LONG                //  (I)  Kinds of tool offset values
                 lToolSetNo As LONG           //  (I)  Tool set #
                 pdOffset As DOUBLE*          //  (O)  Offset value
                 plNo As LONG*                //  (O)  Hypothetical tooltip #
                 )As LONG                     //  (O)  Error code
```

☐Argument

lType: Set tool offset type. Refer to the parameter table.

lKind: Set kinds of tool offset values. Refer to the parameter table.

lToolSetNo: Set tool set #
The number of sets can be got by **GetToolSetSize()**.

pdOffset: Returns the tool offset value. Refer to the parameter table.

plNo: Returns the Hypothetical tooltip #. Refer to the parameter table.
(Note) L-type system only. In the other systems , nothing will be returned.

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_READ_READ: Impossible to read the data
EZNC_DATA_READ_DATATYPE: Data type illegal

☐**Argument**

Parameter table

Value: Type	Value: Kinds of tool offset values		Data range
1: M system type-I	0: Tool offset value		-99999.999 to 99999.999[mm]
4: M system type-II	0: Tool length offset value (Dimension) 1: Ditto (Wear offset value) 2: Tool diameter offset value (Dimension) 3: Ditto (Wear offset value)		-99999.999 to 99999.999[mm]
6: L system type	0: Tooltip wear amount 1: Ditto 2: Ditto	X Z C (Y*)	**Magic64**: -999.999 to 99.999[mm] **M6x5**,**C64**: -99.999 to 99.999[mm] **CNC700**: -99999.999 to 99999.999[mm]
	3: Tool length 4: Ditto 5: Ditto	X Z C (Y*)	**Magic64**: -99.999 to 99.999[mm] **C64**: -999.999 to 999.999[mm] **M6x5**,**CNC700**: -99999.999 to 99999.999[mm]
	6: Tooltip radius	R	**Magic64**,**C64**: 0 to 99.999[mm] **M6x5**: 0 to 999.999[mm] **CNC700**: 0 to 99999.999[mm]
	7: Tooltip radius wear amount r		**Magic64**,**M6x5**,**C64**: 0 to 99.999[mm] **CNC700**: 0 to 99999.999[mm]
	8: Hypothetical tooltip #	P	0 to 8 (Refer to figure 1)

*In the case with **M6x5**, **CNC700**

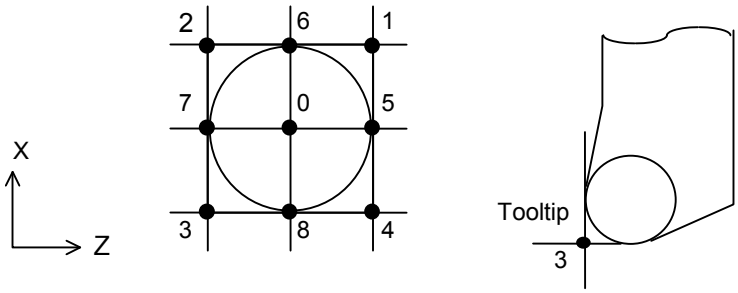


Figure 1 Hypothetical tooltip #

☐**Return value**

Return Value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐**Functions**

This gets the tool offset value of the designated part system and axis.The range indicated in the parameter list varies depending on the NC system command increment, such as inch and mm. For details, refer to the Instruction Manual of each NC system.

☐**Reference**

GetType(), SetOffset(), GetToolSetSize()

☐**Designation**

System

2.14.4 IEZNcTool3::SetOffset — Setting tool offset value

☐Calling procedure (Custom interface)

```
HRESULT        SetOffset(
                 LONG lType,          //  (I)  Tool offset type
                 LONG lKind,          //  (I)  Kinds of tool offset values
                 LONG lToolSetNo,     //  (I)  Tool set #
                 DOUBLE dOffset,      //  (I)  Offset value
                 LONG lNo,            //  (I)  Hypothetical tooltip #
                 LONG* plRet          //  (O)  Error code
                 )
```

☐Calling procedure (Automation interface)

```
               Tool_SetOffset (
                 lType As LONG        //  (I)  Tool offset type
                 lKind As LONG        //  (I)  Kinds of tool offset values
                 lToolSetNo As LONG   //  (I)  Tool set #
                 dOffset As DOUBLE    //  (I)  Offset value
                 lNo As LONG          //  (I)  Hypothetical tooltip #
                 )As LONG             //  (O)  Error code
```

☐Argument

lType: Set tool offset type. Refer to the parameter table.

lKind: Set kinds of tool offset values. Refer to the parameter table.

lToolSetNo: Set tool set #
The number of sets can be got by **GetToolSetSize()**.

dOffset: Set tool offset value. Refer to the parameter table.

lNo: Set Hypothetical tooltip #. Refer to parameter number.
(Note) L- system type only. Invalid with the other system types.

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_WRITE_WRITE: Impossible to write the data
EZNC_DATA_WRITE_DATATYPE: Data type illegal

☐**Argument** Parameter table

Value: Type	Value: Kinds of tool offset values		Data range
1: M system type-I	0: Tool offset value		-99999.999 to 99999.999 [mm]
4: M system type-II	0: Tool length offset value (Dimension) 1: Ditto (Wear offset value) 2: Tool diameter offset value (Dimension) 3: Ditto (Wear offset value)		-99999.999 to 99999.999 [mm]
6: L system type	0: Tooltip wear amount 1: Ditto 2: Ditto	X Z C (Y*)	**Magic64**: -999.999 to 99.999[mm] **M6x5**,**C64**: -99.999 to 99.999[mm] **CNC700**: -99999.999 to 99999.999[mm]
	3: Tool length 4: Ditto 5: Ditto	X Z C (Y*)	**Magic64**: -99.999 to 99.999[mm] **C64**:-999.999 to 999.999[mm] **M6x5**,**CNC700**: -99999.999 to 99999.999[mm]
	6: Tooltip radius R		**Magic64**,**C64**: 0 to 99.999[mm] **M6x5**: 0 to 999.999[mm] **CNC700**: 0 to 99999.999[mm]
	7: Tooltip radius wear amount r		**Magic64**,**M6x5**,**C64**: 0 to 99.999[mm] **CNC700**: 0 to 99999.999[mm]
	8: Hypothetical tooltip #	P	0 to 8 (Refer to Figure 1)

*In the case with **M6x5**, **CNC700**

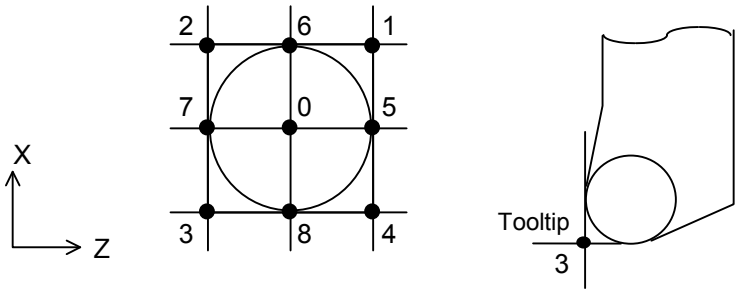


Figure 1 Hypothetical tooltip #

☐**Return value**

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐**Functions** This sets tool offset value of the designated part system and axis.The range indicated in the parameter list varies depending on the NC system command increment, such as inch and mm. For details, refer to the Instruction Manual of each NC system.

☐**Reference** **GetType(), GetOffset(), GetToolSetSize()**

☐**Designation** System

2.14.5 IEZNcTooL3::GetToolWorkOffset — Getting the workpiece coordinate offset

☐Calling procedure (Custom interface)

```
HRESULT        GetToolWorkOffset(
               LONG lAxisNo,          //  (I)  Axis designation
               LONG lIndex,           //  (I)  Workpiece coordinate system #
               DOUBLE* pdOffset,      //  (O)  Offset value
               LONG* plRet            //  (O)  Error code
               )
```

☐Calling procedure (Automation interface)

```
               Tool_ GetToolWorkOffset(
               lAxisNo As LONG        //  (I)  Axis designation
               lIndex As LONG         //  (I)  Workpiece coordinate system #
               pdOffset As DOUBLE*    //  (O)  Offset value
               )As LONG               //  (O)  Error code
```

☐Argument

lAxisNo: Set axis # (“1” or later)

lIndex: Set the workpiece coordinate system # to read

Value	Meaning
54	G54 offset
55	G55 offset
56	G56 offset
57	G57 offset
58	G58 offset
59	G59 offset
60	EXT offset

pdOffset: Returns the workpiece coordinate offset value of the designated part system and axis. Value: -99999.999 to 99999.999 [mm]

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_READ_ADDR: Part system designation is illegal
EZNC_DATA_READ_READ: Impossible to read the data

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This gets the workpiece coordinate offset value of the designated part system and axis. Refer to the operation manual for details.

☐Reference

☐Designation

System, Axis

2.14.6 IEZNcTooL3::SetToolWorkOffset — Setting workpiece coordinate offset

☐Calling procedure (Custom interface)

```
HRESULT     SetToolWorkOffset(
            LONG lAxisNo,          //  (I)  Axis designation
            LONG lIndex,           //  (I)  Workpiece coordinate system #
            DOUBLE dOffset,        //  (I)  Offset value
            LONG lMode,            //  (I)  Mode
            LONG* plRet            //  (O)  Error code
            )
```

☐Calling procedure (Automation interface)

```
            Tool_SetToolWorkOffset(
            lAxisNo As LONG        //  (I)  Axis designation
            lIndex As LONG         //  (I)  Workpiece coordinate system #
            dOffset As DOUBLE      //  (I)  Offset value
            lMode As LONG          //  (I)  Mode
            )As LONG               //  (O)  Error code
```

☐Argument

lAxisNo: Set axis # (“1” or later)

lIndex: Set the workpiece coordinate system # to read

Value	Meaning
54	G54 offset
55	G55 offset
56	G56 offset
57	G57 offset
58	G58 offset
59	G59 offset
60	EXT offset

dOffset: Set the workpiece coordinate offset value of the designated part system and axis.
Value: -99999.999 to 99999.999 [mm]

lMode: Set setting mode (absolute value/incremental value)

Value	Meaning
0	Absolute value setting
1	Incremental value setting

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_WRITE_ADDR: Part system designation is illegal
EZNC_DATA_WRITE_WRITE: Impossible to write the data

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐**Functions**	This sets the workpiece coordinate offset value of the designated part system and axis. Refer to the operation manual for details. As for setting mode: (1) In the case of the absolute value setting The designated offset value is set as current offset value. (2) In the case of the incremental value setting The designated offset value is added to the current offset before being set.
☐**Reference**	
☐**Designation**	System, Axis

2.14.7 IEZNcTool3::GetSurface — Getting the reference surface level

☐Calling procedure (Custom interface)

```
HRESULT     GetSurface(
            LONG lAxisNo,              //  (I)  Axis designation
            DOUBLE* pdHight,           //  (O)  Reference surface level
            LONG* plRet                //  (O)  Error code
            )
```

☐Calling procedure (Automation interface)

```
            Tool_GetSurface (
            lAxisNo As LONG            //  (I)  Axis designation
            pdHight As DOUBLE*         //  (O)  Reference surface level
            )As LONG                   //  (O)  Error code
```

☐Argument

lAxisNo: Set axis # (“1” or later)

pdHight: Returns the reference surface coordinate position for the tool length measurement II of the designated part system and axis.
Value: -99999.999 to 99999.999 [mm]

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_READ_READ: Impossible to read the data
EZNC_DATA_READ_ADDR: Part system designation or the axis designation is illegal

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This gets the reference surface coordinate position for the tool length measurement II of the designated part system and axis.

☐Reference

☐Designation

System, Axis

2.14.8 IEZNcTool3::SetSurface — Setting reference surface level

☐Calling procedure (Custom interface)

```
HRESULT     SetSurface(
            LONG lAxisNo,          //  (I)  Axis designation
            DOUBLE dHight,         //  (I)  Reference surface level
            LONG* plRet            //  (O)  Error code
            )
```

☐Calling procedure (Automation interface)

```
              Tool_SetSurface (
            lAxisNo As LONG        //  (I)  Axis designation
            dHight As DOUBLE       //  (I)  Reference surface level
            )As LONG               //  (O)  Error code
```

☐Argument

lAxisNo: Set axis # (“1” or later)

dHight: Set reference surface coordinate position for the tool length measurement II of the designated part system and axis.
Value: -99999.999 to 99999.999 [mm]

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_WRITE_WRITE: Impossible to write the data
EZNC_DATA_READ_ADDR: Part system designation or the axis designation is illegal

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This sets reference surface coordinate position for the tool length measurement II of the designated part system and axis.

☐Reference

☐Designation

System, Axis

2.14.9 IEZNcTool3::GetToolLifeType2 Getting the tool life management type

☐Calling procedure (Custom interface)

```
HRESULT        GetToolLifeType2(
                LONG* plType,          //  (O)  Management type
                LONG* plRet            //  (O)  Error code
                )
```

☐Calling procedure (Automation interface)

```
                Tool_GetToolLifeType2(
                plType As LONG*        //  (O)  Management type
                )As LONG               //  (O)  Error code
```

☐Argument

plType: Returns the tool life management type

Value	Meaning
0	Invalid (Not returned with **MELDASMAGIC64**.)
1	Type I
2	Type II
3	Type III (M6x5L only)

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_READ_READ: Impossible to read the data
EZ_ERR_DATA_RANGE: Data range of the argument is illegal

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This gets the management type of tool life.

☐Reference

SetToolLifeType2()

☐Designation

2.14.10 IEZNcTool3::SetToolLifeType2 — Setting tool life management type

☐Calling procedure (Custom interface)

```
HRESULT        SetToolLifeType2(
                LONG lType,          //  (I)  Management type
                LONG* plRet          //  (O)  Error code
                )
```

☐Calling procedure (Automation interface)

```
                Tool_SetToolLifeType2(
                lType As LONG        //  (I)  Management type
                )As LONG             //  (O)  Error code
```

☐Argument

lType: Set tool life management type

Value	Meaning
0	Invalid (Not returned with **MELDASMAGIC64**.)
1	Type I
2	Type II
3	Type III (M6x5L only)

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_WRITE_WRITE: Impossible to write the data
EZ_ERR_DATA_RANGE: Data range of the argument is illegal
EZ_ERR_DATA_TYPE: Data type of the argument is illegal

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This selects either Type I or Type II of tool life management. **Magic64** is not corresponded.
With the other types, writing is prohibited by the password mode. (**EZNC_DATA_WRITE_WRITE** will be returned.)

☐Reference

GetToolLifeType2()

☐Designation

M6x5M C64 CNC700

2.14.11 IEZNcTool3::GetToolLifeGroupList — Getting the tool life management group # list

☐Calling procedure (Custom interface)

```
HRESULT        GetToolLifeGroupList(
                LPDWORD lpdwLength,          //  (O)  The number of groups
                LPDWORD* lppdwGroup,         //  (O)  Array of the group #
                LONG* plRet                  //  (O)  Error code
                )
```

☐Calling procedure (Automation interface)

```
                 Tool_ GetToolLifeGroupList (
                pvGroup As VARIANT*          //  (O)  Group #
                )As LONG                     //  (O)  Error code
```

☐Argument

lpdwLength: Returns the number of group sets

lppdwGroup: Returns the group # list as array. The array is saved inside of EZSocket, so the client is required to release the memory area explicitly by using **CoTaskMemFree()**.

Automation argument:
pvGroup: Returns the array of group # as **VARIANT**.

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_READ_READ: Impossible to read the data
EZ_ERR_MEMORY_ALLOC: Memory cannot be saved
EZ_ERR_DATA_RANGE: Data range of the argument is illegal
EZNC_DATA_READ_ADDR: Part system designation is illegal
EZNC_DATA_TLFGROUP_ADDR: Address (part system designation) is illegal

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This gets the group # of tool life.

☐Reference

AddToolLifeGroup(), ChangeToolLifeGroup(), DeleteToolLifeGroup()

☐Designation

System (Only CNC700)

2.14.12 IEZNcTool3::AddToolLifeGroup — Adding tool life management group

☐Calling procedure (Custom interface)

```
HRESULT        AddToolLifeGroup(
                DWORD dwGroup,          //  (I)  Group #
                LONG* plRet             //  (O)  Error code
                )
```

☐Calling procedure (Automation interface)

```
                 Tool_AddToolLifeGroup(
                lGroup As LONG          //  (I)  Group #
                )As LONG                //  (O)  Error code
```

☐Argument

dwGroup: Set group # to add

Automation argument:
lGroup: Refer to the explanation of *dwGroup.*

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_WRITE_WRITE: Impossible to write the data
EZ_ERR_DATA_RANGE: Data range of the argument is illegal
EZNC_DATA_DUPLICATE: The # already exists
EZNC_DATA_TLFGROUP_ADDR: Part system designation is illegal
EZNC_DATA_TLFGROUP_EXIST: Target group number already exists
EZNC_DATA_TLFGROUP_OVER: The number of group registrations over
EZNC_DATA_TLFGROUP_OUTOFSPEC: Group number is out of range

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This is for adding group # of tool life management.

☐Reference

GetToolLifeGroupList(), ChangeToolLifeGroup(), DeleteToolLifeGroup()

☐Designation

System (Only C6/C64)

M6x5M C64 CNC700

2.14.13 IEZNcTool3::ChangeToolLifeGroup — Changing tool life management group

☐Calling procedure (Custom interface)

```
HRESULT        ChangeToolLifeGroup(
                 DWORD dwSrcGroup,        //  (I)  Current group #
                 DWORD dwDstGroup,        //  (I)  New group #
                 LONG* plRet              //  (O)  Error code
                 )
```

☐Calling procedure (Automation interface)

```
                 Tool_ChangeToolLifeGroup(
                 lSrcGroup As LONG        //  (I)  Current group #
                 lDstGroup As LONG        //  (I)  New group #
                 )As LONG                 //  (O)  Error code
```

☐Argument

dwSrcGroup: Set the current group #

dwDstGroup: Set the new group #

Automation argument:
lSrcGroup: Refer to the explanation of *dwSrcGroup.*
lDstGroup: Refer to the explanation of *dwDstGroup.*

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_WRITE_WRITE: Impossible to write the data
EZNC_DATA_NOT_EXIST: The data doesn't exist
EZ_ERR_DATA_RANGE: Data range of the argument is illegal
EZNC_DATA_READ_ADDR: Part system designation is illegal
EZNC_DATA_TLFGROUP_ADDR: Address (part system designation) is illegal
EZNC_DATA_TLFGROUP_EXIST: The group # already exists
EZNC_DATA_TLFGROUP_NONEXIST: The group # doesn't exist
EZNC_DATA_TLFGROUP_OUTOFSPEC: Designated group # is out of the specifications

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This changes the designated group # to new one. The designated current group's tools are changed to be the ones of the new group.

☐Reference

GetToolLifeGroupList(), AddToolLifeGroup(), DeleteToolLifeGroup()

☐Designation

System (Only CNC700)

M6x5M C64 CNC700

2.14.14 IEZNcTool3::DeleteToolLifeGroup — Deleting tool life management group

☐Calling procedure (Custom interface)

```
HRESULT        DeleteToolLifeGroup(
                 DWORD dwGroup,          //  (I)  Group #
                 LONG* plRet             //  (O)  Error code
                 )
```

☐Calling procedure (Automation interface)

```
                 Tool_DeleteToolLifeGroup(
                 lGroup As LONG          //  (I)  Group #
                 )As LONG                //  (O)  Error code
```

☐Argument

dwGroup: Set group # to delete

Automation argument:
lGroup: Refer to the explanation of *dwGroup.*

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_WRITE_WRITE: Impossible to write the data
EZNC_DATA_NOT_EXIST: The data doesn't exist
EZ_ERR_DATA_RANGE: Data range of the argument is illegal
EZNC_DATA_READ_ADDR: Part system designation is illegal
EZNC_DATA_TLFGROUP_ADDR: Address (part system designation) is illegal
EZNC_DATA_TLFGROUP_NONEXIST: The group # doesn't exist
EZNC_DATA_TLFGROUP_OUTOFSPEC: Designated group # is out of the specifications

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This deletes group #.

☐Reference

GetToolLifeGroupList(), AddToolLifeGroup(), ChangeToolLifeGroup()

☐Designation

System (Only CNC700)

M6x5M C64 CNC700

2.14.15 IEZNcTool3::GetToolLifeToolNoList Getting the tool list in the tool life management group

☐Calling procedure (Custom interface)

```
HRESULT        GetToolLifeToolNoList(
                 DWORD dwGroup,              //  (I)  Group #
                 LPDWORD lpdwLength,         //  (O)  The number of tools registered
                 LPDWORD* lppdwToolNo,       //  (O)  Array of tool #s
                 LONG* plRet                 //  (O)  Error code
                 )
```

☐Calling procedure (Automation interface)

```
                 Tool_ GetToolLifeToolNoList (
                 lGroup As LONG              //  (I)  Group #
                 pvToolNo As VARIANT         //  (O)  Array of tool #s
                 )As LONG                    //  (O)  Error code
```

☐Argument

dwGroup: Set group # whose tool # list you wish to get

lpdwLength: Returns the number of tools registered (array length of tool # list) in the group

lppdwToolNo: Returns the tool # list in the group as an array. The array is saved inside of EZSocket, so the client is required to release the memory area explicitly by using **CoTaskMemFree()**.
In the case of L system type, the ToolNo is expressed as the tool # + tool offset #.

Automation argument:
lGroup: Refer to the explanation of *dwGroup.*
pvToolNo: Returns the array of the list of tool # included in the group as **VARIANT**.

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_READ_READ: Impossible to read the data
EZ_ERR_MEMORY_ALLOC: Memory cannot be saved
EZ_ERR_DATA_RANGE: Data range of the argument is illegal
EZNC_DATA_TLFGROUP_ADDR: Address (part system designation) is illegal
EZNC_DATA_TLFGROUP_NONEXIST: The group # doesn't exist
EZNC_DATA_TLFGROUP_OUTOFSPEC: Designated group # is out of the specifications

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This gets tool # list of the designated group.

☐Reference

AddToolLifeToolNo(), ChangeToolLifeToolNo(), DeleteToolLifeToolNo()

☐Designation

System (Only CNC700)

M6x5M C64 CNC700

2.14.16 IEZNcTool3::AddToolLifeToolNo — Adding tool # in the tool life management group

☐Calling procedure (Custom interface)

```
HRESULT        AddToolLifeToolNo(
                DWORD dwGroup,          //  (I)  Group #
                DWORD dwToolNo,         //  (I)  Tool #
                LONG* plRet             //  (O)  Error code
                )
```

☐Calling procedure (Automation interface)

```
                 Tool_ AddToolLifeToolNo(
                lGroup As LONG          //  (I)  Group #
                lToolNo As LONG         //  (I)  Tool #
                )As LONG                //  (O)  Error code
```

☐Argument

dwGroup: Set group # whose tool #s you wish to get

dwToolNo: Set tool # to add

Automation argument:
lGroup: Refer to the explanation of *dwGroup.*
lToolNo: Refer to the explanation of *dwToolNo.*

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_WRITE_WRITE: Impossible to write the data
EZNC_DATA_DUPLICATE: The # already exists
EZ_ERR_DATA_RANGE: Data range of the argument is illegal

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This is for adding tool #s to the designated group.

☐Reference

GetToolLifeToolNoList(), ChangeToolLifeToolNo(), DeleteToolLifeToolNo()

☐Designation

System (Only CNC700)

M6x5M C64 CNC700

2.14.17 IEZNcTool3::ChangeToolLifeToolNo — Changing tool # of tool life management

☐Calling procedure (Custom interface)

```
HRESULT        ChangeToolLifeToolNo(
                 DWORD dwGroup,          //  (I)  Group #
                 DWORD dwSrcToolNo,      //  (I)  Current tool #
                 DWORD dwDstToolNo,      //  (I)  New tool #
                 LONG* plRet             //  (O)  Error code
                 )
```

☐Calling procedure (Automation interface)

```
                  Tool_ChangeToolLifeToolNo (
                 lGroup As LONG          //  (I)  Group #
                 lSrcToolNo As LONG      //  (I)  Current tool #
                 lDstToolNo As LONG      //  (I)  New tool #
                 )As LONG                //  (O)  Error code
```

☐Argument

dwGroup: Set group # whose tool #s you wish to change
(Reserved for **Magic64, M6x5L, and M6x5M**, *dwGroup* = Fixed to 0)

dwSrcToolNo: Set the current tool #
In the case of L system type, the ToolNo is expressed as the tool # + tool offset #.

dwDstToolNo: Set the new tool #

Automation argument:
lGroup: Refer to the explanation of *dwGroup.*
lSrcToolNo: Refer to the explanation of *dwSrcToolNo.*
lDstToolNo: Refer to the explanation of *dwDstToolNo.*

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_WRITE_WRITE: Impossible to write the data
EZNC_DATA_NOT_EXIST: The data doesn't exist
EZ_ERR_DATA_RANGE: Data range of the argument is illegal
EZNC_DATA_TLFGROUP_ADDR: Address (part system designation) is illegal
EZNC_DATA_TLFGROUP_NONEXIST: The group # doesn't exist
EZNC_DATA_TLFGROUP_OUTOFSPEC: Designated group # is out of the specifications
EZNC_DATA_TLFTOOL_EXIST: The tool # already exists
EZNC_DATA_TLFTOOL_NONEXIST: The tool # doesn't exist
EZNC_DATA_TLFTOOL_OUTOFSPEC: Designated tool # is out of the specifications

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This changes the designated tool # to the new one.

☐**Restriction**	When using **ChangeToolLifeToolNo()** for **C64**, please remind that designating the same tool # among different group #s does not show the error. Take care not to designate the same tool # among the applications.
☐**Reference**	**GetToolLifeToolNoList(), AddToolLifeToolNo(), DeleteToolLifeToolNo()**
☐**Designation**	System (Only CNC700)

M6x5M C64 CNC700

2.14.18 IEZNcTool3::DeleteToolLifeToolNo — Deleting tool # of tool life management

☐Calling procedure (Custom interface)

```
HRESULT        DeleteToolLifeToolNo(
                 DWORD dwGroup,          //  (I)  Group #
                 DWORD dwToolNo,         //  (I)  Tool #
                 LONG* plRet             //  (O)  Error code
                 )
```

☐Calling procedure (Automation interface)

```
                 Tool_ DeleteToolLifeToolNo (
                 lGroup As LONG          //  (I)  Group #
                 lToolNo As LONG         //  (I)  Tool #
                 )As LONG                //  (O)  Error code
```

☐Argument

dwGroup: Set group # whose tool #s you wish to delete
(Reserved for **Magic64, M6x5L, and M6x5M**, *dwGroup* = Fixed to 0)

dwToolNo: Set tool # to delete

Automation argument:
lGroup: Refer to the explanation of *dwGroup.*
lToolNo: Refer to the explanation of *dwToolNo.*

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_WRITE_WRITE: Impossible to write the data
EZNC_DATA_NOT_EXIST: The data doesn't exist
EZ_ERR_DATA_RANGE: Data range of the argument is illegal
EZNC_DATA_TLFGROUP_ADDR: Address (part system designation) is illegal
EZNC_DATA_TLFGROUP_NONEXIST: The group # doesn't exist
EZNC_DATA_TLFGROUP_OUTOFSPEC: Designated group # is out of the specifications
EZNC_DATA_TLFTOOL_NONEXIST: The tool # doesn't exist
EZNC_DATA_TLFTOOL_OUTOFSPEC: Designated tool # is out of the specifications

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This deletes the designated tool #.

☐Reference

GetToolLifeToolNoList(), AddToolLifeToolNo(), ChangeToolLifeToolNo()

☐Designation

System (Only CNC700)

2.14.19 IEZNcTool3::GetToolLifeValue — Getting the tool life management data

☐Calling procedure (Custom interface)

```
HRESULT        GetToolLifeValue(
                DWORD dwGroup,              // (I)  Group #
                DWORD dwToolNo,             // (I)  Tool #
                LPOLESTR** lpppwszData,     // (O) Array of tool life management data value character string
                LONG* plRet                 // (O) Error code
                )
```

☐Calling procedure (Automation interface)

```
               Tool_ GetToolLifeValue (
                lGroup As LONG              // (I)  Group #
                lToolNo As LONG             // (I)  Tool #
                pvData As VARIANT*          // (O) Array of tool life management data value character string
                )As LONG                    // (O) Error code
```

☐Argument

dwGroup: Set group # whose tool life data you wish to delete
(Reserved for **Magic64, M6x5L, and M6x5M**, *dwGroup* = Fixed to 0)

dwToolNo: Set tool # whose tool life you wish to get
In the case of L system type, the ToolNo is expressed as the tool # + tool offset #.

lpppwszData: Returns the tool life data value as **UNICODE** character string. The array is saved inside of EZSocket, so the client is required to release the memory area explicitly by using **CoTaskMemFree()**. Refer to the Index table.

Automation argument:
lGroup: Refer to the explanation of *dwGroup.*
lToolNo: Refer to the explanation of *dwToolNo.*
pvData: Returns the array of life management data value (**UNICODE** character string) as **VARIANT**. Refer to the index for life management data value.

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_READ_READ: Impossible to read the data
EZNC_DATA_NOT_EXIST: The data doesn't exist
EZ_ERR_MEMORY_ALLOC: Memory cannot be saved
EZ_ERR_DATA_RANGE: Data range of the argument is illegal
EZNC_DATA_TLFGROUP_ADDR: Address (part system designation) is illegal
EZNC_DATA_TLFGROUP_NONEXIST: The group # doesn't exist
EZNC_DATA_TLFGROUP_OUTOFSPEC: Designated group # is out of the specifications
EZNC_DATA_TLFTOOL_NONEXIST: The tool # doesn't exist
EZNC_DATA_TLFTOOL_OUTOFSPEC: Designated tool # is out of the specifications

Index table

Array index	Kinds of tool life (Data range)	
	M6x5M	**CNC700M**
0	Tool # (1 to 99999999)	Tool # (1 to 99999999)
1	Status (According to the machine builder's spec)	Status (0x00 to 0xFF)
2	Type (000 to 223) *	Type (000 to 222) *
3	Length offset (-/+ 1 to 99999.999)	Length offset (-/+ 99999.999)
4	Diameter offset (-/+ 1 to 99999.999)	Diameter offset (/-+ 99999.999)
5	Life (Time: 0 to 4000, the number of times used: 0 to 9999)	Life (Time: 0 to 4000, the number of times used: 0 to 9999/65000)
6	Usage (Time: 0 to 4000, the number of times used: 0 to 9999)	Usage (Time: 0 to 4000, the number of times used: 0 to 9999/65000)
7	Aux (0 to 65535, according to the machine builder's spec)	Aux (0 to 65535, according to the machine builder's spec)
8	Length wear (Reserved: 0)	Length wear (Reserved: 0)
9	Diameter wear (Reserved: 0)	Diameter wear (Reserved: 0)
10	Group (1 to 99999999)	Group (1 to 99999999)

Array index	Kinds of tool life (Data range)	
	C6/C64M	
0	Tool # (1 to 99999999)	
1	Status (According to the machine tool builder's spec)	
2	Type (000 to 222) *	
3	Length offset (±1 to 99999.999)	
4	Diameter offset (±1 to 99999.999)	
5	Life (Time: 0 to 4000, the number of times used: 0 to 9999)	
6	Usage (Time: 0 to 4000, the number of times used: 0 to 9999)	
7	Aux (0 to 65535, according to the machine tool builder's spec)	
8	Length wear (Reserved: 0)	
9	Diameter wear (Reserved: 0)	
10	Group (1 to 99999999)	

Array index	Kinds of tool life (Data range)	
	M6x5L, **C6/C64L**(TYPE I), **CNC700L** (TYPE I)	**CNC700L** (TYPE II)
0	Time-managed usage (0 to 995959)	Tool # (1 to 999999)
1	Number-of-times-managed usage (0 to 9999)	Offset # (0 to 80)
2	Status A (0 to 2)	Usage (0 to 999999)
3	Time-managed life (0 to 995959)	ST (0 to 3)
4	Number-of-times-managed life (0 to 9999)	Type (Time: 0, the number of times used: 1)
5	Status B (According to the machine tool builder's spec)	Life (0 to 999999)
6 to 10	-	-

Array index	Kinds of tool life (Data range)	
	C6/C64L(TYPE II)	
0	Tool # (1 to 999999)	
1	Group (1 to 9999)	
2	Type (0: Time, 1: Number of times used)	
3	Offset # (1 to 80)	
4	Status (0 to 3)	
5	Life (Time: 0 to 999999, the number of times used: 0 to 999999)	
6	Usage (Time: 0 to 999999, the number of times used: 0 to 999999)	
7 to 10	-	

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This gets the life management data of the designated tool #. The number of elements of character strings that return the life management data depends on the machine model.
*As for "Type" of tool life management data, refer to the Instruction Manual for each system.

☐Reference

SetToolLifeValue()

☐Designation

System (Only CNC700)

M6x5M | M6x5L | C64 | CNC700

2.14.20 IEZNcTool3::SetToolLifeValue — Respective setting of tool life management data

☐Calling procedure (Custom interface)

```
HRESULT        SetToolLifeValue(
                 DWORD dwGroup,          //  (I)  Group #
                 DWORD dwToolNo,         //  (I)  Tool #
                 DWORD dwKind,           //  (I)  Kind of life management data
                 LPCOLESTR lpcwszData,   //  (I)  Life management data
                 LONG* plRet             //  (O)  Error code
                 )
```

☐Calling procedure (Automation interface)

```
                 Tool_SetToolLifeValue (
                 lGroup As LONG          //  (I)  Group #
                 lToolNo As LONG         //  (I)  Tool #
                 lKind As LONG           //  (I)  Kind of life management data
                 lpcwszData As STRING    //  (I)  Life management data
                 )As LONG                //  (O)  Error code
```

☐Argument

dwGroup: Set group # whose tool life data you wish to set
(Reserved for **Magic64, M6x5L, and M6x5M**, *dwGroup* = Fixed to 0)

dwToolNo: Set tool # whose tool life you wish to set

dwKind: Set tool life type. Refer to the parameter table.

lpcwszData: Set life data of the designated kind

Automation argument:
lGroup: Refer to the explanation of *dwGroup.*
lToolNo: Refer to the explanation of *dwToolNo.*
lKind: Refer to the explanation of *dwKind.*

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_WRITE_WRITE: Impossible to write the data
EZNC_DATA_NOT_EXIST: The data doesn't exist
EZ_ERR_DATA_RANGE: Data range of the argument is illegal (*dwKind*)
EZNC_DATA_TLFGROUP_ADDR: Address (part system designation) is illegal
EZNC_DATA_TLFGROUP_NONEXIST: The group # doesn't exist
EZNC_DATA_TLFGROUP_OUTOFSPEC: Designated group # is out of the specifications
EZNC_DATA_TLFTOOL_NONEXIST: The tool # doesn't exist
EZNC_DATA_TLFTOOL_PARAMERR: Designated kind of tool life management data is illegal
EZNC_DATA_TLFTOOL_MAXMINERR: Data set is over the range
EZNC_DATA_TLFTOOL_OUTOFSPEC: Designated tool # is out of the specifications

Parameter table

Value	Kinds of tool life (Data range)	
	M6x5M	**CNC700M**
1	Tool # (1 to 99999999)	Tool # (1 to 99999999)
2	Status (According to the machine builder's spec)	Status (0x00 to 0cFF)
3	Type (000 to 223) *	Type (000 to 222) *
4	Length offset (-/+ 1 to 99999.999)*	Length offset (-/+ 99999.999)*
5	Diameter offset (-/+ 1 to 99999.999)*	Diameter offset (-/+ 99999.999)*
6	Life (Time: 0 to 4000, the number of times used: 0 to 9999)	Life (Time: 0 to 4000, the number of times used: 0 to 9999/65000)
7	Usage (Time: 0 to 4000, the number of times used: 0 to 9999)	Usage (Time: 0 to 4000, the number of times used: 0 to 9999/65000)
8	Aux (0 to 65535, according to the machine builder's spec)	Aux (0 to 65535, according to the machine builder's spec)
9	Length wear (Reserved: 0.000)	Length wear (Reserved: 0)
10	Diameter wear (Reserved: 0.000)	Diameter wear (Reserved: 0)
11	Group (1 to 99999999)	Group (1 to 99999999)

Value	Kinds of tool life (Data range)	
	C6/C64M	
1	Tool # (1 to 99999999)	
2	Status (According to the machine tool builder's spec)	
3	Type (000 to 222) *	
4	Length offset (±1 to 99999.999)	
5	Diameter offset (±1 to 99999.999)	
6	Life (Time: 0 to 4000, the number of times used: 0 to 9999)	
7	Usage (Time: 0 to 4000, the number of times used: 0 to 9999)	
8	Aux (0 to 65535, according to the machine tool builder's spec)	
9	Length wear (Reserved: 0)	
10	Diameter wear (Reserved: 0)	
11	Group (1 to 99999999)	

Value	Kinds of tool life (Data range)	
	M6x5L, **C6/C64L**(TYPE I), **CNC700L** (TYPE I)	**CNC700L** (TYPE II)
1	Time-managed usage (0 to 995959)	Tool # (1 to 999999)
2	Number-of-times-managed usage (0 to 9999)	Offset # (0 to 80)
3	Status A (0 to 2)	Usage (0 to 999999)
4	Time-managed life (0 to 995959)	ST (0 to 3)
5	Number-of-times-managed life (0 to 9999)	Type (Time: 0, the number of times used: 1)
6	Status B (According to the machine tool builder's spec)	Life (0 to 999999)

Value	Kinds of tool life (Data range)	
	C6/C64L(TYPE II)	
1	Tool # (1 to 999999)	
2	Group (1 to 9999)	
3	Type (0: Time, 1: Number of times used)	
4	Offset # (1 to 80)	
5	Status (0 to 3)	
6	Life (Time: 0 to 999999, the number of times used: 0 to 999999)	
7	Usage (Time: 0 to 999999, the number of times used: 0 to 999999)	

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This sets life management data of the designate tool # respectively. As this method is for renewing, follow the following procedure to add a new tool.

1) AddToolLifeGroup( )
2) AddToolLifeToolNo( )
3) SetToolLifeValue( )

*As for "Type" of tool life management data, refer to the Instruction Manual for each system.

☐Reference

GetToolLifeValue(), AddToolLifeGroup(), AddToolLifeToolNo()

☐Designation

System (Only CNC700)

2.14.21 IEZNcTool3::SetToolLifeValue2 — Setting tool life management data

☐Calling procedure (Custom interface)

```
HRESULT     SetToolLifeValue2(
              DWORD dwGroup,              // (I) Group #
              DWORD dwToolNo,             // (I) Tool #
              LPCOLESTR* lppcwszData,     // (I) Array of tool life management data value character string
              LONG* plRet                 // (O) Error code
              )
```

☐Calling procedure (Automation interface)

```
            Tool_SetToolLifeValue2(
            lGroup As LONG                // (I) Group #
            lToolNo As LONG               // (I) Tool #
            vData As VARIANT              // (I) Array of tool life management data value character string
            )As LONG                      // (O) Error code
```

☐Argument

dwGroup: Set group # whose tool life data you wish to set
(Reserved for **Magic64, M6x5L, and M6x5M**, *dwGroup* = Fixed to 0)

dwToolNo: Set tool # whose tool life you wish to get

lppcwszData: Set **UNICODE** character string array of life data of the designated kind (Surely save the 11 size of array.)

Automation argument:
lGroup: Refer to the explanation of *dwGroup.*
lToolNo: Refer to the explanation of *dwToolNo.*
vData: Create UNICODE character string of the specified kind of life data and substitute the character string for *vData*(VARIANT) to set.
Refer to "2.11.13 WriteFile" for the example of substitution.

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_WRITE_WRITE: Impossible to write the data
EZNC_DATA_NOT_EXIST: The data doesn't exist
EZ_ERR_DATA_RANGE: Data range of the argument is illegal
EZ_ERR_NULLPTR: The argument is NULL pointer
EZNC_DATA_TLFGROUP_ADDR: Address (part system designation) is illegal
EZNC_DATA_TLFGROUP_NONEXIST: The group # doesn't exist
EZNC_DATA_TLFGROUP_OUTOFSPEC: Designated group # is out of the specifications
EZNC_DATA_TLFTOOL_NONEXIST: The tool # doesn't exist
EZNC_DATA_TLFTOOL_MAXMINERR: Data set is over the range
EZNC_DATA_TLFTOOL_OUTOFSPEC: Designated tool # is out of the specifications

☐ **Argument**

Array index	Kinds of tool life (Data range)	
	M6x5M	**CNC700M**
0	Tool # (Reserved: 0)	Tool # (Reserved: 0)
1	Status (According to the machine builder's spec)	Status (0x00 to 0xFF)
2	Type (000 to 223) *	Type (000 to 222) *
3	Length offset (-/+ 1 to 99999.999)	Length offset (-/+ 99999.999) *
4	Diameter offset (-/+ 1 to 99999.999)	Diameter offset (-/+ 99999.999) *
5	Life (Time: 0 to 4000, the number of times used: 0 to 9999)	Life (Time: 0 to 4000, the number of times used: 0 to 9999/65000)
6	Usage (Time: 0 to 4000, the number of times used: 0 to 9999)	Usage (Time: 0 to 4000, the number of times used: 0 to 9999/65000)
7	Aux (0 to 65535, according to the machine builder's spec)	Aux (0 to 65535, according to the machine builder's spec)
8	Length wear (Reserved: 0.000)	Length wear (Reserved: 0)
9	Diameter wear (Reserved: 0.000)	Diameter wear (Reserved: 0)
10	Group (Reserved: 0)	Group (Reserved: 0)

Array index	Kinds of tool life (Data range)	
	C6/C64M	
0	Tool # (1 to 99999999)	
1	Status (According to the machine builder's spec)	
2	Type (000 to 222) *	
3	Length offset (±1 to 99999.999)	
4	Diameter offset (±1 to 99999.999)	
5	Life (Time: 0 to 4000, the number of times used: 0 to 9999)	
6	Usage (Time: 0 to 4000, the number of times used: 0 to 9999)	
7	Aux (0 to 65535, according to the machine tool builder's spec)	
8	Length wear (Reserved: 0)	
9	Diameter wear (Reserved: 0)	
10	Group (1 to 99999999)	

Array index	Kinds of tool life (Data range)	
	M6x5L, **C6/C64L**(TYPE I), **CNC700L** (TYPE I)	**CNC700L** (TYPE II)
0	Time-managed usage (0 to 995959)	Tool # (Reserved: 0)
1	Number-of-times-managed usage (0 to 9999)	Offset # (0 to 80)
2	Status A (0 to 2)	Usage (0 to 999999)
3	Time-managed life (0 to 995959)	ST (0 to 3)
4	Number-of-times-managed life (0 to 9999)	Type (Time: 0, the number of times used: 1)
5	Status B (According to the machine tool builder's spec)	Life (0 to 999999)
6 to 10	-	-

☐**Argument**

Array index	Kinds of tool life (Data range)	
	C6/C64L(TYPE II)	
0	Tool # (1 to 999999)	
1	Group (1 to 9999)	
2	Type (0: Time, 1: Number of times used)	
3	Offset # (1 to 80)	
4	Status (0 to 3)	
5	Life (Time: 0 to 999999, the number of times used: 0 to 999999)	
6	Usage (Time: 0 to 999999, the number of times used: 0 to 999999)	
7-10	-	

☐**Return value**

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐**Functions**

This sets life management data of the designate tool #. As this method is for renewing, follow the following procedure to add a new tool.

1) AddToolLifeGroup( )
2) AddToolLifeToolNo( )
3) SetToolLifeValue2( )

*As for "Type" of tool life management data, refer to the Instruction Manual for each system.

[E.g.] In the case of M6x5M

```
LPOLESTR* lppwszData;
lppwszData = new LPOLESTR[11];
lppwszData[0] =L"0";
lppwszData[1] =L"1";
lppwszData[2] =L"220";
lppwszData[3] =L"10.000";
lppwszData[4] =L"20.000";
lppwszData[5] =L"40.000";
lppwszData[6] =L"18.000";
lppwszData[7] =L"0";
lppwszData[8] =L"0.000";
lppwszData[9] =L"0.000";
lppwszData[10] =L"0";
hr = pIEZNcTool->SetToolLifeValue2( 1, 100, (LPCOLESTR*)lppwszData, &lRet);
if( S_OK != hr ){
        wprintf(L"HRESULT Code = 0x%x, lRet Code = 0x%x\n", hr, lRet );
}
delete[ ] lppwszData;
```

☐**Reference**

GetToolLifeValue(), AddToolLifeGroup(), AddToolLifeToolNo()

☐**Designation**

System (Only CNC700)

2.14.22 IEZNcTool3::GetSpareTool — Getting the spare tool for tool life management

☐Calling procedure (Custom interface)

```
HRESULT        GetSpareTool(
                 DWORD dwNo,                  //  (I)  #
                 DWORD dwToolKind,            //  (I)  Kind of spare tool
                 LPOLESTR** lpppwszData,      //  (O)  Array of tool exchange data value character string
                 LONG* plRet                  //  (O)  Error code
                 )
```

☐Calling procedure (Automation interface)

```
                 Tool_ GetSpareTool(
                 lNo As LONG                  //  (I)  #
                 lToolKind As LONG            //  (I)  Kind of spare tool
                 pvData As VARIANT*           //  (O)  Array of tool exchange data value character string
                 )As LONG                     //  (O)  Error code
```

☐Argument

dwNo: Set # whose tool exchange data you wish to get

dwToolKind: Set kind of the spare tool

Value	Meaning
0	Master tool
1	Spare tool 1
2	Spare tool 2
3	Spare tool 3

lpppwszData: Returns the tool exchange data as **UNICODE** character string array. The array is saved inside of EZSocket, so the client is required to release the memory area explicitly by using **CoTaskMemFree()**. Refer to the index table.

Automation argument:
lNo: Refer to the explanation of *dwNo.*
lToolKind: Refer to the explanation of *dwToolKind.*
pvData: Returns the array of tool exchange data value (**UNICODE** character string) as **VARIANT**.

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_READ_READ: Impossible to read the data
EZ_ERR_MEMORY_ALLOC: Memory cannot be saved
EZ_ERR_DATA_RANGE: Data range of the argument is illegal

Index table

Array index	Kind of tool exchange data	Data range	Remarks
0	Tool #	-	Refer to the Instruction Manual of **M6x5L**
1	Status	0 to 2	
2	Offset	-	Refer to the Instruction Manual of **M6x5L**

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐**Functions**	This gets the tool exchange data of the designated # and spare tool kind.
☐**Reference**	**SetSpareTool()**
☐**Designation**	

2.14.23 IEZNcTool3::SetSpareTool — Setting spare tool for tool life management

☐Calling procedure (Custom interface)

```
HRESULT        SetSpareTool(
                 DWORD dwNo,                  //  (I)  #
                 DWORD dwToolKind,            //  (I)  Kind of spare tool
                 DWORD dwKind,                //  (I)  Kind of tool exchange data
                 LPCOLESTR lpcwszData,        //  (I)  Tool exchange data
                 LONG* plRet                  //  (O)  Error code
                 )
```

☐Calling procedure (Automation interface)

```
                 Tool_SetSpareTool(
                 lNo As LONG                  //  (I)  #
                 lToolKind As LONG            //  (I)  Kind of spare tool
                 lKind As LONG                //  (I)  Kind of tool exchange data
                 lpcwszData As STRING         //  (I)  Tool exchange data
                 )As LONG                     //  (O)  Error code
```

☐Argument

dwNo: Set # whose tool exchange data you wish to set

dwToolKind: Set kind of the spare tool

Value	Meaning
0	Master tool
1	Spare tool 1
2	Spare tool 2
3	Spare tool 3

dwKind: Set kind of the spare tool data. Refer to the parameter table.

lpcwszData: Set life data of the designated kind

Automation argument:
lNo: Refer to the explanation of *dwNo.*
lToolKind: Refer to the explanation of *dwToolKind.*
lKind: Refer to the explanation of *dwToolKind.*

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_WRITE_WRITE: Impossible to write the data
EZ_ERR_DATA_RANGE: Data range of the argument is illegal (*dwToolKind, dwKind*)

Parameter table

Value	Kind of tool exchange data	Data range	Remarks
1	Tool #	-	Refer to the Instruction Manual of M6x5L
2	Status	0 to 2	
3	Offset	-	Refer to the Instruction Manual of M6x5L

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐**Functions**	This sets the tool exchange data of the designated # and spare tool kind.
☐**Reference**	**GetSpareTool()**
☐**Designation**	

2.15 IEZNcATC2 Interface

Magic64 | M6x5M | C64 | CNC700

2.15.1 IEZNcATC2::GetMGNControl — Getting the control parameter for ATC tool register

☐Calling procedure (Custom interface)

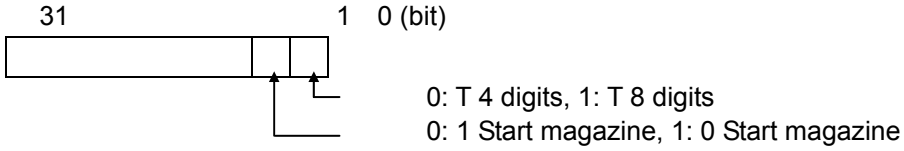
```
HRESULT        GetMGNControl(
                LONG* plData,          //  (O)  Patameter value
                LONG* plRet            //  (O)  Error code
                )
```

☐Calling procedure (Atomation interface)

```
                ATC_ GetMGNControl (
                plData As LONG*        //  (O)  Patameter value
                )As LONG               //  (O)  Error code
```

☐Argument

plData: Returns the parameter that controls the start magazine



plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_READ_READ: Impossible to read the data

☐Return value

Return value	Details
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This gets the control parameter of the ATC tool registration.

☐Reference

☐Designation

Magic64 M6x5M C64 CNC700

2.15.2 IEZNcATC2::GetMGNSize — Getting the total number of the ATC magazine pot sets

☐Calling procedure (Custom interface)

```
HRESULT        GetMGNSize(
                 LONG* plSize,          //  (O)  The number of magazine pot sets
                 LONG* plRet            //  (O)  Error code
                 )
```

☐Calling procedure (Automation interface)

```
                 Tool_ GetMGNSize (
                 plSize As LONG*        //  (O)  The number of magazine pot sets
                 )As LONG               //  (O)  Error code
```

☐Argument

plSize: Returns the total number of magazine pot sets
Value: 0 to 360 (Max)

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_READ_READ: Impossible to read the data

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This gets the total number of magazine pot sets.

☐Reference

☐Designation

M6x5M C64 CNC700

2.15.3 IEZNcATC2::GetMGNSize2 — Getting the number of pots of each ATC magazine

☐Calling procedure (Custom interface)

```
HRESULT        GetMGNSize2(
                LONG lMagazineNo,        //  (I)  Magazine #
                LONG* plSize,            //  (O)  The number of magazine pot sets
                LONG* plRet              //  (O)  Error code
                )
```

☐Calling procedure (Automation interface)

```
                 Tool_ GetMGNSize2(
                lMagazineNo As LONG      //  (I)  Magazine #
                plSize As LONG*          //  (O)  The number of magazine pot sets
                ) As LONG                //  (O)  Error code
```

☐Argument

lMagazineNo: Set magazine #
Value: 1 to 5 (Max)

plSize: Returns the total number of magazine pot sets
Value: 0 to 360 (Max)

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_READ_READ: Impossible to read the data
EZ_ERR_DATA_RANGE: Data range of the argument is illegal (*lMagazineNo*)

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This gets the total number of magazine pot sets designated.

☐Reference

☐Designation

Magic64 M6x5M C64 CNC700

2.15.4 IEZNcATC2::GetMGNReady — Getting the ATC ready tool

☐Calling procedure (Custom interface)

```
HRESULT        GetMGNReady(
                 LONG lReady,          //  (I)  Ready status
                 LONG* plToolNo,       //  (O)  Tool #
                 LONG* plRet           //  (O)  Error code
                 )
```

☐Calling procedure (Automation interface)

```
                 Tool_ GetMGNReady(
                 lReady As LONG        //  (I)  Ready status
                 plToolNo As LONG*     //  (O)  Tool #
                 ) As LONG             //  (O)  Error code
```

☐Argument

lReady: Set ready status

Value	Meaning
0	Tool # of the tool mounted
1	Tool # of the tool ready 1
2	Tool # of the tool ready 2
3	Tool # of the tool ready 3
4	Tool # of the tool ready 4

plToolNo: Returns the tool #
Valute: 0 to 99999999 (Max)

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_READ_READ: Impossible to read the data
EZ_ERR_DATA_RANGE: Data range of the argument is illegal (*lReady, lIndex*)

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This gets the tool #s of ATC tool register.

☐Reference

☐Designation

Magic64 M6x5M C64 CNC700

2.15.5 IEZNcATC2::GetMGNPot — Getting the tool # of the ATC magazine pot

☐Calling procedure (Custom interface)

```
HRESULT        GetMGNPot(
                 LONG lIndex,          //  (I)  Magazine pot #
                 LONG* plToolNo,       //  (O)  Tool #
                 LONG* plRet           //  (O)  Error code
                 )
```

☐Calling procedure (Automation interface)

```
                 Tool_ GetMGNPot (
                 lIndex As LONG        //  (I)  Magazine pot #
                 plToolNo As LONG*     //  (O)  Tool #
                 ) As LONG             //  (O)  Error code
```

☐Argument

lIndex: Set the magazine pot #
Value: 1 to 360 (Max)

plToolNo: Returns the tool #
Valute: 0 to 99999999 (Max)

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_READ_READ: Impossible to read the data
EZ_ERR_DATA_RANGE: Data range of the argument is illegal (*lIndex*)

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This gets the tool #s in the designated pot of Magazine.

☐Reference

SetMGNPot(), GetMGNPotEx()

☐Designation

M6x5M C64 CNC700

2.15.6 IEZNcATC2::GetMGNPot3 — Getting the tool # of each ATC magazine pot

☐Calling procedure (Custom interface)

```
HRESULT        GetMGNPot3(
                 LONG lMagazineNo,        //  (I)  Magazine #
                 LONG lIndex,             //  (I)  Pot #
                 LONG* plToolNo,          //  (O)  Tool #
                 LONG* plRet              //  (O)  Error code
                 )
```

☐Calling procedure (Automation interface)

```
                 Tool_ GetMGNPot3(
                 lMagazineNo As LONG      //  (I)  Magazine #
                 lIndex As LONG           //  (I)  Pot #
                 plToolNo As LONG*        //  (O)  Tool #
                 ) As LONG                //  (O)  Error code
```

☐Argument

lMagazineNo: Set the magazine #
Value: 1 to 5 (Max)

lIndex: Set the pot #
Value: 1 to 360 (Max)

plToolNo: Returns the tool #
Valute: 0 to 99999999 (Max)

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_READ_READ: Impossible to read the data
EZ_ERR_DATA_RANGE: Data range of the argument is illegal (*lMagazineNo* , *lIndex*)

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This gets the tool #s in the designated magazine's pot.

☐Reference

SetMGNPot3()

☐Designation

Magic64 M6x5M C64 CNC700

2.15.7 IEZNcATC2::SetMGNPot — Setting the tool # of the ATC magazine pot

☐Calling procedure (Custom interface)

```
HRESULT        SetMGNPot(
                 LONG lIndex,          //  (I)  Pot #
                 LONG lToolNo,         //  (I)  Tool #
                 LONG* plRet           //  (O)  Error code
                 )
```

☐Calling procedure (Automation interface)

```
                 Tool_SetMGNPot (
                 lIndex As LONG        //  (I)  Pot #
                 lToolNo As LONG       //  (I)  Tool #
                 ) As LONG             //  (O)  Error code
```

☐Argument

lIndex: Set the pot #
Value: 1 to 360 (Max)

lToolNo: Set the tool #
Valute: 1 to 99999999 (Max)

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_WRITE_WRITE: Impossible to write the data
EZ_ERR_DATA_RANGE: Data range of the argument is illegal (*lIndex, lToolNo*)

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This sets tool #s to store in Magazine's pot. Make sure not to duplicate tool #s as duplication won't be checked. An alarm will occur if duplicated. The alarm information can be got by **GetAlarm()**.If a tool # with 5 digits or more is designated for **C64, M6x5L, M6x5M** and **CNC 700M/L** while the specification adopts 4 digits, only the last 4 digits are registered, rounding the rest of the digits off.
Tool # designation with 5 or more digits is not available for **Magic64** (an error occurs).

(Note) Magic64 has different setting ranges of magazine pot # and tool # from those described above. Confirm the ranges in the Instruction Manual for **Magic64**.

☐Reference

GetMGNPot(), SetMGNPotEx(), GetAlarm()

☐Designation

M6x5M C64 CNC700

2.15.8 IEZNcATC2::SetMGNPot3 — Setting tool # of the pot of each ATC magazine

☐Calling procedure (Custom interface)

```
HRESULT        SetMGNPot3(
                 LONG lMagazineNo,      //  (I)  Magazine #
                 LONG lIndex,           //  (I)  Pot #
                 LONG lToolNo,          //  (I)  Tool #
                 LONG* plRet            //  (O)  Error code
                 )
```

☐Calling procedure (Automation interface)

```
                 Tool_SetMGNPot3(
                 lMagazineNo As LONG    //  (I)  Magazine #
                 lIndex As LONG         //  (I)  Pot #
                 lToolNo As LONG        //  (I)  Tool #
                 ) As LONG              //  (O)  Error code
```

☐Argument

lMagazineNo: Set the magazine #
Value: 1 to 5 (Max)

lIndex: Set the pot #
Value: 1 to 360 (Max)

lToolNo: Set the tool #
Valute: 1 to 99999999 (Max)

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_WRITE_WRITE: Impossible to write the data
EZ_ERR_DATA_RANGE: Data range of the argument is illegal (*lMagazineNo, lIndex, lToolNo*)

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This sets tool #s to store in the designated magazine's pot. Make sure not to duplicate tool #s as duplication won't be checked. An alarm will occur if duplicated. The alarm information can be got by **GetAlarm()**.When the tool # of NC system is designed to hold 4 digits, and 5 digits or more are specified, only the last 4 digits are registered, rounding the rest of the digits off.

☐Reference

GetMGNPot3(), GetAlarm()

☐Designation

2.15.9 IEZNcATC2::GetMGNPotEx — Getting the tool # of the ATC extended magazine pot

☐Calling procedure (Custom interface)

```
HRESULT        GetMGNPotEx(
                 LONG lIndex,              //  (I)  Pot #
                 LONG* plToolNo,           //  (O)  Tool #
                 LONG* plRet               //  (O)  Error code
                 )
```

☐Calling procedure (Automation interface)

```
                 Tool_GetMGNPotEx(
                 lIndex As LONG            //  (I)  Pot #
                 plToolNo As LONG*         //  (O)  Tool #
                 ) As LONG                 //  (O)  Error code
```

☐Argument

lIndex: Set the pot #
Value: 1 to 80

plToolNo: Returns the tool #
Value: 1 to 80

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_READ_READ: Impossible to read the data

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This gets the tool #s in magazine pot.

☐Reference

GetMGNPotEx(), SetMGNPot()

☐Designation

2.15.10 IEZNcATC2::SetMGNPotEx — Setting tool # of the ATC extended magazine pot

☐Calling procedure (Custom interface)

```
HRESULT        SetMGNPotEx(
                 LONG lIndex,          //  (I)  Magazine pot #
                 LONG lToolNo,         //  (I)  Tool #
                 LONG* plRet           //  (O)  Error code
                 )
```

☐Calling procedure (Automation interface)

```
                 Tool_SetMGNPotEx(
                 lIndex As LONG        //  (I)  Magazine pot #
                 lToolNo As LONG       //  (I)  Tool #
                 ) As LONG             //  (O)  Error code
```

☐Argument

lIndex: Set magazine pot #
Value: 1 to 80

lToolNo: Set tool #
Value: 1 to 80

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_WRITE_WRITE: Impossible to write the data

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This sets the tool #s in magazine pot.

☐Reference

SetMGNPotEx(), GetMGNPot()

☐Designation

2.15.11 IEZNcATC2::GetMGNAux — Getting the ATC user PLC interface

☐Calling procedure (Custom interface)

```
HRESULT        GetMGNAux(
                   LONG* plData,          //  (O)  Data
                   LONG* plRet            //  (O)  Error code
                   )
```

☐Calling procedure (Automation interface)

```
               Tool_ GetMGNAux(
                   plData, As LONG*       //  (O)  Data
                   ) As LONG              //  (O)  Error code
```

☐Argument

plData: Returns the data for sequence processing of user PLC

Model	Value
M6x5M	0 to 65535
Magic64	0 to 99

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_READ_READ: Impossible to read the data

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This returns the data for sequence processing of user PLC.

☐Reference

☐Designation

2.15.12 IEZNcATC2::SetMGNAux — Setting the ATC user PLC interface

☐Calling procedure (Custom interface)

```
HRESULT        SetMGNAux(
                   LONG lData,          //  (I)  Data
                   LONG* plRet          //  (O)  Error code
                   )
```

☐Calling procedure (Automation interface)

```
               Tool_SetMGNAux(
                   lData, As LONG       //  (I)  Data
                   ) As LONG            //  (O)  Error code
```

☐Argument

lData: Set data for sequence processing of user PLC

Model	Value
M6x5M	0 to 65535
Magic64	0 to 99

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_READ_READ: Impossible to read the data

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This sets the data for sequence processing of user PLC.

☐Reference

☐Designation

2.16 IEZNcParameter2 Interface

Magic64 Limited	M6x5M	M6x5L	C64	C70	CNC700

2.16.1 IEZNcParameter2::GetParameterData — Getting the parameters

☐Calling procedure (Custom interface)

```
HRESULT        GetParameterData(
                 LONG lGroup,                  //  (I)   Group #
                 LONG lItem,                   //  (I)   Head item #
                 LONG lSize,                   //  (I)   The number of items
                 LONG lAxis,                   //  (I)   Axis designation
                 LPOLESTR* lppwszValue,        //  (O)   Parameter value character string array
                 LONG* plRet                   //  (O)   Error code
                 )
```

☐Calling procedure (Automation interface)

```
                 Parameter_ GetParameterData (
                 lGroup As LONG                //  (I)   Group #
                 lItem As LONG                 //  (I)   Head item #
                 lSize As LONG                 //  (I)   The number of items
                 lAxis As LONG                 //  (I)   Axis designation
                 pvValue As VARIANT*           //  (O)   Parameter value character string array
                 )As LONG                      //  (O)   Error code
```

☐ **Argument**

lGroup: Set group # of parameter

Model	Support
M6x5M	Supported (Must be set)
M6x5L	Not supported
MELDASMAGIC64	Not supported
C64	Not supported
CNC C70	Not supported
CNC700M/L	Not supported

In the case of **M6x5M**, group # is the first 2 digits of parameter # (6 digits) in the parameter manual (BNP-B2238*).

lItem: Set the head item # of parameter. This setting cannot be omitted. The parameter #s in the parameter manual will be item #s.

Model	Parameter manual	Item # so designate
M6x5M	BNP-B2238*	Lower 4 digits of 6 digit parameter #
M6x5L	BNP-B2233*	4 digit parameter #
MELDASMAGIC64	BNP-B2201*	4 digit parameter #
MELDAS C6/C64	BNP-B2267*	Parameter # up to 5 digits
CNC C70	IB-1500264	Parameter #
CNC 700M/L	IB-1500123	Parameter #

lSize: Set the number of items of parameter. The data ranges from 1.

lAxis: Set axis whose parameters you wish to get. In the case of parameters that don't depend on axis, this argument is invalid

lppwszValue: Gets the parameter value as **UNICODE** character string array. Memory area for character strings is saved inside of EZSocket, however, it is required to save pointers (for lSize) to character strings on the client side. If your client is a VC++ client, it is necessary to release the memory area explicitly by using **CoTaskMemFree()**.

pvValue:Refer to the explanation of *lppwszValue*.

plRet: This returns error code. (When using automation interface, returns return value instead.)
S_OK : Normal termination
EZNC_DATA_READ_ADDR: Part system designation or the axis designation is illegal
EZNC_PARAM_FILENOTEXIST: Parameter information file doesn't exist
EZNC_DATA_NOT_EXIST: Data doesn't exist

☐ **Return value**

Return value	Details
S_OK	Normal termination
S_FALSE	Communication failure

☐ **Functions**

This gets parameter.
If NC System Disk's version of **Magic64** (ISA NC card) is A*, machine error compensation parameter cannot be got. The parameter can be got by Ver. B0 and later.
(Note 1) Not supported with **Magic64** (PCI NC card). ISA NC card limits the function. Contact MITSUBISHI before using the card.

(Note 2) Mitsubishi CNC C70 Series doesn't support the automation interface.

☐ **Reference Designation**

SetParameterData()

2.16.2 IEZNcParameter2::SetParameterData — Setting parameters

☐Calling procedure (Custom interface)

```
HRESULT        GetParameterData(
                 LONG lGroup,                  //  (I)   Group #
                 LONG lItem,                   //  (I)   Head item #
                 LONG lSize,                   //  (I)   The number of items
                 LONG lAxis,                   //  (I)   Axis designation
                 LPCOLESTR* lppcwszValue,      //  (I)   Parameter value character string array
                 LONG* plRet                   //  (O)   Error code
                 )
```

☐Calling procedure (Automation interface)

```
                 Parameter_ GetParameterData (
                 lGroup As LONG                //  (I)   Group #
                 lItem As LONG                 //  (I)   Head item #
                 lSize As LONG                 //  (I)   The number of items
                 lAxis As LONG                 //  (I)   Axis designation
                 vValue As VARIANT             //  (I)   Parameter value character string array
                 )As LONG                      //  (O)   Error code
```

☐Argument

lGroup: Set group # of parameter

Model	Support
M6x5M	Supported (Must be set)
M6x5L	Not supported
MELDASMAGIC64	Not supported
MELDAS C6/C64	Not supported
CNC C70	Not supported
CNC 700M/L	Not supported

In the case of **M6x5M**, group # is the first 2 digits of parameter # (6 digits) in the parameter manual (BNP-B2238*).

lItem: Set the head item # of parameter. This setting cannot be omitted. The parameter #s in the parameter manual will be item #s.

Model	Parameter manual	Item # so designate
M6x5M	BNP-B2238*	Lower 4 digits of 6 digit parameter #
M6x5L	BNP-B2233*	4 digit parameter #
MELDASMAGIC64	BNP-B2201*	4 digit parameter #
MELDAS C6/C64	BNP-B2267*	Parameter # up to 5 digits
CNC C70	IB-1500264	Parameter #
CNC 700M/L	IB-1500123	Parameter #

lSize: Set the number of items of parameter. The data ranges from 1.

lAxis: Set axis whose parameters you wish to get. In the case of parameters that don't depend on axis, this argument is invalid

lppcwszValue: Set the parameter value as **UNICODE** character string array
vValue: Refer to the explanation of *lppcwszValue*.

plRet: This returns error code. (When using automation interface, returns return value instead.)
S_OK : Normal termination
EZNC_DATA_WRITE_ADDR: Part system designation or the axis designation is illegal
EZNC_PARAM_FILENOTEXIST: Parameter information file doesn't exist

☐**Return value**	Return value	Details
	S_OK	Normal termination
	S_FALSE	Communication failure
☐**Functions**	This sets parameter. To set machine parameters, the NC must be in the machine parameter setting mode. For how to set the machine parameter setting mode, refer to the Setup Manual for each NC system. If NC System Disk's version of **Magic64** (ISA NC card) is A*, machine error compensation parameter cannot be got. The parameter can be got by Ver. B0 and later. **(Note 1)** Not supported with **Magic64** (PCI NC card). ISA NC card limits the function. Contact MITSUBISHI before using the card. **(Note 2)** Mitsubishi CNC C70 Series doesn't support the automation interface.	
☐**Reference**	**GetParameterData()**	
☐**Designation**		

2.17 IEZNcOperation Interface

Magic64	M6x5M	M6x5L	C64	C70	CNC700

2.17.1 IEZNcOperation::Search — Search

☐Calling procedure (Custom interface)

```
HRESULT        Search(
                 LPCOLESTR                  //  (I)  Program file name
                 lpcwszSelectProgram,       //  (I)  Sequence #
                 LONG lSequenceNo,          //  (I)  Block #
                 LONG lBlockNo,             //  (O)  Error code
                 LONG* plRet
                 )
```

☐Calling procedure (Automation interface)

```
                 Operation_Search(
                 lpcwszSelectProgram As STRING   //  (I)  Program file name
                 lSequenceNo As LONG             //  (I)  Sequence #
                 lBlockNo As LONG*               //  (I)  Block #
                 )As LONG                        //  (O)  Error code
```

☐Argument

lpcwszSelectProgram: Set program file name for operation search as **UNICODE** character strings

lSequenceNo: Set sequence # to search

lBlockNo: Set block # to search

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK : Normal termination
EZNC_OPE_SELECTPRG_ADDR: Part system designation is illegal
EZNC_OPE_SELECTPRG_FILESYSTEM: File system is abnormal
EZNC_OPE_SELECTPRG_NOPRG: No program file
EZNC_OPE_SELECTPRG_PRGFORMAT: Program file name format is illegal
EZNC_DATA_SELECTPRG_RUNNING: Program is running

☐Return value

Return value	Details
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This executes operation search.
The name of the program file to start running is designated with *IpwcszSelectProgram*.
The program files that can be designated are in \PRG\USER\ directory of NC card or in M?\IC1\ directory of CF card on NC.
Note that the drive name and directory path are not required for the file name in \PRG\USER\ directory of NC card.

Below is an example of the file designation as character strings.

Other than **CNC700**------- "Program #. PRG" **(Example)** "1000.PRG"
CNC700 ------- "Program file name" **(Example)** "1000"
CNC700 CF card ---- "M?:\IC1\Program file name" **(Example)** "M01:\IC1\1000"

It is also possible to designate the sequence number and block number to start running by *lsequenceNo*. and *lBlockNo*. To run the program from the beginning, set "0" for both *lSequenceNo* and *lBlockNo*.

(Note) Mitsubishi CNC C70 Series doesn't support the automation interface.

☐Reference **IEZNcProgram::CurrentBlockRead()**

☐Designation System

2.17.2 IEZNcOperation::Run — Starting PLC program

☐**Calling procedure (Custom interface)**

```
HRESULT        Run(
                 LONG* plRet                    //    (O)    Error code
                 )
```

☐**Calling procedure (Automation interface)**

```
                 Operation_Run( ) As LONG       //    (O)    Error code
```

☐**Argument**

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK : Normal termination
EZNC_OPE_ACTPLC_ADDR: NC card is illegal

☐**Return value**

Return value	Details
S_OK	Normal termination
S_FALSE	Communication failure

☐**Functions**

This starts running PLC program.

(Note) Mitsubishi CNC C70 Series doesn't support the automation interface.

☐**Reference**

Stop()

☐**Designation**

2.17.3 IEZNcOperation::Stop — Stopping PLC program

☐Calling procedure (Custom interface)

```
HRESULT        Stop(
                 LONG* plRet                    //    (O)    Error code
               )
```

☐Calling procedure (Automation interface)

```
                 Operation_Stop( ) As LONG      //    (O)    Error code
```

☐Argument

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK : Normal termination
EZNC_OPE_ACTPLC_ADDR: NC card is illegal

☐Return value

Return value	Details
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This stops running PLC program.

(Note) Mitsubishi CNC C70 Series doesn't support the automation interface.

☐Reference

Run()

☐Designation

2.18 IEZNcDevice Interface

Magic64	M6x5M	M6x5L	C64	CNC700

2.18.1 IEZNcDevice::SetDevice — Setting device

☐Calling procedure (Custom interface)

```
HRESULT        SetDevice(
                 DWORD dwLength,                  //  (I)  The number of device points
                 LPCOLESTR*                       //  (I)  Device character string
                 lppcwszDevice,                   //  (I)  Data type
                 LPDWORD lpdwDataType,            //  (I)  Device value array
                 LPDWORD lpdwValue,               //  (O)  Error code
                 LONG* plRet
                 )
```

☐Calling procedure (Automation interface)

```
                 Device_SetDevice(
                 vDevice As VARIANT,              //  (I)  Device character string
                 vDataType As VARIANT,            //  (I)  Data type
                 vValue As VARIANT                //  (I)  Device value array
                 ) As LONG                        //  (O)  Error code
```

☐Argument

dwLength: Set the number of device points. The maximum number is 1k points.

lppcwszDevice: Specify the array of target device character string to be set. Set the device character string in **UNICODE**. If the data type is the word type (double word), set the device character string in multiple of 16 (32).

lpdwDataType: Set data type of each device as array

Value	Meaning	Unit in Table 2-8.
EZNC_PLC_BIT	Bit type	1 bit
EZNC_PLC_BYTE	Byte type	8 bit
EZNC_PLC_WORD	Word type	16 bit
EZNC_PLC_DWORD	Double word type	32 bit

lpdwValue: Set the array to which you set device value. In reading, set a dummy value having the same number of arrays as device character string.

Automation argument:
vDevice: Set array of device character string as **VARIANT**. Set the device character string in **UNICODE**. If the data type is the word type (double word), set the device character string in multiple of 16 (32).
vDataType: Set array of the data type of the device value as **VARIANT**.

Value	Meaning	Unit in Table 2-8.
EZNC_PLC_BIT	Bit type	1 bit
EZNC_PLC_BYTE	Byte type	8 bit
EZNC_PLC_WORD	Word type	16 bit
EZNC_PLC_DWORD	Double word type	32 bit

☐ **Argument**

vValue: Set array of the device value as **VARIANT**. In reading, set a dummy value having the same number of arrays as device character string.

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK : Normal termination
EZNC_DATA_READ_DATATYPE: Data type illegal

☐ **Return value**

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐ **Functions**

This decides the device used by user PLC. Device setting is all done by single shot. In the case of single shot, the previous setting may return by the PLC's one cycle. If you set again, the previous setting is invalidated.

☐ **Reference**

ReadDevice(), WriteDevice(), DeleteDeviceAll()

☐ **Designation**

Table 2-8 List of devices available

Device	Name	Unit	Model				
			Magic64	C64	M6x5L	M6x5M	CNC700M / L
B	Counter (Fixed counter)	1 bit	B0 to B103（104 points）	-	-	-	-
		1 bit/16 bit/32 bit	-	-	-	-	B0 to B1FFF(8192 points)
CI	Counter coil	1 bit	CO to C23（24 points）	CO to C23（24 points）	CI0 to CI127(128 points)	CI0 to CI127(128 points)	-
		1 bit/16 bit/32 bit	-	-	-	-	C0 to C255(256 points)
CO	Counter contact	1 bit	-	C24 to C127（104 points）	CO0 to CO127(128 points)	CO0 to CO127(128 points)	-
CS	Counter setting value *2	16 bit	-	-	CS0 to CS127(128 points)	CS0 to CS127(128 points)	-
CA	Counter current value *2	16 bit	-	-	CA0 to CA127(128 points)	CA0 to CA127(128 points)	-
D	Data register	16 bit/32 bit	D0 to D1023（1024 points）	D0 to D1023（1024 points）	D0 to D1023（1024 points）	D0 to D1023（1024 points）	D0 to D2047(2048 points)
E	Special relay	1 bit	EO to E127（128 points）	-	-	-	-
		1 bit/16 bit/32 bit	-	-	-	-	-
F	Alarm message interface, temporary memory	1 bit	F0 to F127（128 points）	F0 to F127（128 points）	F0 to F127（128 points）	F0 to F255（256 points）	-
		1 bit/16 bit/32 bit	-	-	-	-	F0 to F1023(1024 points)
G	Temporary memory	1 bit	G0 to G3071(3072 points)	-	-	-	-
		1 bit/16 bit/32 bit	-	-	-	-	-
I	I device	1 bit	I0 to I3FF(1024 points)	-	-	-	-
		1 bit/16 bit/32 bit	-	-	-	-	-
J	J device	1 bit	J0 to J63F(1600 points)	-	-	-	-
		1 bit/16 bit/32 bit	-	-	-	-	-
L	Latch relay (Backup memory)	1 bit	L0 to L255（256 points）	L0 to L255（256 points）	L0 to L255（256 points）	L0 to L255（256 points）	-
		1 bit/16 bit/32 bit	-	-	-	-	L0 to L511(512 points)
M	Temporary memory		M0 to M5119（5120 points）	M0 to M8191(8192 points)	M0 to M8191（8192 points）	M0 to M8191（8192 points）	-
		1 bit/16 bit/32 bit	-	-	-	-	M0 to M10239(10240 points)

Device	Name	Unit	Model				
			Magic64	C64	M6x5L	M6x5M	CNC700M / L
Q	Q device		Q0 to Q151（152 points）	-	-	-	-
		1 bit/16 bit/32 bit	-	-	-	-	-
R	File register *1	16 bit/32 bit	R0 to R8191（8192 points）	R0 to R8191（8192 points）	R0 to R8191（8192 points）	R0 to R8191（8192 points）	R0 to R13311(13312 points)
S	S device	1 bit	S0 to S13F（320 points）	-	-	-	-
		1 bit/16 bit/32 bit	-	-	-	-	-
SM	Special relay *1	1 bit	-	SM0 to SM127(128 points)	SM0 to SM127（128 points）	SM0 to SM127（128 points）	-
		1 bit/16 bit/32 bit	-	-	-	-	SM0 to SM1023(1024 points)
SB	Special relay for link	1 bit	-	-	-	-	-
		1 bit/16ㄷ bit/32 bit	-	-	-	-	SB0 to SB1FF(512 points)
SD	Special register	16 bit/32 bit	-	-	-	-	SD0 to SD1023(1024 points)
ST		1 bit/16 bit/32 bit	-	-	-	-	ST0 to ST63(64 points)
SW	Special register for link	16 bit/32 bit	-	-	-	-	SW0 to SW1FF(512 points)
TI	10ms-incremental timer coil	1 bit	T0 to T15（16 points）	T0 to T15（16 points）	TI0 to TI55(56 points)	TI0 to TI55(56 points)	T0 to T703(704 points)
	100ms-incremental timer coil	1 bit	T16 to T95（80 points）	T16 to T95（80 points）	TI56 to TI231(176 points)	TI56 to TI231(176 points)	-
	100ms-multiple timer coil	1 bit	T96 to T103（8 points）	T96 to T103（8 points）	TI232 to TI255(24 points)	TI232 to T255(24 points)	-
	10ms-incremental timer coil	1 bit	-	T104 to T143（40 points）	-	-	-
	100ms-incremental timer coil	1 bit	-	T144 to T239（96 points）	-	-	-
	100ms-multiple timer coil	1 bit	-	T240 to T255（16 points）	-	-	-
TO	10ms-incremental timer contact	1 bit	-	-	TO0 to TO55(56 points)	TO0 to TO55(56 points)	-
	100ms-incremental timer contact	1 bit	-	-	TO56 to TO231(176 points)	TO56 to TO231176 points)	-
	100ms-multiple timer contact	1 bit	-	-	TO232 to TO255(24 points)	TO232 to TO255(24 points)	-

Device	Name	Unit	Model				
			Magic64	C64	M6x5L	M6x5M	CNC700M / L
TS	10ms-incremental timer setting value *2	16 bit	-	-	TS0 to TS 55(56 points)	TS0 to TS55(56 points)	-
	100ms-incremental timer setting value *2	16 bit	-	-	TS56 to TS 231(176 points)	TS56 to TS231(176 points)	-
	100ms-multiple timer setting value *2	16 bit	-	-	TS 232 to TS255(24 points)	TS232 to TS255(24 points)	-
TA	10ms-incremental timer current value *2	16 bit	-	-	TA 0 to TA 55(56 points)	TA0 to TA55(56 points)	-
	100ms-incremental timer current value *2	16 bit	-	-	TA56 to TA231(176 points)	TA56 to TA231(176 points)	-
	100ms-multiple timer current value *2	16 bit	-	-	TA232 to TA255(24 points)	TA232 to TA255(24 points)	-
U	For input signal (to PLC) 2 system *1	1 bit	U0 to U17F（384 points）	-	-	-	-
		1 bit/16 bit/32 bit	-	-	-	-	-
V	V device	1 bit	-	V0 to V255(256 points)	-	-	-
		1 bit/16 bit/32 bit	-	-	-	-	V0 to V255(256 points)
W	For output signal (to PLC) 2 system *1	1 bit	W0 to W1FF（512 points）	-	-	-	-
		1 bit/32 bit	-	-	-	-	W0 to W1FFF(8192 points)
X	Input signal to PLC *1	1 bit/16 bit	X0 to X4BF（1216 points）	X0 to XAFF（2816 points）	X0 to XABF（2752 points）	X0 to XAFF（2816 points）	-
		1 bit/16 bit/32 bit	-	-	-	-	X0 to X1FFF(8192 points)
Y	Output signal to PLC *1	1 bit/16 bit	Y0 to Y53F（1344 points）	Y0 to YE7F（3712 points）	Y0 to YDFF（3584 points）	Y0 to YDFF（3584 points）	-
		1 bit/16 bit/32 bit	-	-	-	-	Y0 to Y1FFF(8192 points)
Z	Address index	16 bit	-	-	-	-	-

*1: The usage of this device is fixed. Even if the device is undefined and blank, do not use it unless it corresponds with the input/output signals between the machine.
*2: This device is dedicated for reading.

2.18.2 IEZNcDevice::DeleteDeviceAll — Deleting all device setting

☐**Calling procedure (Custom interface)**

```
HRESULT        DeleteDeviceAll(
                   LONG* plRet          //    (O)    Error code
                   )
```

☐**Calling procedure (Automation interface)**

```
               Device_DeleteAll( ) As LONG    //    (O)    Error code
```

☐**Argument**

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK : Normal termination

☐**Return value**

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐**Functions**

This deletes all the data set by **SetDevice**().

☐**Reference**

SetDevice()

☐**Designation**

2.18.3 IEZNcDevice::ReadDevice — Reading the device

☐**Calling procedure (Custom interface)**

```
HRESULT        ReadDevice(
                   LPDWORD lpdwLength,   //  (O)  The number of device points that were read
                   LPDWORD*              //  (O)  Array of device value that was read
                   lppdwValue,           //  (O)  Error code
                   LONG* plRet
                   )
```

☐**Calling procedure (Automation interface)**

```
               Device_Read(
                   pvValue As VARIANT*   //  (O)  Array of device value
                   )As LONG              //  (O)  Error code
```

☐**Argument**

lpdwLength: Returns the number of devices that were read

lppdwValue: Returns the array in which device value is stored. The array is saved inside of EZSocket, so the client is required to release the memory area explicitly by using **CoTaskMemFree()**.

Automation argument:
pvValue: Returns the array of device value as **VARIANT**.

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_READ_DATATYPE: Data type illegal
EZNC_DATA_READ_READ: Impossible to read the data
EZNC_DATA_READ_WRITEONLY: Writing-dedicated data

☐**Return value**

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐**Functions**

For batch-reading of the devices set by **SetDevice()**.

☐**Reference**

SetDevice(),
WriteDevice()

☐**Designation**

2.18.4 IEZNcDevice::WriteDevice — Writing device

☐**Calling procedure (Custom interface)**

```
HRESULT        WriteDevice(
                   LONG* plRet          //    (O)    Error code
               )
```

☐**Calling procedure (Automation interface)**

```
               Device_Write()As LONG    //    (O)    Error code
```

☐**Argument**

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_WRITE_DATATYPE: Data type illegal
EZNC_DATA_ WRITE_WRITE: Impossible to write the data
EZNC_DATA_ WRITE_READONLY: Reading-dedicated data

☐**Return value**

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐**Functions**

For batch-writing of the devices set by **SetDevice()**.

☐**Reference**

SetDevice(), ReadDevice()

☐**Designation**

2.19 IEZNcGeneric2 Interface

Magic64	M6x5M	M6x5L	C64	CNC700

2.19.1 IEZNcGeneric2::ReadData — Reading the generic data

☐Calling procedure (Custom interface)

```
HRESULT        ReadData(
                 LONG lAxisNo,                  //  (I)    Axis designation
                 LONG lSectionNum,              //  (I)    Class #
                 LONG lSubSectionNum,           //  (I)    Sub-class #
                 VARIANT* pvReadData,           //  (I/O)  Data storage area
                 LONG* plRet                    //  (O)    Error code
                 )
```

☐Calling procedure (Automation interface)

```
                 Generic_ReadData(
                 lAxisNo As LONG                //  (I)    Axis designation
                 lSectionNum As LONG            //  (I)    Class #
                 lSubSectionNum As LONG         //  (I)    Sub-class #
                 pvReadData As VARIANT*         //  (I/O)  Data storage area
                 ) As LONG                      //  (O)    Error code
```

☐Argument

lAxisNo: Set axis # ("1" or later)
In the case of M6x5M: Set with axis # in the part system ("1" or later) or variable #.
Set sub #s in the custom application programming interface manual BNP-B3950-023.

lSectionNum: Set class #

lSubSectionNum: Set sub-class #

pvReadData: Set pointer to the data storage area
Available data type (Set in pvReadData.vt.)
EZ_T_CHAR: 1 byte integer
EZ_T_SHORT: 2 byte integer
EZ_T_LONG: 4 byte integer
EZ_T_DOUBLE: 8 type real number
EZ_T_STR: Character string
Data area
EZ_T_CHAR: pvReadData.bVal
EZ_T_SHORT: pvReadData.iVal
EZ_T_LONG: pvReadData.lVal
EZ_T_DOUBLE: pvReadData.dblVal
EZ_T_STR: pvReadData.bstrVal

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_READ_ADDR: Part system/axis designation is illegal
EZNC_DATA_READ_DATASIZE: Too much data for the buffer prepared by the application
EZNC_DATA_READ_DATATYPE: Data type illegal
EZNC_DATA_READ_READ: Impossible to read the data
EZNC_DATA_READ_SECT: Class # illegal
EZNC_DATA_READ_SUBSECT: Sub-class # illegal
EZNC_DATA_READ_WRITEONLY: Writing-dedicated data

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐ **Functions**	This reads data designated with the class # and sub-class # of the designated axis. In the case of EZ_T_STR, make sure to allocate the memory for 256 character strings.
☐ **Reference**	**WriteData()**
☐ **Designation**	System, Axis

2.19.2 IEZNcGeneric2::WriteData — Writing generic data

☐Calling procedure (Custom interface)

```
HRESULT        WriteData(
                 LONG lAxisNo,              //  (I)  Axis designation
                 LONG lSectionNum,          //  (I)  Class #
                 LONG lSubSectionNum,       //  (I)  Sub-class #
                 VARIANT vWriteData,        //  (I)  Data to write
                 LONG* plRet                //  (O)  Error code
                 )
```

☐Calling procedure (Automation interface)

```
                 Generic_WriteData(
                 lAxisNo As LONG            //  (I)  Axis designation
                 lSectionNum As LONG        //  (I)  Class #
                 lSubSectionNum As LONG     //  (I)  Sub-class #
                 vWriteData As VARIANT      //  (I)  Data to write
                 ) As LONG                  //  (O)  Error code
```

☐Argument

lAxisNo: Set axis # (“1” or later)
In the case of M6x5M: Set with axis # in the part system ("1" or later) or variable #.
Set sub #s in the custom application programming interface manual BNP-B3950-023.

lSectionNum: Set class #

lSubSectionNum: Set sub-class #

vWriteData: Set data to write
Available data type (Set in vWriteData.vt.)
EZ_T_CHAR: 1 byte integer
EZ_T_SHORT: 2 byte integer
EZ_T_LONG: 4 byte integer
EZ_T_DOUBLE: 8 type real number
EZ_T_STR: Character string
Data area
EZ_T_CHAR: vWriteData.bVal
EZ_T_SHORT: vWriteData.iVal
EZ_T_LONG: vWriteData.lVal
EZ_T_DOUBLE: vWriteData.dblVal
EZ_T_STR: vWriteData.bstrVal

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_WRITE_ADDR: Part system/axis designation is illegal
EZNC_DATA_ WRITE _DATASIZE: Too much data for the buffer prepared by the application
EZNC_DATA_ WRITE _DATATYPE: Data type illegal
EZNC_DATA_ WRITE _ WRITE: Impossible to write the data
EZNC_DATA_ WRITE _SECT: Class # illegal
EZNC_DATA_ WRITE _SUBSECT: Sub-class # illegal
EZNC_DATA_ WRITE _READONLY: Reading-dedicated data

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions This writes data designated with the class # and sub-class # of the designated axis.

☐Reference **ReadData()**

☐Designation System, Axis

2.19.3 IEZNcGeneric2::SetData — Setting data

☐Calling procedure (Custom interface)

```
HRESULT     SetData(
            DWORD dwLength,                  //  (I)  The number of data points to set
            EZNCST_GENDATA* pstGenData,      //  (I)  Structure array of data to set
            LONG* plRet                      //  (O)  Error code
            )
```

☐Calling procedure (Automation interface)

```
            Generic_SetData(
            vAxisNo As VARIANT               //  (I)  Axis # array
            vSectionNum As VARIANT           //  (I)  Class # array
            vSubSectionNum As VARIANT        //  (I)  Sub-class # array
            vDataType As VARIANT             //  (I)  Data type # array
            vSrcData As VARIANT              //  (I)  Data value character string array
            )As LONG                         //  (O)  Error code
```

☐Argument

lLength: Set the number of data points. The maximum number is 1k points.

pstGetData: Set data you wish to set. This is a pointer to point the EZNCST_GENDATA structure.

lAxisNo: Set axis # (In the case of M6x5M, sub-class #)
lSectionNum: Set class # (In the case of M6x5M, group #)
lSubSectionNum: Set sub/class # (In the case of M6x5M, item #)
lDataType: Set data type #
lpcwszSrcData: Set data value as character string (Only in the case of writing (WriteBlockData))

Automation argument:
vAxisNo: Refer to the explanation of *lAxisNo.*
vSectionNum: Refer to the explanation of *lSectionNum.*
vSubSectionNum: Refer to the explanation of *lSubSectionNum.*
vDataType: Refer to the explanation of *lDataType.*
Available data type
EZ_T_CHAR: 1 byte integer
EZ_T_SHORT: 2 byte integer
EZ_T_LONG: 4 byte integer
EZ_T_DLONG: 8 byte integer（CNC C70 only）
EZ_T_DOUBLE: 8 type real number
EZ_T_STR: Character string
vSrcData: Refer to the explanation of *lpcwszSrcData.* A string possible to set is up to 256 characters long (including a NULL).

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK : Normal termination
EZNC_DATA_READ_DATATYPE: Data type illegal
EZNC_DATA_READ_ADDR: Axis designation is illegal

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This sets data as array. If you change the settings, the latest setting is validated.

(Note 1) The lDataType, EZ_T_DLONG is enabled for CNC C70 only. When used in a model other than CNC C70, EZ_ERR_DATA_TYPE is returned to plRet.
(Note 2) Mitsubishi CNC C70 Series doesn't support the automation interface.

☐ **Reference** **ReadBlockData(), WriteBlockData(), DeleteDataAll()**

☐ **Designation** Axis

2.19.4 IEZNcGeneric2::DeleteDataAll — Deleting all data setting

☐Calling procedure (Custom interface)

```
HRESULT        DeleteDataAll(
                   LONG* plRet          //    (O)    Error code
                   )
```

☐Calling procedure (Automation interface)

```
               Generic_DeleteAll()As LONG   //    (O)    Error code
```

☐Argument

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK : Normal termination

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This deletes all the data set by **SetData()**.
(Note) Mitsubishi CNC C70 Series doesn't support the automation interface.

☐Reference

SetData()

☐Designation

2.19.5 IEZNcGeneric2::ReadBlockData — Batch-reading of data

☐Calling procedure (Custom interface)

```
HRESULT        ReadBlockData(
                   LPDWORD lpdwLength,          //  (O) The number of data points that was read
                   LPOLESTR** lpppwszData,      //  (O) Data value character string array
                   LONG* plRet                  //  (O) Error code
                   )
```

☐Calling procedure (Automation interface)

```
                Generic_ ReadBlockData (
                   pvData As VARIANT*           //  (O) Data value character string
                   )As LONG                     //  (O) Error code
```

☐Argument

lpdwLength: Returns the number of data points to set

lpppwszData: Returns the data value as **UNICODE** character string array. The data value's array and character string area are saved inside of EZSocket, so the client is required to release the memory area explicitly by using **CoTaskMemFree()**.

Example)
```
DWORD dwLength = 0 ;
LPOLESTR* lppwszData = NULL ;
LONG lRet = 0 ;

HRESULT hr = pIEZNcGeneric->ReadBlockData( &dwLength, &lppwszData, &lRet ) ;
if ( lppwszData != NULL ) {
    // Release the array's character string area.
    for ( DWORD dwCnt = 0 ; dwCnt < dwLength ; dwCnt++ ) {
        if ( lppwszData[dwCnt] != NULL ) {
            CoTaskMemFree( lppwszData[dwCnt] ) ;
            lppwszData[dwCnt] = NULL ;
        }
    }
    // Release the array.
    CoTaskMemFree( lppwszData ) ;
    lppwszData = NULL ;
}
```

Automation argument:
pvData: Returns the data value character string as **VARIANT**.

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK : Normal termination
EZNC_DATA_READ_DATATYPE: Data type illegal
EZNC_DATA_READ_READ: Impossible to read the data
EZNC_DATA_READ_WRITEONLY: Writing-dedicated data

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

For batch-reading of the data set by **SetData()**.
(Note) Mitsubishi CNC C70 Series doesn't support the automation interface.

☐Reference

SetData(), WriteBlockData()

☐Designation

2.19.6 IEZNcGeneric2::WriteBlockData — Batch-writing of data

☐Calling procedure (Custom interface)

```
HRESULT        WriteBlockData(
                    LONG* plRet                    //    (O)    Error code
                    )
```

☐Calling procedure (Automation interface)

```
               Generic_WriteBlockData ()As LONG    //    (O)    Error code
```

☐Argument

plRet: Returns an error code. (When using automation interface, it returns a return value instead.)
S_OK: Normal termination
EZNC_DATA_WRITE_DATATYPE: Data type illegal
EZNC_DATA_ WRITE _ WRITE: Impossible to write the data
EZNC_DATA_ WRITE _READONLY: Reading-dedicated data

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

For batch-writing of the data set by **SetData()**.
(Note) Mitsubishi CNC C70 Series doesn't support the automation interface.

☐Reference

SetData(), ReadBlockData()

☐Designation

2.20 IEZNcSubFunction2 Interface

Magic64	M6x5M	M6x5L	C64	CNC700

2.20.1 IEZNcSubFunction2::ChangeInit — Initializing sub function

☐Calling procedure (Custom interface)

```
HRESULT        ChangeInit(
                 LONG lSystemType,      //  (I)  NC system type setting
                 LONG lReserve1,        //  (I)  Reservation 1
                 LONG lReserve2,        //  (I)  Reservation 2
                 LONG* plRet            //  (O)  Error code
                 )
```

☐Calling procedure (Automation interface)

```
                 ChangeInit(
                 lSystemType As LONG    //  (I)  NC system type setting
                 lReserve1 As LONG      //  (I)  Reservation 1
                 lReserve2 As LONG      //  (I)  Reservation 2
                 ) As LONG              //  (O)  Error code
```

☐Argument

lSystemType: Set the NC system's type

Value	Meaning
EZNC_SYS_MELDAS6x5M	Initialization performed with **M6x5M**
EZNC_SYS_MELDAS6x5L	Initialization performed with **M6x5L**
EZNC_SYS_MAGICBOARD64	Initialization performed with **Magic64**
EZNC_SYS_MELDASC6C64	Initialization performed with **C64**
EZNC_SYS_MELDAS700M	Initialization performed with **CNC700M**
EZNC_SYS_MELDAS700L	Initialization performed with **CNC700L**

lReserve1: Not used (Always set to 0.)
lReserve2: Not used (Always set to 0.)

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK : Normal termination
EZ_ERR_DATA_RANGE: Data range is illegal

☐Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐Functions

This initializes **IEZNcSubFunction**.

☐Reference

☐Designation

M6x5M M6x5L

2.20.2 IEZNcSubFunction2::GetToolLifeValueOfFile — Getting the tool life management data of tool life management file

☐Calling procedure (Custom interface)

```
HRESULT      GetToolLifeValueOfFile(
             LPCOLESTR lpcwszFileName,   //  (I)   File name with path
             DWORD dwHead,               //  (I)   Part system
             DWORD dwToolNo,             //  (I)   Tool #
             LPOLESTR** lpppwszData,     //  (O)   Life management data character string array
             LONG* plRet                 //  (O)   Error code
             )
```

☐Calling procedure (Automation interface)

```
             GetToolLifeValueOfFile (
             bstrFileName As STRING      //  (I)   File name with path
             lHeadAs LONG                //  (I)   Part system
             lToolNoAs LONG              //  (I)   Tool #
             pvData As VARIANT*          //  (O)   Life management data character string array
             ) As LONG                   //  (O)   Error code
```

☐Argument

lpcwszFileName: Set file name (including path) of tool life management file as **UNICODE** character string.
Set the file by absolute path as follows,
Drive name + ":" + \Directory name\File name

dwHead: Set part system

dwToolNo: Set tool # whose life management data you wish to get

lpppwszData: Returns the tool life management data value as **UNICODE** character string array. The array is saved inside of EZSocket, so the client is required to release the memory area explicitly by using **CoTaskMemFree()**.

lpppwszData	Kinds of tool life management data (Data range)		Remarks
	M6x5M	**M6x5L**	
0	Tool # (1 to 99999999)	Time-managed usage (0 to 995959)	
1	Status (According to the machine builder's spec)	Number-of-times-managed usage (0 to 9999)	
2	Type (000 to 223) *	Status A (0 to 2)	
3	Length offset (-/+ 1 to 99999.999)	Time-managed life (0 to 995959)	
4	Diameter offset (-/+ 1 to 99999.999)	Number-of-times-managed life (0 to 9999)	
5	Life (Time: 0 to 4000, the number of times used: 0 to 9999)	Status B (According to the machine builder's spec)	
6	Usage (Time: 0 to 4000, the number of times used: 0 to 9999)	-	
7	Aux (0 to 65535, according to the machine builder's spec)	-	
8	Length wear (Reserved: 0)	-	
9	Diameter wear (Reserved: 0)	-	
10	Group (1 to 99999999)	-	

☐ **Argument**

Automation argument:
bstrFileName: Refer to the explanation of *lpcwszFileName.*
lHead: Refer to the explanation of *dwHead.*
lToolNo: Refer to the explanation of *dwToolNo.*
pvData: Returns the array of tool life management data value (**UNICODE** character string) as **VARIANT**.

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK : Normal termination
EZNC_FILE_OPEN_FILENOTEXIST: The file doesn't exist
EZNC_FILE_READFILE_READ: Impossible to read the data
EZNC_FILE_READFILE_ILLIGALFILE: The file is illegal
EZNC_DATA_NOT_EXIST: The file doesn't exist
EZ_ERR_MEMORY_ALLOC: Memory cannot be saved

☐ **Return value**

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐ **Functions**

This gets the designated tool #'s tool life management data from the tool life management file. The number of elements of character strings that return the life management data depends on the model.
*As for "Type" of tool life management data, refer to the "Tool life screen" chapter of "Mitsubishi Numerical Controller MELDAS 600M series Instruction manual".
Make sure to initialize **IEZNcSubFunction** by executing **ChangeInit()** before executing. If not, an error occurs at executing.

☐ **Reference**

OpenFile3(), CloseFile2(), ReadFile2(), WriteFile(), SetToolLifeValueOfFile()

☐ **Designation**

M6x5M M6x5L

2.20.3 IEZNcSubFunction2::SetToolLifeValueOfFile — Setting the tool life management data of tool life management file

☐Calling procedure (Custom interface)

```
HRESULT        SetToolLifeValueOfFile (
                 LPCOLESTR                  //  (I)  File name with path
                 lpcwszFileName,            //  (I)  Setting mode
                 DWORD dwMode,              //  (I)  Part system
                 DWORD dwHead,              //  (I)  Tool #
                 DWORD dwToolNo,            //  (I)  Life management data character string array
                 LPCOLESTR* lppcwszData,    //  (O)  Error code
                 LONG* plRet
                 )
```

☐Calling procedure (Automation interface)

```
                 SetToolLifeValueOfFile (
                 bstrFileName As STRING     //  (I)  File name with path
                 lModeAs LONG               //  (I)  Setting mode
                 lHeadAs LONG               //  (I)  Part system
                 lToolNo As LONG            //  (I)  Tool #
                 vData As VARIANT           //  (I)  Life management data character string array
                 ) As LONG                  //  (O)  Error code
```

☐Argument

lpcwszFileName: Set file name (including path) of tool life management file as **UNICODE** character string.
Set the file by absolute path as follows,
Drive name + ":" + \Directory name\File name
Set a file of the PC (client) side. (A file of the NC side cannot be set.)

dwMode: Set setting mode of the tool life management file

Value	Meaning
EZNC_FILE_CREATE	Setting a new tool life management file
EZNC_FILE_OPEN	Setting in an existing tool life management file

dwHead: Set part system

dwToolNo: Set tool # whose life management data you wish to get

☐ **Argument**

lpppwszData: Returns the tool life management data value of the designated kind as **UNICODE** character string array.

lppcwszData	Kinds of tool life management data (Data range)	
	M6x5M	**M6x5L**
0	Tool #	Time-managed usage (0 to 995959)
1	Status (According to the machine builder's spec)	The-number-of-times-managed usage (0 to 9999)
2	Type (000 to 223) *	Status A (0 to 2)
3	Length offset (-/+ 1 to 99999.999)	Time-managed life (0 to 995959)
4	Diameter offset (-/+ 1 to 99999.999)	The-number-of-times-managed life (0 to 9999)
5	Life (Time: 0 to 4000, the number of times used: 0 to 9999)	Status B (According to the machine builder's spec)
6	Usage (Time: 0 to 4000, the number of times used: 0 to 9999)	-
7	Aux (0 to 65535, according to the machine builder's spec)	-
8	Length wear (Reserved: 0.000)	-
9	Diameter wear (Reserved: 0.000)	-
10	Group	-

Automation argument:
bstrFileName: Refer to the explanation of *lpcwszFileName*.
lMode: Refer to the explanation of *dwMode.*
lHead: Refer to the explanation of *dwHead.*
lToolNo: Refer to the explanation of *dwToolNo.*
vData: Set the array of **UNICODE** character string of the specified kind of tool life management data value and substitute the array for vdata(**VARIANT**) to specify.

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK : Normal termination
EZNC_FILE_OPEN_FILENOTEXIST: The file doesn't exist
EZNC_FILE_OPEN_OPEN: Impossible to open the file
EZNC_FILE_WRITEFILE_ILLIGALFILE: The file is illegal
EZNC_FILE_WRITEFILE_WRITE: Impossible to write the data
EZ_ERR_NULLPTR: Argument is NULL pointer

☐ **Return value**

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐ **Functions**

This sets life management data in the designated tool life management file.
*As for "Type" of tool life management data, refer to the Instruction Manual for each NC system.

[E.g.] In the case of M6x5M

```
LPOLESTR* lppwszData;
lppwszData = new LPOLESTR[11];
lppwszData[0] =L"100";
lppwszData[1] =L"1";
lppwszData[2] =L"220";
lppwszData[3] =L"10.000";
lppwszData[4] =L"20.000";
lppwszData[5] =L"40.000";
lppwszData[6] =L"18.000";
lppwszData[7] =L"0";
lppwszData[8] =L"0.000";
lppwszData[9] =L"0.000";
lppwszData[10] =L"1";
hr = pIEZNcTool->SetToolLifeValueOfFile(L"C:\TEMP\TOOLLIFE.TLF",EZNC_FILE_OPEN,
1,100, (LPCOLESTR*)lppwszData, &lRet);
if( S_OK != hr ){
        wprintf(L"HRESULT Code = 0x%x, lRet Code = 0x%x\n", hr, lRet );
}
delete[ ] lppwszData;
```

☐ **Reference**

OpenFile3(), CloseFile2(), ReadFile2(), WriteFile(), GetToolLifeValueFile()

☐ **Designation**

M6x5M M6x5L C64

2.20.4 IEZNcSubFunction2::GetToolLifeValueOfFile2 Getting the tool life management data of tool life management file2

☐Calling procedure (Custom interface)

```
HRESULT        GetToolLifeValueOfFile2(
               LPCOLESTR lpcwszFileName,      //  (I)  File name with path
               DWORD dwGroup,                 //  (I)  Tool group
               DWORD dwHead,                  //  (I)  Part system
               DWORD dwToolNo,                //  (I)  Tool #
               LPOLESTR** lpppwszData,        //  (O)  Life management data character string array
               LONG* plRet                    //  (O)  Error code
               )
```

☐Calling procedure (Automation interface)

```
               GetToolLifeValueOfFile2(
               bstrFileName As STRING         //  (I)  File name with path
               lGroupAs LONG                  //  (I)  Tool group
               lHeadAs LONG                   //  (I)  Part system
               lToolNoAs LONG                 //  (I)  Tool #
               pvData As VARIANT*             //  (O)  Life management data character string array
               ) As LONG                      //  (O)  Error code
```

☐Argument

lpcwszFileName: Set file name (including path) of tool life management file as **UNICODE** character string.
Set the file by absolute path as follows,
Drive name + ":" + \Directory name\File name

dwGroup: Set tool group (Valid only with **C64**)

dwHead: Set part system

dwToolNo: Set tool # whose life management data you wish to get

☐ **Argument**

lpppwszData: Returns the tool life management data value as **UNICODE** character string array. The array is saved inside of EZSocket, so the client is required to release the memory area explicitly by using **CoTaskMemFree()**.

Model / *lpppwszData*	**M6x5M**	**M6x5L**	**C64**
0	Tool # (1 to 99999999)	Time-managed usage (0 to 995959)	Tool # (1 to 99999999)
1	Status (According to the machine builder's spec)	The-number-of-times-managed usage (0 to 9999)	Status (According to the machine builder's spec)
2	Type (000 to 223) *	Status A (0 to 2)	Type (000 to 223) *
3	Length offset (-/+ 1 to 99999.999)	Time-managed life (0 to 995959)	Length offset (-/+ 1 to 99999.999)
4	Diameter offset (-/+ 1 to 99999.999)	The-number-of-times-managed life (0 to 9999)	Diameter offset (-/+ 1 to 99999.999)
5	Life (Time: 0 to 4000, the number of times used: 0 to 9999)	Status B (According to the machine builder's spec)	Life (Time: 0 to 4000, the number of times used: 0 to 9999)
6	Usage (Time: 0 to 4000, the number of times used: 0 to 9999)	-	Usage (Time: 0 to 4000, the number of times used: 0 to 9999)
7	Aux (0 to 65535, according to the machine builder's spec)	-	Aux (0 to 65535, according to the machine builder's spec)
8	Length wear (Reserved: 0)	-	Length wear (Reserved: 0)
9	Diameter wear (Reserved: 0)	-	Diameter wear (Reserved: 0)
10	Group (1 to 99999999)	-	Group (1 to 99999999)

* As for "Type" of tool life management data, refer to the Instruction Manual for each NC system.

Automation argument:
bstrFileName: Refer to the explanation of *lpcwszFileName.*
lGroup: Refer to the explanation of *dwGroup.*
lHead: Refer to the explanation of *dwHead.*
lToolNo: Refer to the explanation of *dwToolNo*.
pvData: Returns the array of tool life management data value (**UNICODE** character string) as **VARIANT**.

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK: Normal termination
EZNC_FILE_OPEN_FILENOTEXIST: The file doesn't exist
EZNC_FILE_READFILE_READ: Impossible to read the data
EZNC_FILE_READFILE_ILLIGALFILE: The file is illegal
EZNC_DATA_NOT_EXIST: The file doesn't exist
EZ_ERR_MEMORY_ALLOC: Memory cannot be saved

☐ **Return value**

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐ **Functions**	This gets the designated tool #'s tool life management data from the tool life management file. The number of elements of character strings that return the life management data depends on the model. Make sure to initialize **IEZNcSubFunction** by executing **ChangeInit()** before executing. If not, an error occurs at executing.
☐ **Reference**	**OpenFile3(), CloseFile2(), ReadFile2(), WriteFile(), SetToolLifeValueOfFile2()**
☐ **Designation**	

M6x5M M6x5L C64

2.20.5 IEZNcSubFunction2::SetToolLifeValueOfFile2 Setting the tool life management data of tool life management file

☐Calling procedure (Custom interface)

```
HRESULT        SetToolLifeValueOfFile 2(
               LPCOLESTR lpcwszFileName,   //  (I)   File name with path
               DWORD dwMode,               //  (I)   Setting mode
               DWORD dwGroup,              //  (I)   Tool group
               DWORD dwHead,               //  (I)   Part system
               DWORD dwToolNo,             //  (I)   Tool #
               LPCOLESTR* lppcwszData,     //  (I)   Life management data character string array
               LONG* plRet                 //  (O)   Error code
               )
```

☐Calling procedure (Automation interface)

```
               SetToolLifeValueOfFile2(
               bstrFileName As STRING      //  (I)   File name with path
               lModeAs LONG                //  (I)   Setting mode
               lGroupAs LONG               //  (I)   Tool group
               lHeadAs LONG                //  (I)   Part system
               lToolNo As LONG             //  (I)   Tool #
               vData As VARIANT            //  (I)   Life management data character string array
               ) As LONG                   //  (O)   Error code
```

☐Argument

lpcwszFileName: Set file name (including path) of tool life management file as **UNICODE** character string.
Set the file by absolute path as follows,
Drive name + ":" + \Directory name\File name
Set a file of the PC (client) side. (A file of the NC side cannot be set.)

dwMode: Set setting mode of the tool life management file. Normally, create a file for each piece of data by **EZNC_FILE_CREATE**, and transfer them to the NC.

Value	Meaning
EZNC_FILE_CREATE	Setting a new tool life management file
EZNC_FILE_OPEN	Setting to an existing tool life management file

dwGroup: Set tool group

dwHead: Set part system

dwToolNo: Set tool # whose life management data you wish to get

lppcwszData: Returns the tool life management data value of the designated kind as **UNICODE** character string array.
Refer to the "lpppwszData" index table in "2.20.4IEZNcSubFunction2::GetToolLifeValueOfFile2".

Automation argument:
bstrFileName: Refer to the explanation of *lpcwszFileName.*
lMode: Refer to the explanation of *dwMode.*
lGroup: Refer to the explanation of *dwGroup.*
lHead: Refer to the explanation of *dwHead.*
lToolNo: Refer to the explanation of *dwToolNo*.
vData: Create tool life management data value of the specified kind in the array of **UNICODE** character string and substitute the data for vData(**VARIANT**) to specify.

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK: Normal termination
EZNC_FILE_OPEN_FILENOTEXIST: The file doesn't exist
EZNC_FILE_OPEN_OPEN: Impossible to open the file
EZNC_FILE_WRITEFILE_ILLIGALFILE: The file is illegal
EZNC_FILE_WRITEFILE_WRITE: Impossible to write the data
EZ_ERR_NULLPTR: Argument is NULL pointer

☐ Return value

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐ Functions

This sets file management data in the designated tool life management file.
Make sure to initialize **IEZNcSubFunction** by executing **ChangeInit()** before executing. If not, an error occurs at executing.

☐ Reference

OpenFile3(), CloseFile2(), ReadFile2(), WriteFile(), GetToolLifeValueFile2()

☐ Designation

2.20.6 IEZNcSubFunction2::GetSpareToolOfFile — Getting the spare tool exchange data of tool life management file

☐Calling procedure (Custom interface)

```
HRESULT        GetSpareToolOfFile(
                LPCOLESTR lpcwszFileName,   // (I)  File name with path
                DWORD dwHead,               // (I)  Part system
                DWORD dwNo,                 // (I)  #
                LPOLESTR** lpppwszData,     // (O)  Tool exchange value character string array
                LONG* plRet                 // (O)  Error code
                )
```

☐Calling procedure (Automation interface)

```
                GetSpareToolOfFile (
                bstrFileName As STRING      // (I)  File name with path
                lHeadAs LONG                // (I)  Part system
                lNo As LONG                 // (I)  #
                pvData As VARIANT*          // (O)  Tool exchange value character string
                ) As LONG                   // (O)  Error code
```

☐Argument

lpcwszFileName: Set file name (including path) of tool life management file as **UNICODE** character string.
Set the file by absolute path as follows,
Drive name + ":" + \Directory name\File name

dwHead: Set part system

dwNo: Set # whose spare tool exchange data you wish to get

lpppwszData: Returns the spare tool exchange data as **UNICODE** character string array. The array is saved inside of EZSocket, so the client is required to release the memory area explicitly by using **CoTaskMemFree()**.

lpppwszData	Kinds of spare tool exchange data	Data range	Remarks
0	Master tool #	-	
1	Spare tool # 1	-	
2	Spare tool # 2	-	
3	Spare tool # 3	-	
4	Offset	0 to 2	

Automation argument:
bstrFileName: Refer to the explanation of *lpcwszFileName.*
lHead: Refer to the explanation of *dwHead.*
lNo: Refer to the explanation of *dwNo.*
pvData: Returns the spare tool exchange data value as **VARIANT**.

☐**Argument**

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK: Normal termination
EZNC_FILE_OPEN_FILENOTEXIST: The file doesn't exist
EZNC_FILE_READFILE_READ: Impossible to read the data
EZNC_DATA_NOT_EXIST: The file doesn't exist
EZ_ERR_MEMORY_ALLOC: Memory cannot be saved

☐**Return value**

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐**Functions**

This gets the spare tool exchange data of the designated **#** and spare tool kind. For details, refer to the Instruction Manual of **M6x5L**.
Make sure to initialize **IEZNcSubFunction** by executing **ChangeInit()** before executing. If not, an error occurs at executing.

☐**Reference**

OpenFile3(), CloseFile2(), ReadFile2(), WriteFile(), SetSpareToolFile()

☐**Designation**

2.20.7 IEZNcSubFunction2::SetSpareToolOfFile — Setting the spare tool exchange data of tool life management file

☐Calling procedure (Custom interface)

```
HRESULT        SetSpareToolOfFile (
                LPCOLESTR lpcwszFileName,   //  (I)   File name with path
                DWORD dwMode,               //  (I)   Setting mode
                DWORD dwHead,               //  (I)   Part system
                DWORD dwNo,                 //  (I)   #
                LPCOLESTR*  lppcwszData,    //  (I)   Tool exchange value character string array
                LONG* plRet                 //  (O)   Error code
                )
```

☐Calling procedure (Automation interface)

```
                SetSpareToolOfFile (
                bstrFileName As STRING      //  (I)   File name with path
                lModelAs LONG               //  (I)   Setting mode
                lHeadAs LONG                //  (I)   Part system
                lNo As LONG                 //  (I)   #
                vData As VARIANT            //  (I)   Tool exchange value character string array
                ) As LONG                   //  (O)   Error code
```

☐Argument

lpcwszFileName: Set file name (including path) of tool life management file as **UNICODE** character string.
Set the file by absolute path as follows,
Drive name + ":" + \Directory name\File name

dwMode: Set setting mode of tool life management file

Value	Meaning
EZNC_FILE_CREATE	Setting a new tool life management file
EZNC_FILE_OPEN	Setting in an existing tool life management file

dwHead: Set part system

dwNo: Set # whose spare tool exchange data you wish to get

lppcwszData: Set spare tool exchange data as UNICODE character string.

lppcwszData	Kinds of spare tool exchange data	Data range	Remarks
0	Master tool #	-	
1	Spare tool # 1	-	
2	Spare tool # 2	-	
3	Spare tool # 3	-	
4	Offset	0 to 2	

Automation argument:
bstrFileName: Refer to the explanation of *lpcwszFileName.*
lHead: Refer to the explanation of *dwHead.*
lMode: Refer to the explanation of *dwMode.*
lNo: Refer to the explanation of *dwNo.*
vData: Create spare tool exchange data as **UNICODE** character string and substitute the data for *vData*(**VARIANT**) to set.

☐ **Argument**

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK: Normal termination
EZNC_FILE_OPEN_FILENOTEXIST: The file doesn't exist
EZNC_FILE_OPEN_OPEN: Impossible to open the file
EZNC_FILE_WRITEFILE_WRITE: Impossible to write the data
EZ_ERR_NULLPTR: Argument is NULL pointer

☐ **Return value**

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐ **Functions**

This sets the spare tool exchange data of the designated **#** and spare tool kind. For details, refer to the Instruction Manual of **M6x5L**.
Make sure to initialize **IEZNcSubFunction** by executing **ChangeInit()** before executing. If not, an error occurs at executing.

☐ **Reference**

OpenFile3(), CloseFile2(), ReadFile2(), WriteFile(), GetSpareToolFile()

☐ **Designation**

2.20.8 IEZNcSubFunction2::GetToolWorkOffsetOfFile Getting the data from workpiece offset file

☐Calling procedure (Custom interface)

```
HRESULT    GetToolWorkOffsetOfFile(
           LPCOLESTR lpcwszFileName,      // (I) File name with path
           LONG lHead,                    // (I) Part system
           LONG lIndex,                   // (I) Workpiece coordinate system #
           LPCOLESTR* lppcwszAxis,        // (I) Axis name character string array
           LPOLESTR** lpppwszData,        //(O)Workpice coordinate data value character string array
           LONG* plRet                    //(O)Error code
           )
```

☐Calling procedure (Automation interface)

```
           GetToolWorkOffsetOfFile (
           bstrFileName As STRING         //(I)File name with path
           lHeadAs LONG                   //(I)Part system
           lIndex As LONG                 //(I)Workpiece coordinate system #
           vAxis As VARIANT               //(I) Axis name character string array
           pvData As VARIANT*             //(O)Workpice coordinate data value character string array
           ) As LONG                      //(O)Error code
```

☐Argument

lpcwszFileName: Set file name (including path) of tool life management file as **UNICODE** character string.
Set the file by absolute path as follows,
Drive name + ":" + \Directory name\File name

lHead: Set part system

lIndex: Set workpiece coordinate system # to read

Value	Meaning	**M6x5M**	**M6x5L**	**Magic64**	**C64**
54	G54 offset	Available	Available	Available	Available
55	G55 offset	Available	Available	Available	Available
56	G56 offset	Available	Available	Available	Available
57	G57 offset	Available	Available	Available	Available
58	G58 offset	Available	Available	Available	Available
59	G59 offset	Available	Available	Available	Available
60	EXT offset	Available	Available	Available	Available
61	P1 offset	Available	-	Available	Available
62	P2 offset	Available	-	Available	Available
:	:	:	:	:	:
155	P95 offset	Available	-	Available	Available
156	P96 offset	Available	-	Available	Available

☐ **Argument**

lppcwszAxis: Set axis name as **UNICODE** character string array. (E.g.: "X"). The number of elements of the array is 8 (0 to 7). Set **NULL** character string for the axes not existing. (**NULL** pointer cannot be designated.) Refer to the table above only in the case of **Magic64** or **C64**. In the cases of the other models, set **NULL** character string for all the elements.

lpppwszData: Returns the workpiece offset data value as **UNICODE** character string array. The array is saved inside of EZSocket, so the client is requied to release the memory area explicitly by using **CoTaskMemFree()**.

lpppwszData	Kinds of tool exchange data	Remarks
0	1st axis	
1	2nd axis	
2	3rd axis	
3	4th axis	
4	5th axis	
5	6th axis	
6	7th axis	
7	8th axis	

The unit depends on the NC's parameter setting, [inch] or [mm]. (However, with **M6x5M**, fixed to [mm].)

Automation argument:
bstrFileName: Refer to the explanation of *lpcwszFileName.*
vAxis: Refer to the explanation of *lppcwszAxis.*
pvData: Returns the workpiece offset data value as **VARIANT.**

plRet: Returns an error code (When using automation interface, returns a return value instead.)
S_OK: Normal termination
EZNC_FILE_OPEN_FILENOTEXIST: The file doesn't exist
EZNC_FILE_OPEN_OPEN: Impossible to open the file
EZNC_FILE_READFILE_READ: Impossible to read the data
EZNC_DATA_NOT_EXIST: The file doesn't exist
EZ_ERR_MEMORY_ALLOC: Memory cannot be saved

☐ **Return value**

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐ **Functions**

This gets the workpiece coordinate system offset value of the designated part system and axis. Make sure to initialize **IEZNcSubFunction** by executing **ChangeInit()** before executing. If not, an error occurs at executing.

☐ **Reference**

OpenFile3(), CloseFile2(), ReadFile2(), WriteFile(), SetToolWorkOffsetFile()

☐ **Designation**

2.20.9 IEZNcSubFunction2::SetToolWorkOffsetOfFile Setting data to workpiece offset file

☐Calling procedure (Custom interface)

```
HRESULT    SetToolWorkOffsetOfFile (
           LPCOLESTR lpcwszFileName,   //  (I)  File name with path
           LONG lMode,                 //  (I)  Setting mode
           LONG lHead,                 //  (I)  Part system
           LONG lIndex,                //  (I)  Workpiece coordinate system #
           LPCOLESTR* lppcwszAxis,     //  (I)  Axis name character string array
           LPCOLESTR* lppcwszData,     //  (I)  Workpice coordinate data value character string array
           LONG* plRet                 //  (O)  Error code
           )
```

☐Calling procedure (Automation interface)

```
           SetToolWorkOffsetOfFile (
           bstrFileName As STRING      //  (I)  File name with path
           lModeAs LONG                //  (I)  Setting mode
           lHeadAs LONG                //  (I)  Part system
           lIndex As LONG              //  (I)  Workpiece coordinate system #
           vAxis As VARIANT            //  (I)  Axis name character string array
           vData As VARIANT            //  (I)  Workpice coordinate data value character string array
           ) As LONG                   //  (O)  Error code
```

☐Argument

lpcwszFileName: Set file name (including path) of tool life management file as **UNICODE** character string.
Set the file by absolute path as follows,
Drive name + ":" + \Directory name\File name

dwMode: Set setting mode of tool life management file

Value	Meaning
EZNC_FILE_CREATE	Setting a new tool life management file
EZNC_FILE_OPEN	Setting in an existing tool life management file

lHead: Set part system

lIndex: Set workpiece coordinate system # to read

Value	Meaning	**M6x5M**	**M6x5L**	**Magic64**	**C64**
54	G54 offset	Available	Available	Available	Available
55	G55 offset	Available	Available	Available	Available
56	G56 offset	Available	Available	Available	Available
57	G57 offset	Available	Available	Available	Available
58	G58 offset	Available	Available	Available	Available
59	G59 offset	Available	Available	Available	Available
60	EXT offset	Available	Available	Available	Available
61	P1 offset	Available	-	Available	Available
62	P2 offset	Available	-	Available	Available
:	:	:	:	:	:
155	P95 offset	Available	-	Available	Available
156	P96 offset	Available	-	Available	Available

☐ **Argument**

lppcwszAxis: Set axis name as **UNICODE** character string array. Set **NULL** character string for the axes not existing. Refer to the table above only in the case of **Magic64** or **C64**. In the cases of the other models, set **NULL** character string for all the elements.

lppcwszData: Set workpiece offset data as **UNICODE** character string. Set **NULL** character string for the axes not existing.
The unit depends on the NC's parameter setting, [inch] or [mm]. (However, with **M6x5M**, fixed to [mm].)

lppcwszData	Kinds of tool exchange data	Remarks
0	1st axis	
1	2nd axis	
2	3rd axis	
3	4th axis	
4	5th axis	
5	6th axis	
6	7th axis	
7	8th axis	

Automation argument:
bstrFileName: Refer to the explanation of *lpcwszFileName.*
vAxis: Refer to the explanation of *lppcwszAxis.*
vData: Create workpiece offset data as **UNICODE** character string and substitute the data for *vData*(**VARIANT**) to set. For workpiece offset data, refer to the explanation of *lppcwszData* and the index table above.

plRet: Returns an error code. (When using automation interface, returns a return value instead.)
S_OK: Normal termination
EZNC_FILE_OPEN_FILENOTEXIST: The file doesn't exist
EZNC_FILE_OPEN_OPEN: Impossible to open the file
EZNC_FILE_WRITEFILE_WRITE: Impossible to write the data
EZ_ERR_NULLPTR: Argument is NULL pointer

☐ **Argument**

```
[E.g.] In the case of M6x5M
LPOLESTR* lppwszAxis;
lppwszAxis = new LPOLESTR[8];
lppwszAxis [0] =L"";
lppwszAxis [1] =L"";
lppwszAxis [2] =L"";
lppwszAxis [3] =L"";
lppwszAxis [4] =L"";
lppwszAxis [5] =L"";
lppwszAxis [6] =L"";
lppwszAxis [7] =L"";
LPOLESTR* lppwszData;
lppwszData = new LPOLESTR[11];
lppwszData[0] =L"-1.000";
lppwszData[1] =L"1.000";
lppwszData[2] =L"3.000";
lppwszData[3] =L"";
lppwszData[4] =L"";
lppwszData[5] =L"";
lppwszData[6] =L"";
lppwszData[7] =L"";
hr = pIEZNcTool->SetToolWorkOffsetOfFile (L"C:\TEMP\OFFSET.WRK",EZNC_FILE_OPEN, 1,
54, (LPCOLESTR*)lppwszAxis ,(LPCOLESTR*)lppwszData, &lRet);
if( S_OK != hr ){
        wprintf(L"HRESULT Code = 0x%x, lRet Code = 0x%x\n", hr, lRet );
}
delete[ ] lppwszData;
```

☐ **Return value**

Return value	Meaning
S_OK	Normal termination
S_FALSE	Communication failure

☐ **Functions**

This sets the workpiece coordinate system offset value of the designated part system and axis. Make sure to initialize **IEZNcSubFunction** by executing **ChangeInit()** before executing. If not, an error occurs at executing.

☐ **Reference**

OpenFile3(), CloseFile2(), ReadFile2(), WriteFile(), GetToolWorkOffsetFile()

☐ **Designation**

3. ERROR CODE LIST

The error codes for EZSocket are as follows.

Table 3-1 Error code list

No	Error code	#	Description
1.	EZ_ERR_NOT_OPEN	0x80A00101	Communication line is not opened
2.	EZ_ERR_DOUBLE_OPEN	0x80A00104	Double open error
3.	EZ_ERR_DATA_TYPE	0x80A00105	Data type of the argument is illegal
4.	EZ_ERR_DATA_RANGE	0x80A00106	Data range of the argument is illegal
5.	EZ_ERR_NOT_SUPPORT	0x80A00107	Not supported
6.	EZ_ERR_CANNOT_OPEN	0x80A00109	Impossible to open communication line
7.	EZ_ERR_NULLPTR	0x80A0010A	Argument is NULL pointer
8.	EZ_ERR_DATA_LENGTH	0x80A0010B	Argument data illegal
9.	EZ_ERR_OPEN_COMM	0x80A0010C	COMM port handle error
10.	EZ_ERR_MEMORY_ALLOC	0x80B00101	Memory cannot be saved
11.	EZNC_ERR_CANNOT_GETPCERR	0x80B00102	Impossible to get an error of EZSocketPc
12.	EZNC_FILE_OPEN_MODE	0x80B00201	Open mode illegal
13.	EZNC_FILE_OPEN_NOTOPEN	0x80B00202	File cannot be opened
14.	EZNC_FILE_OPEN_FILEEXIST	0x80B00203	Target file already exists
15.	EZNC_FILE_OPEN_ALREADYOPENED	0x80B00204	File already open
16.	EZNC_FILE_OPEN_CREATE	0x80B00205	Impossible to create files
17.	EZNC_FILE_WRITEFILE_NOTOPEN	0x80B00206	File is not open in the writing mode
18.	EZNC_FILE_WRITEFILE_LENGTH	0x80B00207	Size of data to write is illegal
19.	EZNC_FILE_WRITEFILE_WRITE	0x80B00208	Impossible to write the data
20.	EZNC_FILE_READFILE_NOTOPEN	0x80B00209	File is not open in the reading mode
21.	EZNC_FILE_READFILE_READ	0x80B0020A	Impossible to read the data
22.	EZNC_FILE_READFILE_CREATE	0x80B0020B	Impossible to create temporary file
23.	EZNC_FILE_OPEN_FILENOTEXIST	0x80B0020C	The file doesn’t exist (READ mode)
24.	EZNC_FILE_OPEN_OPEN	0x80B0020D	Impossible to open the file
25.	EZNC_FILE_OPEN_ILLEGALPATH	0x80B0020E	Illegal path
26.	EZNC_FILE_READFILE_ILLEGALFILE	0x80B0020F	The file is illegal
27.	EZNC_FILE_WRITEFILE_ILLEGALFILE	0x80B00210	The file is illegal
28.	EZNC_COMM_NOTSETUP_PROTOCOL	0x80B00302	TCP/IP communication has not been set
29.	EZNC_COMM_ALREADYOPENED	0x80B00303	Setting impossible as the line is already open
30.	EZNC_COMM_NOTMODULE	0x80B00304	Submodules don’t exist
31.	EZNC_COMM_CREATEPC	0x80B00305	Impossible to create an EZSocketPc object
32.	EZNC_DATA_NOT_EXIST	0x80B00401	The data doesn’t exist
33.	EZNC_DATA_DUPLICATE	0x80B00402	Data already exists
34.	EZNC_PARAM_FILENOTEXIST	0x80B00501	Parameter information file doesn’t exist
35.	EZNC_SYSFUNC_IOCTL_ADDR	0x80020190	NC card # is illegal
36.	EZNC_SYSFUNC_IOCTL_NOTOPEN	0x80020102	Device not open
37.	EZNC_SYSFUNC_CREATEPC	0x80020133	Communication parameter data range illegal
38.	EZNC_FILE_DIR_FILESYSTEM	0x80030143	File system is abnormal
39.	EZNC_FILE_DIR_NODIR	0x80030191	Directory doesn't exist
40.	EZNC_FILE_DIR_NODRIVE	0x8003019B	Drive doesn't exist
41.	EZNC_PCFILE_DIR_NODIR	0x800301A2	Directory doesn't exist
42.	EZNC_PCFILE_DIR_NODRIVE	0x800301A8	Drive doesn't exist
43.	EZNC_OPE_CURRALM_ADDR	0x80050D90	Part system/axis designation is illegal
44.	EZNC_OPE_CURRALM_ALMTYPE	0x80050D02	Alarm type is illegal
45.	EZNC_OPE_CURRALM_DATAERR	0x80050D03	Communication data errors between NC and PC
46.	EZNC_OPE_CURRALM_DATASIZE	0x80050D93	Too much data for the buffer prepared by the application
47.	EZNC_OPE_CURRALM_DATATYPE	0x80050D94	Data type illegal
48.	EZNC_OPE_CURRALM_NOS	0x80050D01	The number of messages is illegal

No	Error code	#	Description
49.	EZNC_OPE_GETPRGBLK_ADDR	0x80050C90	Part system designation is illegal
50.	EZNC_OPE_GETPRGBLK_NOS	0x80050D01	The designation of the number of blocks illegal
51.	EZNC_OPE_SELECTPRG_RESET	0x80051004	Operation search impossible (In resetting)
52.	EZNC_OPE_SELECTPRG_LONGPATH	0x80051005	Path too long
53.	EZNC_OPE_SELECTPRG_TIMEOUT	0x80051007	Timeout
54.	EZNC_OPE_SELECTPRG_ADDR	0x80051090	Part system designation is illegal
55.	EZNC_OPE_SELECTPRG_FILEREAD	0x80051094	File reading error
56.	EZNC_OPE_SELECTPRG_FILEWRITE	0x80051095	File writing error
57.	EZNC_OPE_SELECTPRG_FILESYSTEM	0x80051043	File system is abnormal
58.	EZNC_OPE_SELECTPRG_NOPRG	0x80051002	No program file
59.	EZNC_OPE_SELECTPRG_PRGFORMAT	0x80051001	Program file name format is illegal
60.	EZNC_OPE_SELECTPRG_RUNNING	0x80051003	Program is running
61.	EZNC_OPE_ACTPLC_ADDR	0x80050990	NC card is illegal
62.	EZNC_DATA_TLFGROUP_ADDR	0x80041090	Address (part system designation) is illegal
63.	EZNC_DATA_TLFGROUP_EXIST	0x80041091	The group # already exists
64.	EZNC_DATA_TLFGROUP_NONEXIST	0x80041092	The group # doesn’t exist
65.	EZNC_DATA_TLFGROUP_OVER	0x80041093	Registered number of groups is over the range
66.	EZNC_DATA_TLFGROUP_NONFORMAT	0x80041094	Format incomplete
67.	EZNC_DATA_TLFGROUP_UNMACH	0x80041096	Entered group # doesn’t match
68.	EZNC_DATA_TLFGROUP_OUTOFSPEC	0x80041097	Designated group # is out of the specifications
69.	EZNC_DATA_TLFTOOL_ADDR	0x80041190	Address (part system designation) is illegal
70.	EZNC_DATA_TLFTOOL_EXIST	0x80041191	The tool # already exists
71.	EZNC_DATA_TLFTOOL_NONEXIST	0x80041192	The tool # doesn’t exist
72.	EZNC_DATA_TLFTOOL_OVER	0x80041193	Registered number of tools is over the range
73.	EZNC_DATA_TLFTOOL_PARAMERR	0x80041194	Designated kind of tool life management data is illegal
74.	EZNC_DATA_TLFTOOL_MAXMINERR	0x80041195	Data set is over the range
75.	EZNC_DATA_TLFTOOL_UNMACH	0x80041196	Entered tool # doesn't match
76.	EZNC_DATA_TLFTOOL_OUTOFSPEC	0x80041197	Designated tool # is out of the specifications
77.	EZNC_DATA_READ_ADDR	0x80040190	Part system/axis designation is illegal
78.	EZNC_DATA_READ_SECT	0x80040191	Class # illegal
79.	EZNC_DATA_READ_SUBSECT	0x80040192	Sub-class # illegal
80.	EZNC_DATA_READ_DATASIZE	0x80040196	Too much data for the buffer prepared by the application
81.	EZNC_DATA_READ_DATATYPE	0x80040197	Data type illegal
82.	EZNC_DATA_READ_READ	0x8004019D	Impossible to read the data
83.	EZNC_DATA_READ_WRITEONLY	0x8004019F	Writing-dedicated data
84.	EZNC_DATA_WRITE_ADDR	0x80040290	Part system/axis designation is illegal
85.	EZNC_DATA_WRITE_SECT	0x80040291	Class # illegal
86.	EZNC_DATA_WRITE_SUBSECT	0x80040292	Sub-class # illegal
87.	EZNC_DATA_WRITE_DATASIZE	0x80040296	Too much data for the buffer prepared by the application
88.	EZNC_DATA_WRITE_DATATYPE	0x80040297	Data type illegal
89.	EZNC_DATA_WRITE_READONLY	0x8004029B	Reading-dedicated data
90.	EZNC_DATA_WRITE_WRITE	0x8004029E	Impossible to write the data
91.	EZNC_DATA_MDLCANCEL_NOTREGIST	0x80040501	Not registered as high speed reading
92.	EZNC_DATA_MDLREGIST_PRIORITY	0x80040402	Priority designation illegal
93.	EZNC_DATA_MDLREGIST_REGIST	0x80040401	The number of registrations over
94.	EZNC_FILE_DIR_ALREADYOPENED	0x80030101	Another directory is already open
95.	EZNC_FILE_DIR_DATASIZE	0x80030103	Data size over
96.	EZNC_FILE_DIR_NAMELENGTH	0x80030148	File name too long
97.	EZNC_FILE_DIR_NOTOPEN	0x80030190	Not opened
98.	EZNC_FILE_DIR_READ	0x80030194	File information reading error

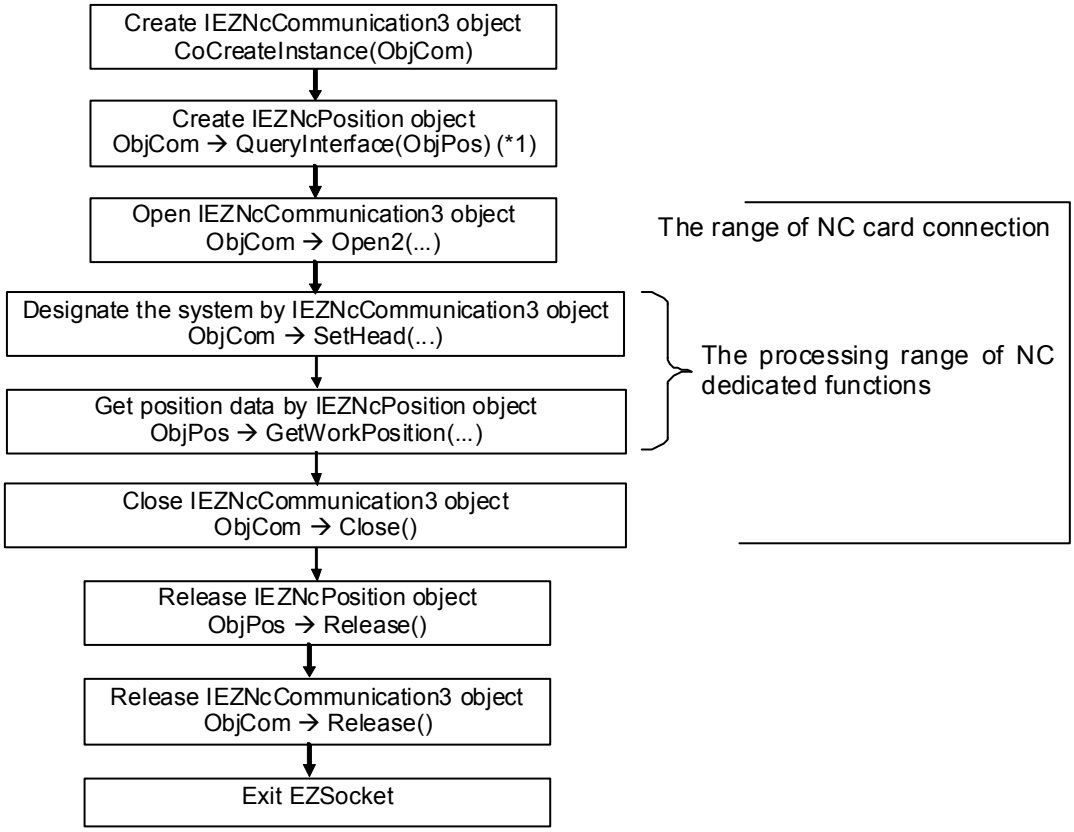
No	Error code	#	Description
99.	EZNC_PCFILE_DIR_NOTOPEN	0x800301A0	Not opened
100.	EZNC_PCFILE_DIR_NOFILE	0x800301A1	The file doesn’t exist
101.	EZNC_PCFILE_DIR_READ	0x800301A5	File information reading error
102.	EZNC_FILE_COPY_BUSY	0x80030447	Impossible to make copies (In operation)
103.	EZNC_FILE_COPY_ENTRYOVER	0x80030403	The number of registrations over
104.	EZNC_FILE_COPY_FILEEXIST	0x80030401	Target file already exists
105.	EZNC_FILE_COPY_FILESYSTEM	0x80030443	File system is abnormal
106.	EZNC_FILE_COPY_NAMELENGTH	0x80030448	File name too long
107.	EZNC_FILE_COPY_ILLEGALNAME	0x80030498	File name format is illegal
108.	EZNC_FILE_COPY_MEMORYOVER	0x80030404	Memory over
109.	EZNC_FILE_COPY_NODIR	0x80030491	Directory doesn’t exist
110.	EZNC_FILE_COPY_NODRIVE	0x8003049B	Drive doesn’t exist
111.	EZNC_FILE_COPY_NOFILE	0x80030442	The file doesn’t exist
112.	EZNC_FILE_COPY_PLCRUN	0x80030446	Impossible to make copies (PLC running)
113.	EZNC_FILE_COPY_READ	0x80030494	Impossible to read the source file
114.	EZNC_FILE_COPY_WIRTE	0x80030495	Impossible to write in the target file
115.	EZNC_FILE_COPY_PROTECT	0x8003044A	Impossible to copy (Protected)
116.	EZNC_PCFILE_COPY_CREATE	0x800304A4	Impossible to create files (PC)
117.	EZNC_PCFILE_COPY_OPEN	0x800304A3	Impossible to open files (PC)
118.	EZNC_PCFILE_COPY_FILEEXIST	0x80030402	Target file already exists
119.	EZNC_PCFILE_COPY_ILLEGALNAME	0x800304A7	File name format is illegal
120.	EZNC_PCFILE_COPY_NODIR	0x800304A2	Directory doesn't exist
121.	EZNC_PCFILE_COPY_NODRIVE	0x800304A8	Drive doesn't exist
122.	EZNC_PCFILE_COPY_NOFILE	0x800304A1	The file doesn't exist
123.	EZNC_PCFILE_COPY_READ	0x800304A5	Impossible to read the source file
124.	EZNC_PCFILE_COPY_WIRTE	0x800304A6	Impossible to write in the target file
125.	EZNC_FILE_DEL_NOTDELETE	0x80030201	Impossible to delete the file
126.	EZNC_FILE_DEL_NOFILE	0x80030242	The file doesn't exist
127.	EZNC_FILE_DEL_FILESYSTEM	0x80030243	File system is abnormal
128.	EZNC_FILE_DEL_BUSY	0x80030247	Impossible to delete files (In operation)
129.	EZNC_FILE_DEL_NAMELENGTH	0x80030248	File name too long
130.	EZNC_FILE_DEL_PROTECT	0x8003024A	Impossible to delete (Protected)
131.	EZNC_FILE_DEL_NODIR	0x80030291	Directory doesn't exist
132.	EZNC_FILE_DEL_ILLEGALNAME	0x80030298	File name format is illegal
133.	EZNC_FILE_DEL_NODRIVE	0x8003029B	Drive doesn't exist
134.	EZNC_PCFILE_DEL_NOTDELETE	0x80030202	Impossible to delete the file
135.	EZNC_PCFILE_DEL_ILLEGALNAME	0x800302A7	File name format is illegal
136.	EZNC_PCFILE_DEL_NODIR	0x800302A2	Directory doesn't exist
137.	EZNC_PCFILE_DEL_NODRIVE	0x800302A8	Drive doesn't exist
138.	EZNC_PCFILE_DEL_NOFILE	0x800302A1	The file doesn't exist
139.	EZNC_FILE_REN_FILEEXIST	0x80030301	New file name already exists
140.	EZNC_FILE_REN_NOFILE	0x80030342	The file doesn't exist
141.	EZNC_FILE_REN_FILESYSTEM	0x80030343	File system is abnormal
142.	EZNC_FILE_REN_BUSY	0x80030347	Impossible to rename (In operation)
143.	EZNC_FILE_REN_NAMELENGTH	0x80030348	File name too long
144.	EZNC_FILE_REN_PROTECT	0x8003034A	Impossible to rename (Protected)
145.	EZNC_FILE_REN_NODIR	0x80030391	Directory doesn't exist
146.	EZNC_FILE_REN_ILLEGALNAME	0x80030398	File name format is illegal
147.	EZNC_FILE_REN_NODRIVE	0x8003039B	Drive doesn't exist
148.	EZNC_PCFILE_REN_NOTRENAME	0x80030303	Impossible to rename
149.	EZNC_PCFILE_REN_SAMENAME	0x80030305	New file name is the same as the former name

No	Error code	#	Description
150.	EZNC_PCFILE_REN_FILEEXIST	0x80030302	New file name already exists
151.	EZNC_PCFILE_REN_ILLEGALNAME	0x800303A7	File name format is illegal
152.	EZNC_PCFILE_REN_NODIR	0x800303A2	Directory doesn't exist
153.	EZNC_PCFILE_REN_NODRIVE	0x800303A8	Drive doesn't exist
154.	EZNC_PCFILE_REN_NOFILE	0x800303A1	File doesn't exist
155.	EZNC_FILE_DISKFREE_NODIR	0x80030691	Directory doesn't exist
156.	EZNC_FILE_DRVLIST_DATASIZE	0x80030701	Too much data for the buffer prepared by the application
157.	EZNC_FILE_DRVLIST_READ	0x80030794	Drive information reading error
158.	EZNC_ENET_ALREADYOPEN	0x82020001	File already open
159.	EZNC_ENET_NOTOPEN	0x82020002	Not opened
160.	EZNC_ENET_CARDNOTEXIST	0x82020004	The card doesn't exist
161.	EZNC_ENET_BADCHANNEL	0x82020006	Channel # illegal
162.	EZNC_ENET_BADFD	0x82020007	File descriptor illegal
163.	EZNC_ENET_NOTCONNECT	0x8202000A	Not connected
164.	EZNC_ENET_NOTCLOSE	0x8202000B	Not closed
165.	EZNC_ENET_TIMEOUT	0x82020014	Timeout
166.	EZNC_ENET_DATAERR	0x82020015	Data illegal
167.	EZNC_ENET_CANCELED	0x82020016	Terminated by the cancel request
168.	EZNC_ENET_ILLEGALSIZE	0x82020017	Packet size illegal
169.	EZNC_ENET_TASKQUIT	0x82020018	Terminated by task completion
170.	EZNC_ENET_UNKNOWNFUNC	0x82020032	Command illegal
171.	EZNC_ENET_SETDATAERR	0x82020033	Setting data illegal
172.	Error code output by the NC	0xF00000FF	Argument is illegal
173.	Error code output by the NC	0xFFFFFFFF	Impossible to read/write the data
174.	Error code output by EZSocket for MELSEC PLC (EZSocketPc) (Only C70)	0x01XXXXXX 0x02XXXXXX 0x03XXXXXX 0x04XXXXXX 0x10XXXXXX	For details, refer to the following manuals. ・EZSocket Standard Reference Manual (MELSEC) (BAD-801Q074) ・EZSocket ProFX Reference Manual (MELSEC) (BAD-801Q117)
175.	Other than above	0x8007XXXX	Error during file operation (CNC700 only) Check the following factors. • The same file already exists in "Write open". • More than 10 files were opened simultaneously. • "Write open" is not possible during "read open". • Write/read file is not open. • Memory size in target file is insufficient. • Argument is incorrect.

4. HOW TO USE API

4.1 EZSocketAPI Operation Procedure

The following shows the operation procedure and precautions when using EZSocket.
The flow chart below shows the operation procedure of EZSocket.



4.2 Initialization for OLE/COM Interface Use

EZSocket uses OLE/COM interface. Thus, VC++ project has to support OLE/COM. If OLE/COM is not supported when programming the project, it is possible to use OLE/COM by correcting appropriate part in the 2 files shown below.
The following explanation is the case when the project is titled as "Project".

Project.cpp

```
BOOL CProjectApp::InitInstance()
{
    // Initializing OLE library
    if (!AfxOleInit())
    {
        // Show error message
        return FALSE;
    }

    // The following text is abridged.
}
```

Stdafx.h

```
// stdafx.h : Describe the standard system include file,
//            or the project exclusive include file
//            that are often referred and rarely changed.
//
#define VC_EXTRALEAN   // Except the stuff that are rarely used
                       // from Windows header.
#include <afxwin.h>          // MFC core and standard component
#include <afxext.h>          // Extended part of MFC
#include <afxdisp.h>         // MFC OLE/COM (Necessary for EZSocket)
#ifndef _AFX_NO_AFXCMN_SUPPORT
#include <afxcmn.h>          // MFC Windows common control support
#endif // _AFX_NO_AFXCMN_SUPPORT
```

4.3 Create EZSocket Object

EZSocket uses COM interface, thus, it is necessary to initialize COM (call CoInitialize) at the head of thread, and also necessary to terminate COM (call CoUninitialize) at the end of thread.

In the example mentioned below, the main aim of position data display application is displaying, and it is also a single thread, thus it creates the object when creating View windows, and releases when closing the windows.

First of all, create IEZNcCommunication3 object by using CoCreateInstance of COM library. Then create the other objects including IEZNcPosition by using QueryInterface. How to create IEZNcCommunication3 and IEZNcPosition object is mentioned below.

Table 4–1 Creating IEZNcCommunication3 object

How to create IEZNcCommunication3 communication object	
Calling procedure	CLSID clsid; IEZNcCommunication pComm; HRESULT hr = CLSIDFromProgID(L"EZSocketNc.EZNcCommunication",&clsid); *1 hr = CoCreateInstance(clsid, NULL, CLSCTX_INPROC_SERVER, IID_IEZNcCommunication, (VOID**)&pComm);
Return value	When S_OK is returned, succeeded in creating objects. When the other value is returned, failed in creating objects
Functions	This creates the communication object, and returns its address to parameter pComm.

*1 Refer to *2 of "1.7.1Programming procedure with VC++ (1)".

Table 4-2 Creating IEZNcPosition object

How to create IEZNcPosition parameter object	
Calling procedure	IEZNcPosition pPos; HRESUTL hr = pComm->QueryInterface(IID_IEZNcPosition,(void**)&pPos);
Return value	When S_OK is returned, succeeded in creating objects. When the other value is returned, failed in creating objects
Functions	This creates the parameter object, and returns its address to parameter pPos.

4.4 EZSocket Include Files

Include the header files mentioned below to use EZSocket when it is necessary.

#include "EZSocketNc.h"	EZSocket header file for method definition
#include “EZSocketNcStr.h”	EZSocket header file for structure definition (*)
#include "EZSocketNcDef.h"	EZSocket header file for various definition
#include "EZSocketNcErr.h"	EZSocket header file for error definition
#include “EasysocketDef.h”	EZSocket header file for various (MELSEC ATL interface) deficition

(*) Required to use EZSocket with Mitsubishi CNC C70 Series.

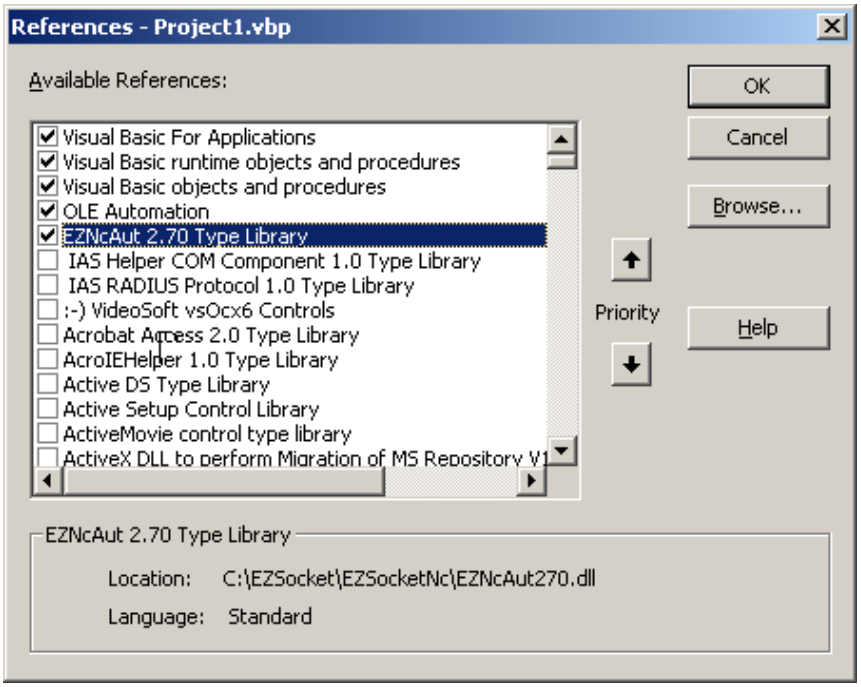
4.5 Programming EZSocket Automation Interface by VB

This chapter is about EZSocket programming by Microsoft VisualBasic (VB). The program on VC++ and VB can be programmed nearly the same way, thus, it is possible to verify the application in early stage by creating prototype by VB.
It is also possible to program timely and easily by using VB programming support function.

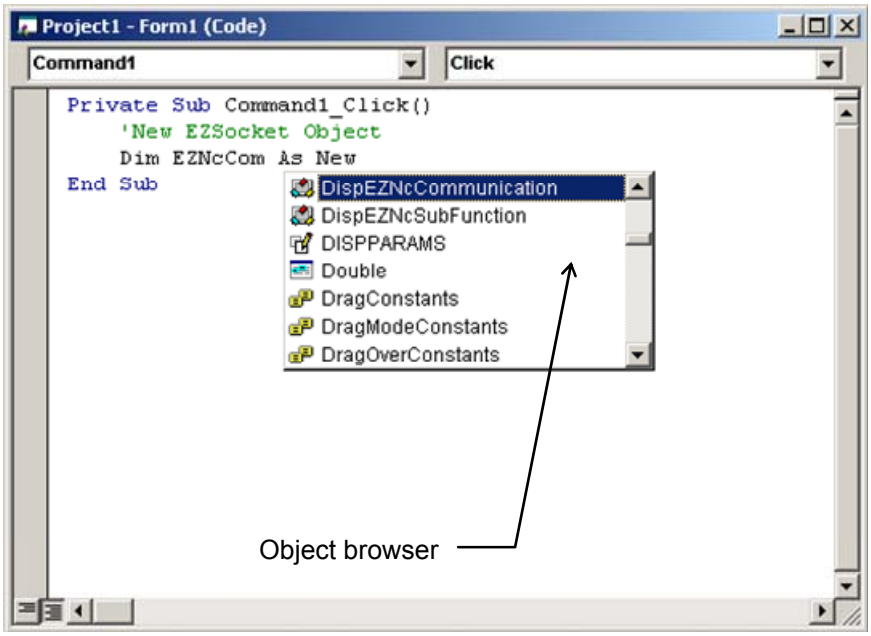
4.5.1 How to use EZSocket OLE automation interface by VB

1) Reference set: Select EZSocket object library.

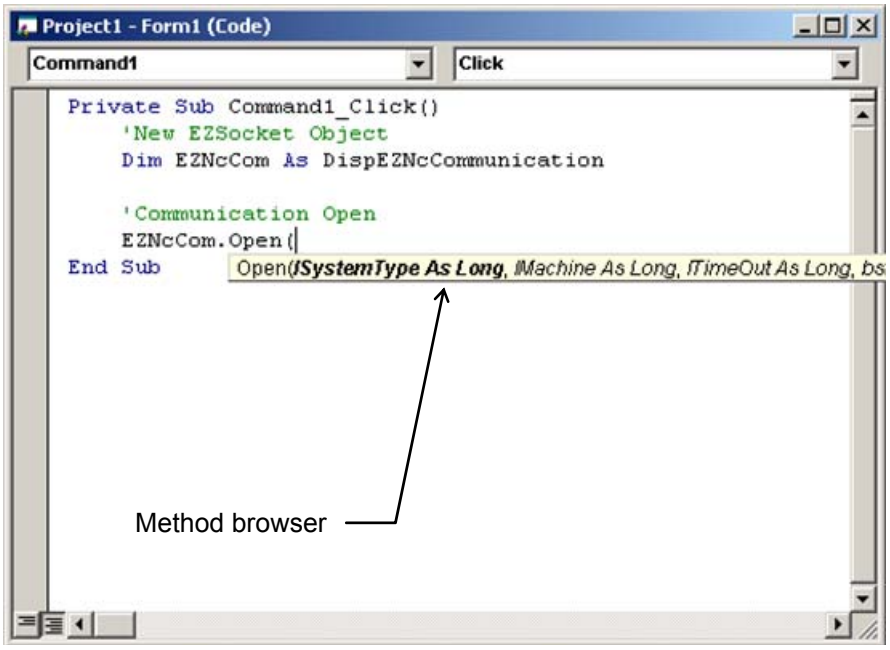
The method mentioned here is early binding by reference set. By setting reference set, it enables to use VB object browser function.
(Install EZSocket in advance)



2) Object browser: Select EZNcCommunication object on program.



3) Method browser: Select the method of EZNcCom object, and confirm the argument on the program.

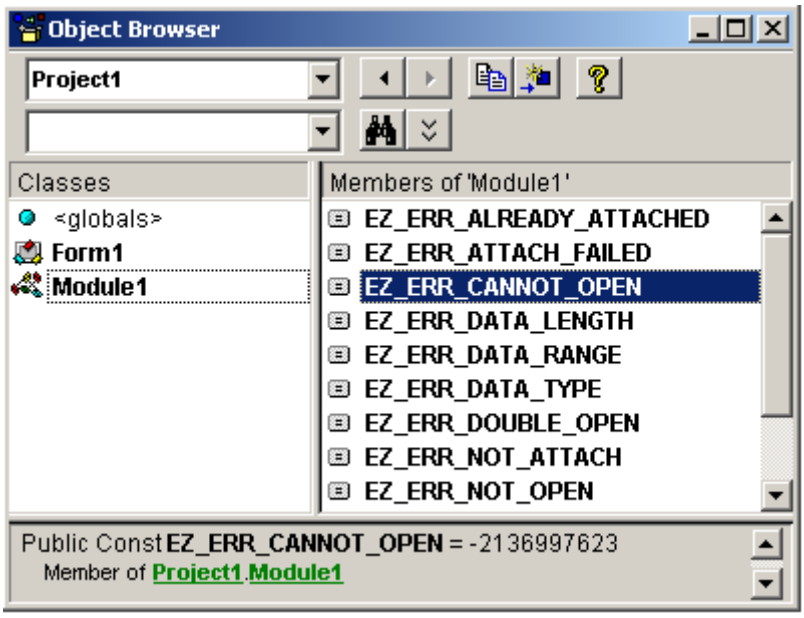


4) EZSocket module file: Set the definition, error code reference.

Add EZSocket module file so that EZSocket definitions and error code definitions can be used on VB.

Add EZNcDef.bas, EZNcErr.bas, EZComErr.bas module filea to project in order of [Project] → [Adding standard modules] → [Existing files].

This enables to refer to the definition and the error code definition easily by VB object browser function.



4.5.2 How to program by VB (1)

How to program by early binding. It is necessary to set EZSocket reference set.

```
Private Sub Command1_Click()
    'Create EZSocket object
    Dim EZNcCom As New DispEZNcCommunication

    'Open communication lines
    Dim lRet As Long
    lRet = EZNcCom.Open2(EZNC_SYS_MAGICBOARD64, 1, 1)
    If lRet <> 0 Then GoTo Error_Proc

    'Various processing
    lRet = EZNcCom.SetHead(1)
    If lRet <> 0 Then GoTo Error_Proc

    Dim CurPos(1 To 3) As Double
    For Axis = 1 To 3
        lRet = EZNcCom.Position_GetCurrentPosition(Axis,CurPos(Axis))
        If lRet <> 0 Then GoTo Error_Proc
    Next Axis

    X.Text = CurPos(1)
    Y.Text = CurPos(2)
    Z.Text = CurPos(3)

    'Close lines
    lRet = EZNcCom.Close
    If lRet <> 0 Then GoTo Error_Proc

    GoTo Last_Proc
Error_Proc:
    MsgBox ("Error! Code = " + "&H" + CStr(Hex(lRet)))
Last_Proc:
    'Release the object
    Set EZNcCom = Nothing
End Sub
```

4.5.3 How to program by VB (2)

How to program by late binding. It is not necessary to set EZSocket reference set. However, VB's object browser function cannot be used.

```
Private Sub Command1_Click()
    'Create EZSocket object
    Dim EZNcCom As Object
    Set EZNcCom = CreateObject("EZNcAut.DispEZNcCommunication","10.20.123.12")

    'Open communication lines
    Dim lRet As Long
    lRet = EZNcCom.Open2(EZNC_SYS_MAGICBOARD64, 1, 1)
    If lRet <> 0 Then GoTo Error_Proc

    'Various processing
    lRet = EZNcCom.SetHead(1)
    If lRet <> 0 Then GoTo Error_Proc

    Dim CurPos(1 To 3) As Double
    For Axis = 1 To 3
        lRet = EZNcCom.Position_GetCurrentPosition(Axis,CurPos(Axis))
        If lRet <> 0 Then GoTo Error_Proc
    Next Axis

    X.Text = CurPos(1)
    Y.Text = CurPos(2)
    Z.Text = CurPos(3)

    'Close lines
    lRet = EZNcCom.Close
    If lRet <> 0 Then GoTo Error_Proc

    GoTo Last_Proc
Error_Proc:
    MsgBox ("Error! Code = " + "&H" + CStr(Hex(lRet)))
Last_Proc:
    'Release the object
    Set EZNcCom = Nothing
End Sub
```

Set the IP address or domain name of the target.

(Note) For how to set the 1st argument of CreateObject(), refer to *2 of "1.7.1Programming procedure with VC++ (1)".

5. APPLICATION INSTALLATION PROCEDURE

5.1 Overview

To redistribute and perform the applications that uses EZSocket, it is necessary to copy and the software modules and EZSocket files onto the computer to perform the application, and also necessary to set them in the system registry.
This chapter explains its outline and procedure.

The following two methods are available for redistributing EZSocket. Depending on the customer's application environment, select either one of them.
(1) Use the EZSocket installer for redistribution incorporated in the EZSocket product CD.
(2) Have the customer create an installer according to the EZSocket redistribution procedure.

<<Point of selection and cautions>>
If the EZSocket installer for redistribution is used (See (1) above.), creation of installer can be omitted. Note that, however, an additional mechanism is required so that the EZSocket installer for redistribution can be executed from the installer on the customer application.
If an installer is created according to the EZSocket redistribution procedure (See (2) above.), that installer can be incorporated with the installer on the customer application. Especially, this method is used when the EZSocket installer for redistribution is difficult to be incorporated.
EZSocket can be used by several applications. Thus, follow the instructions to prevent the other applications from troubles when installing or uninstalling applications.

5.2 Redistribution by Using EZSocket Installer for Redistribution

In this section, redistribution method by using the EZSocket installer for redistribution incorporated in the EZSocket product CD is explained.
This EZSocket installer for redistribution, which was created according to the "5.3 Rules on EZSocket Redistribution", allows easy redistribution.

5.2.1 Path to the folder in which EZSocket installer for redistribution is saved

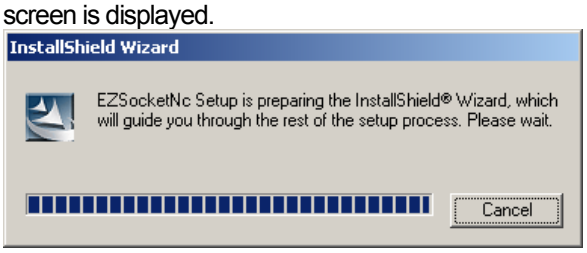
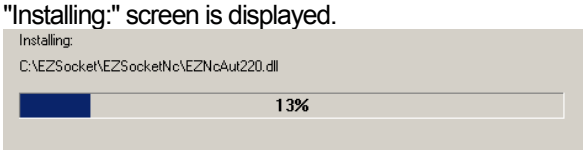
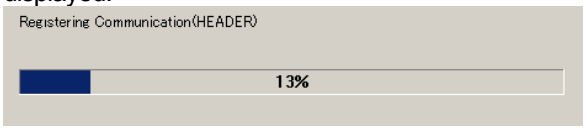
The EZSocket installer for redistribution that can be incorporated in the customer product is saved in the following folder within the EZSocket product CD.
Path to the folder: EZSocketNc\RedistributableInstaller folder

5.2.2 Specification and processing of the EZSocket installer for redistribution

Specification and processing of the EZSocket installer for redistribution are explained here.

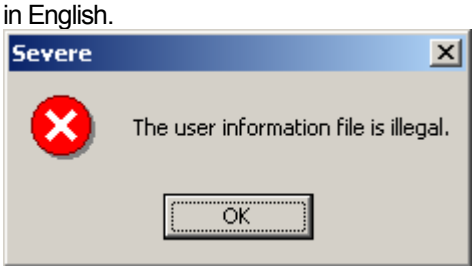
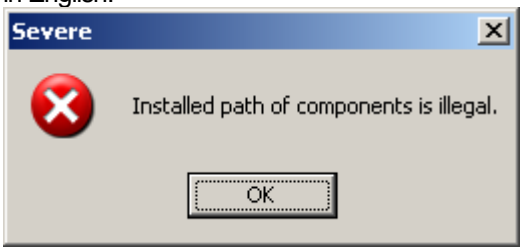
Careful operation check is required when incorporating the EZSocket installer for redistribution in the customer's product.

No	Processing	Specification
1	Execute the installer of customer's product.	Execute the following items for the installer of customer's product. (1) Customer's installer creates EZSNCSET.INI file in the manageable folder. Specification of EZSNCSET.INI file is as follows. [Specification of EZSNCSET.INI] [USER] Name=User name (Max. 256 byte) Company=User company name (Max. 256 byte) [SETUP] Target=Path for EZSocket installation destination [USER] section The value of Name is registered in "Name" in table 5-1. The value of Company is registered in "Organization" in table 5-1. [SETUP] section The value of Target is registered in "InstallPath" in table 5-1. Note that if InstallPath is registered already, the InstallPath is given priority. EZSocket is installed in (Target)\EasysocketNc. Recommended folder to be specified for "Target" is as shown below. Target=C:\EZSocket [Example of EZSNCSET.INI] [USER] Name=Taro Mitsubishi Company=MITSUBISHI [SETUP] Target=C:\Socket
2	Execute the EZSocket installer for redistribution from the installer of customer product's.	Execute setup.exe of the EZSocket installer for redistribution stored in the media of the customer's product (ex.CD-ROM) with the following command line. Full path including Setup.exe△EZSNCSET.INI. △ indicates space. Ex.) Setup.exe C:\temp Put EZSNCSET.INI in C:\temp folder.

No	Processing	Specification
3	The EZSocket installer for redistribution preparation screen is displayed.  	Screen for EZSocket installation preparation is displayed. If OS version is other than Japanese, this is displayed in English.
4	"Installing:" screen is displayed.  	EZSocket is installed in the (Target)\EasySocket folder specified with EZSNCSET.INI. If OS version is other than Japanese, this is displayed in English.
5	"Registering Communication (HEADER)" screen is displayed.  	Registry information required for EZSocket is registered according to the EZSocket installation procedure. If OS version is other than Japanese, this is displayed in English.
6	Upon completion of registry registration, the screen is closed and installation of EZSocket is ended.	Upon completion of registry registration, "Registering Communication (HEADER)" screen is closed and installation of EZSocket is ended.
7	Customer's product installer refers to the Error value in the [ERROR] section of EZSNCSET.INI. Execute the installation operation after that. When the EZSocket installer operation results in error, troubleshoot the error state and execute the EZSocket installer for redistribution again according to the procedure above. Refer to (2) Troubleshooting for details on common errors.	Termination result is added to EZSNCSET.INI when EZSocket installer is ended. [Specification of EZSNCSET.INI] [USER] Name=User name (Max. 256 byte) Company=User company name (Max. 256 byte) [SETUP] Target=EZSocket installation folder [ERROR] Error=0 * 0 for normal end, 1 for abnormal end [ERROR] section The value of Error is 0 when installation is ended normally and 1 when ended abnormally.
8	Delete EZSNCSET.INI from customer product's installer.	EZSNCSET.INI is a common file. Delete it at the end.

5.2.3 Troubleshooting

The remedies for the errors occurred with the EZSocket installer for redistribution are shown below.

No	Error case	Remedy
1	When EZSPCSET.INI is illegal, the following is displayed. Press [OK] to end. If OS version is other than Japanese, this is displayed in English.  	[Cause] (1) User name or user company name cannot be obtained. (2) User name or user company name exceeds 256 byte in size. (3) EZSocket installation destination path cannot be obtained. [Remedy] EZSNCSET.INI [ERROR] Error=1 Check the details of EZSNCSET.INI and set correctly. Or put EZSNCSET.INI in designated folder.
2	When the installation path is illegal, the following is displayed. Press [OK] to end. If OS version is other than Japanese, this is displayed in English.  	[Cause] The read installation path is illegal. [Remedy] EZSNCSET.INI [ERROR] Error=1 The path specified in Target of EZSNCSET.INI is illegal. Specify the correct path.

5.3 Rules on EZSocket Redistribution

Rules concerning redistribution of EZSocket are explained in this section.

5.3.1 Redistributable module group

Redistributable module group

- EZSocket redistributable files

This file will be installed in hard disk when installing EZSocket on the developed machine, however, do not make copies from the hard disk but make copies from EZSocket install CD to include the right version of copies in distribute disk.

5.3.2 Redistributable files of EZSocket

Redistributable files of EZSocket mentioned below are in EZSocket install CD. When distributing, distribute both EZSocket custom interface and EZSocket automation interface. If the version of each file are different, a failure can occur.

\Lib\EZSocketNc.dll --- DLL for EZSocket custom interface
\Lib\EZNcAut.dll -------- DLL for EZSocket automation interface
(A number of the version is put into the "xxx" part of file name.)
\Lib\CommServer ----- EZSocket-related folder
\Lib\Parameter --------- EZSocket-related folder
\Lib\Ini\melcfg.ini ------ EZSocket initialization file

5.4 Installation

5.4.1 Upgrading redistributable files

Occasionally, an old file whose name is the same as the redistributed file is distributed by other applications. In this case, do not fail to overwrite the new version on the old version. Never overwrite the old version on the new version. Version check is normally done by the setup program. If the application does not have the setup program, it is necessary to check the version on your own when including the redistributed files.

(Note 1) If the EZSocket initialization file "melcfg.ini" already exists in the personal computer, do not overwirte even if the date is old.

(Note 2) In the case of EZSocket automation I/F, the file name of DLL is different by version, so do not delete the DLL file of the EZSocket automation I/F already installed. If deleted, applications that are supported by the already installed EZSocket automation I/F cannot be used.

5.4.2 Install destination directory of EZSocket files

When installing EZSocket for the first time, it is possible to install in an optional directory. When installing for and after the second time, install EZSocket in the same directory that the last install has done, and make sure that more than one EZSocket don't exist in one computer.

To make sure, follow the procedure below.

(1) How to judge whether it is the first install or not.

Check whether the registry key mentioned below exists or not, and whether the new path name is registered as data.

Registry key: HKEY_LOCAL_MACHINE\SOFTWARE\MITSUBISHI\EZSocketNc\CurrentVersion\InstallPath
Data (example): C:\EZSocket\EZSocketNc

When the right path has already been registered in registry key data, it means that your installation is the second installation (or installed more times). When the registry key does not exit or the path name is not registered, it means the first installation. Do not add '\" at the end of installation path; EZSocket-related files will not be read.

(2) In the first installation

Designate the install destination directory for EZSocket by users, and then register the directory designated here as an install destination directory of the registry. It is necessary to set the other registries at the same time. Refer to 5.6.

(3) In the second installation (or after)

Install EZSocket in the directory that has already registered. When there is no directory, create one. When making file copies, do not overwrite the old version on the new one.

5.5 Structure of the Install Destination Directory

The structure of the install destination directory is shown below.
Copy the files from EZSocket install CD to the directory mentioned below.

Install directory	The path in install CD
Arbitrary directory	
EZSocket	
EZSocketNc	Files in a directory such as \Lib\EZSocketNc.dll,EZNcAutxxx.dll
CommServer	Exists in \Lib\CommServer
C64	
C70	
M700	
Parameter	Exists in \Lib\Parameter
C64	Parameter files for C64 series
C70	Parameter files for C70 series
M700	Parameter files for 700 series
Windows folder	WindowsOS folder
melcfg.ini	\Lib\Ini\melcfg.ini
App	(Example) Application files

5.6 Registry Setting

5.6.1 Renewing EZSocket registry

The registry required to start EZSocket is shown in the table below.
To install EZSocket, register the data by creating the registry structure shown below.

Table 5-1EZSocket registry list

Key	Name	Type	Data	Remarks
HKEY_LOCAL_MACHINE				
└ SOFTWARE				
└─ MITSUBISHI				
└ EZSocketNc				
├─ CurrentVersion	Description	Character string	"EZSocketNc"	Fixed data
	Organization	Character string	Company name designated by user	Register the company name designated by user when installing
	Name	Character string	User name designated by user	Register the company name designated by user when installing
	MajorVersion	DWORD value	Version	
	MinorVersion	Character string	Version	
	InstallPath (Note1)	Character string	"The directory designated by user\EZSocketNc"	Register the company name designated by user when installing
	Meldasmagic64_ncapi32	Character string	“C:\melpcnc\bin32\ncapi32.dll”	(Note 3) Necessary with MAGIC64
	Meldas6x5M_ncapi32	Character string	“C:\ncsys\m6dll\ncapi32.dll”	(Note 3) Necessary with M6x5M
	Meldas6x5M_ncapi	Character string	“C:\ncsys\m6dll\ncapi.dll”	(Note 3) Necessary with M6x5M
	Meldas6x5M_alarm	Character string	“C:\ncsys\alarm.dll”	(Note 3) Necessary with M6x5M
	Meldas6x5L_ncapi32	Character string	“C:\ncsys\melpcnc\bin32\ncapi32.dll”	(Note 3) Necessary with M6x5L
├─ Custom	FileVersion (Note2)	Character string	EZSocketNc.dll File date	YYYY-MM-DD type
	EZSocketNcName	Character string	“EZSocketNc.dll”	Fixed data
├─ Automation	FileVersion (Note2)	Character string	EZNcAut.dll File date	YYYY-MM-DD type
	EZSocketNcName	Character string	“EZNcAutxxx.dll”	“xxx” are numerical numbers.

(Note 1)
The data to register in "InstallPath" has to be
"Drive:The directory designated by user when package installing+\EZSocketNc"

(Note 2)
Make copies of install destination files of the data registered in "FileVersion" to HD, get the time stamp of the designated files, and then register the data.

(Note 3)
In there is no file in the designated path, change the path of the registry.

5.7 System Environment Variable Setting

Necessary system environment variable to use EZSocket is as follows.
To install EZSocket, add the following to the system environment variables.
The followings are default values. If there is no file in the designated path, change the path.

Table 5-2 EZSocket system environment variable list

Model	System environment variable (Default)
MELDASMAGIC64	PATH=C:\melpcnc\bin32;
M615M, M635M, M655M	PATH=C:\ncsys\m6dll;C:\ncsys\alarm;
M615L, M635L	PATH=C:\ncsys\melpcnc\bin32;
MITSUBISHI CNC700 Series	PATH=EZSocket Installation path (Example: C:\EZSocket\EZSocketNc)

5.8 COM Information Registry Setting

EZSocketNc.dll and EZNcAutxxx.dll in the install destination directory require to register the COM information in the registry. It is possible to register them as follows by using redistributable REGSVR32.EXE command, which is attached to Microsoft Visual C++, when installing.

REGSVR32/s install destination directory\EZSocketNc.dll
REGSVR32/s install destination directory\EZNcAutxxx.dll

It is not essential to use REGSV32.EXE command as long as COM information is registered according to the specification of installer package. For example, when using InstallShieled5.5, create EZSocket items in the tab "FileGroup", set EZSocketNc.dll, EZNc.Autxxx.dll, and select "YES" for the "Self registering". Then it is registered in the registry automatically when installing.

5.9 Precaution for Uninstalling

EZSocket can be used by more than one application. Thus, If you delete EZSocket with uninstalling an application, it can influence on the rest of the applications. Do not delete EZSocket files and registries when uninstalling the applications which include EZSocket.

6. SAMPLE APPLICATION

6.1 Overview of Sample Application

Sample applications using EZSocket are provided with the project files which are able to compile Visual C++ Ver.6.0 and Visual Basic Ver.6.0. We also provide macro sample programs using EZSocket OLE interface macro that can call EZSocket custom interface easily. EZSocket OLE interface macro is provide as a sample.

EZSocket sample application includes the applications mentioned below.

- Position data display application:\samples\Vc\Position\Position.dsw
- Monitoring application: \samples\Vb\EZNcAutSample\EZNcAutSample.vbp
- Macro sample program: \samples\Vc\Macros\MacSmp\MacSmp.dsw

6.2 Position Data Display Application

This chapter is about the sample application using EZSocket for Visual C++ Ver.6.0

6.2.1 Performance environment

The sample application is operated in the system configuration mentioned below.

OS	WindowsNT 4.0, Windows2000 Windows 95, 98, WindowsXP
Compiler	Microsoft Visual C++ Ver.6.0
Main Body	EZSocketNc
Controller	MELDASMAGIC64, M6x5M/L, MELDAS C6/C64, CNC 700M/L, CNC C70
H/W	A personal computer that can operate the OS, compiler and controller mentioned above.

6.2.2 Installation and uninstallation

How to install and uninstall the sample application is as follows.
Refer to each manual about installing OS, VC++, and about operation of the H/W.

(1) Installation

The sample application is created in the sample folder when installing EZSocket itself.
The sample application has its source code and executable file within the project name folder. The sample application includes the project work space file for Visual C++ Ver. 6.0 By opening the corresponding project work space file, it is possible to open the project by Visual C++.

(2) Uninstallation

Delete the project name folder or sample folder to uninstall the sample application.

6.2.3 Executing sample application

Execute the sample application as follows.
The executable file is in Debug or Release folder under the sample application folder. In the case of the position data display application, execute Position.exe.
Refer to the following chapters about the position data display application.
FYI, the sample application mentioned above is the monitor application of MELDAS CNC. It is necessary to operate operation search, and cycle start etc. for the CNC. Refer to the Instruction Manual of each CNC.

6.2.4 Function list

The sample application's functions are as follows.
The position data display application monitors the designated position data and displays the result as a counter.

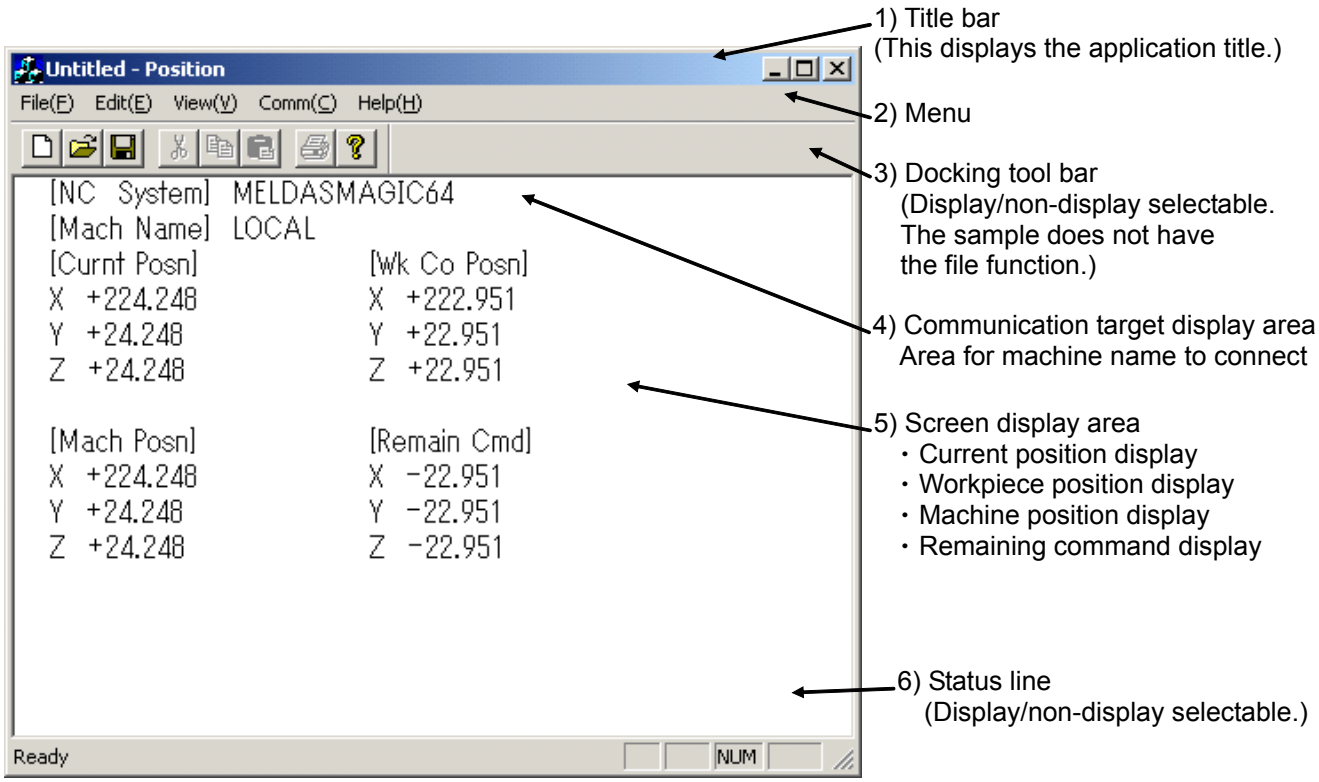
Table 6–1 Function list of position data display application

[File]	[Exiting from the application]	Exiting from position data display application.
[Editing]	[Position data]	This edits the position data to display. • Current position • Workpiece coordinate position • Machine position • Remaining commands
[Display]	[Refresh cycle]	This edits the refresh cycle for screen display.
[Communication]	[Communication selection]	This selects communication target. • MELDASMAGIC64 • MELDAS6x5M • MELDAS6x5L • MELDASC6/C64 • CNC C70 • CNC 700M • CNC 700L
	[Execution]	Communication start/stop
[Help]	[Version information]	This displays the version of position data display application.

6.2.5 Screen configuration and functions

The screen configuration of the position data display application and functions by menu items is as follows.

(1) Basic screen configuration
Below is the basic screen configuration.

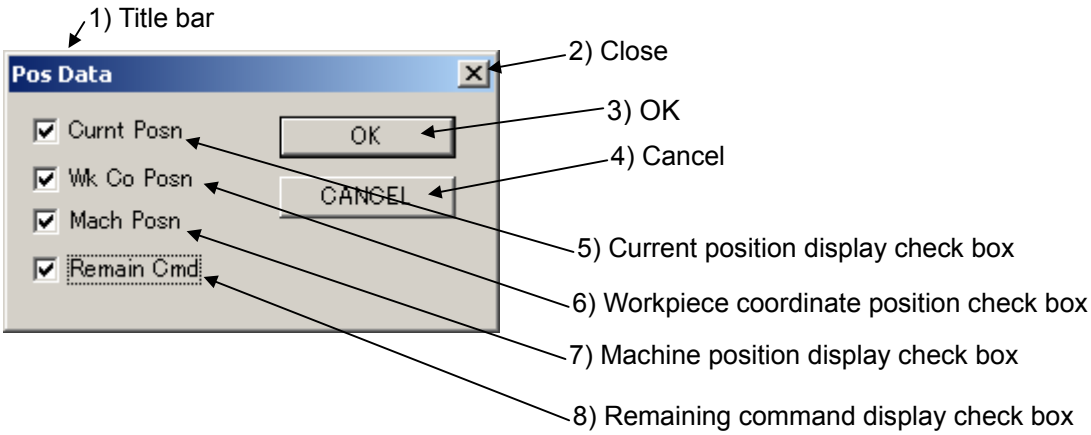


(2) File Functions
Position data display application does not have file function except for exiting the application.
This application does not execute the file selection.

(3) Editing functions

a. Position data dialogue box

Select position data type to display in the dialogue box.



1) to 4): Explanation is omitted here and afterwards.

5) Current position display check box:

This selects if it displays the current position at the moment of Dog-type return to origin, or the relative position from preset point by G92/origin set/counter set.

6) Workpiece coordinate position display check box:

This selects if it displays the position in the current workpiece coordinate system.

7) Machine position display check box:

This selects if it displays each axis position in base machine coordinate system.

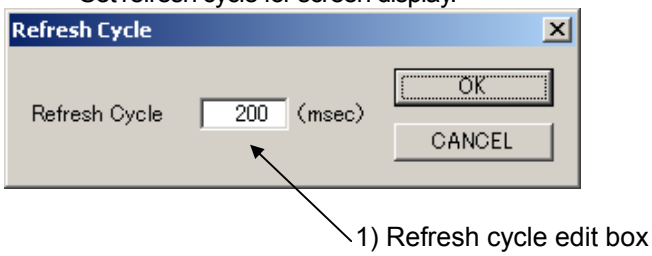
8) Remaining command display check box:

This selects if it displays the residual distance of the movement command, which is currently executed.

(4) Display function

Refresh cycle function dialogue

Set refresh cycle for screen display.



1) Refresh cycle edit box:

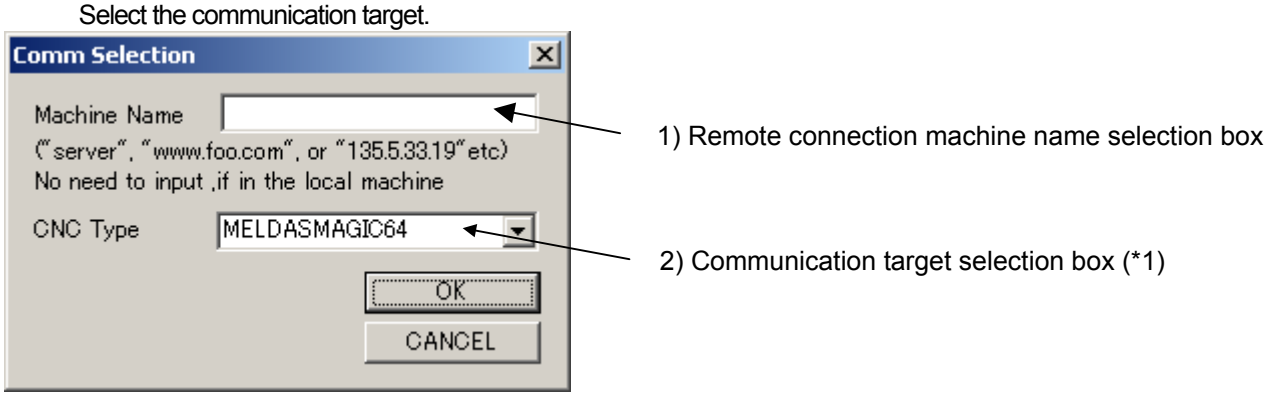
Designate the refresh cycle for screen display.

Data range: 200 to 10000 (ms)

(5) Communication function

Communication selection dialogue box

Select the communication target.



1) Remote connection machine name selection box:

Designate the machine name of the personal computer on which the NC is mounted.

It is possible to designate the domain name and IP address.

2) Communication target selection box:

This selects NC card communication target

Alternative: MELDASMAGIC64, MELDAS6x5M, MELDAS6x5L, MELDASC6/C64, CNC C70, CNC 700M and CNC 700L.

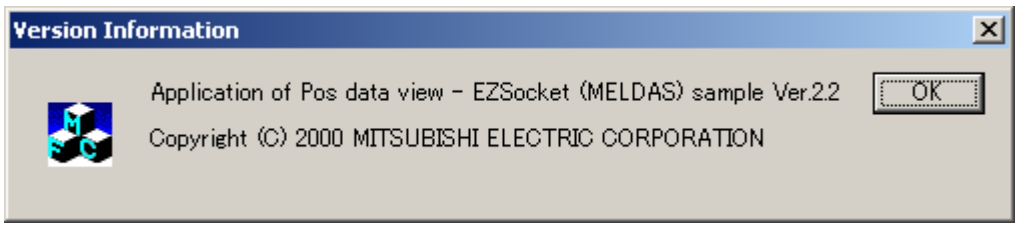
(6) Executable function

This connects/disconnects the communication with selected target.

When failing in connecting, this outputs the error messages in the message box.

(7) Version information display

"Help, Version information" displays the dialogue box that shows the version information of the position data display application.



6.2.6 Set project work space

The setting of project work space used to create the position data display application is as follows. The configuration of the application is in the table below.

Table 6-2 Project configuration

Setting items	Settings
Application type	SDI (Single Document Interface)
Data base support	Not supported
Automation support	Supported
OLE multiple document support	Not supported
The functions included in the application	Docking tool bar Initial status bar 3D control
MAPI support	Not supported
Windows socket support	Not supported
The number of the latest files to display	4 files (Default for high grade setting)

6.2.7 IEZNcCommunication object3

IEZNcCommunication3 object is the object to connect with communication lines.
The sample application uses the methods mentioned below.

Open2(): Method to open lines
Close(): Method to disconnect lines
SetHead(): Part system designation method

6.2.8 IEZNcPosition object

IEZNcPosition gets the position information of the opened NC card. The sample application uses the method mentioned below.

GetWorkPosition(): Method to get workpiece coordinate position
GetMachinePosition(): Method to get machine position
GetCurrentPosition: Method to get current position
GetDistance(): Method to get remaining command

6.3 Monitoring Application

Details of the sample application for Visual Basic Ver.6.0 using EZSocket are as follows.

6.3.1 Performance environment

The sample application performs in the following system configuration.

OS	WindowsNT 4.0, Windows2000 Windows95, 98, WindowsXP
Compiler	Microsoft Visual Basic Ver6.0
Main body	EZSocketNc
Controller	MELDASMAGIC64, M6x5M/L, MELDAS C6/C64, CNC 700M/L
H/W	A personal computer that can operate the OS, compiler, and controller mentioned above.

6.3.2 Installation and uninstallation

How to install and uninstall the sample application is as follows.
Refer to each manual about installing OS, VC++, and about operation of the H/W.

(1) Installation
The sample application is created in the sample folder when installing EZSocket itself.
The sample application has its source code and executable file within the project name folder. The sample application includes the project work space file for Visual C++ Ver. 5.5. By opening the corresponding project work space file, it is possible to open the project by Visual Baxis.

(2) Uninstallation
Delete the project name folder or sample folder to uninstall the sample application.

6.3.3 Executing sample application

Execute the sample application as follows.
The executable file is in a folder under the sample application folder. In the case of the monitoring application, execute EZNcAutSample.exe.
Refer to the following chapters about the monitoring application.
FYI, the sample application mentioned above is the monitor application of CNC. It is necessary to operate operation search, and cycle start etc. for the CNC. Refer to the Instruction Manual of each CNC.

6.3.4 Function list

This chapter is about the functions of sample application.
Monitoring application monitors current position or the NC program which is currently executed, and displays the result as counter.

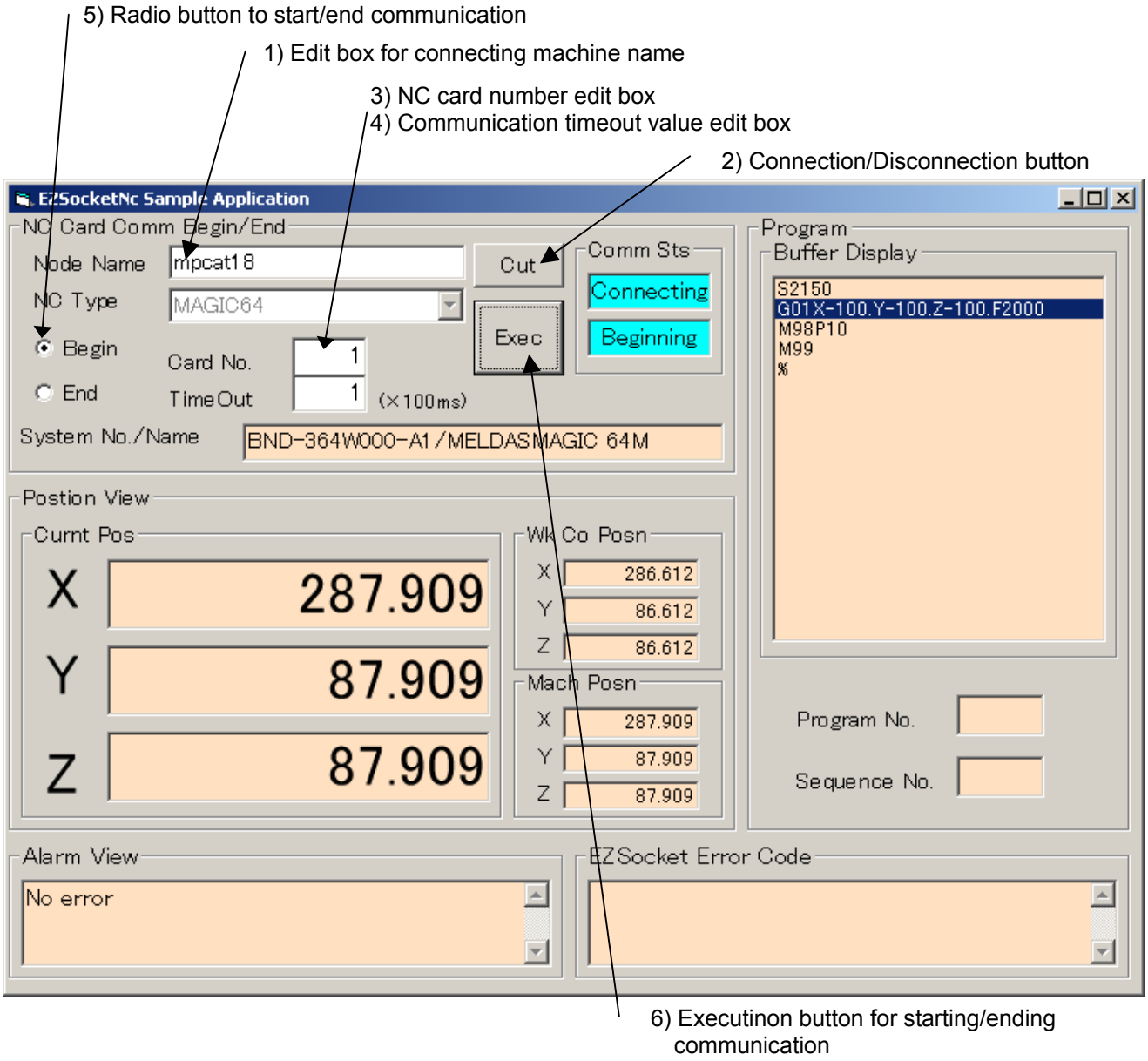
Table 6-3 Function list of the monitoring application.

Function items	Overview
NC card communication start/end	This sets NC card communication parameter and starts/ends communication. ・ Set NC card number ・ Set communication timeout period ・ Execute communication ・ Display NC system version
Position display	This reads the position data. ・ Display current coordinate position ・ Display workpiece coordinate position ・ Display machine coordinate position
Alarm display	This displays the alarms which are currently occurring.
Program display	This displays the program which is currently executed and the line currently executed. ・ Display the program list which is currently executed ・ Display current block position, and sequence number
EZSocket error display	This displays EZSocketAPI error code.

6.3.5 Screen configuration and functions

Screen configuration of position data display application and its functions are as follows.

(1) Basic screen configuration



Connected machine name edit box.

This designates the name of the machine on which the NC is mounted.

It is possible to designate domain name and IP address.

1) Connection/Disconnection button

This connects/disconnects to the designated machine.

2) NC card number edit box

This designates NC card number. NC card number is decided when setting up NC card

3) Communication timeout period edit box

This designates timeout amount when communicating with the NC card.

4) Radio button to start/end communication.

This starts/ends communicating with the NC card.

5) Execution button to start/end communication

After setting 3) to 6), communication is started/ended.

6.3.6 Setting project work space

How to set project work space used to create the position data display application is as follows. The project configuration of the application is mentioned below.
In this project, EZSocket's method is called by late binding.

Table 6-4 Project configuration

Items	Settings
Application type	Standard EXE
Adding standard module	Select EZNcDef.bas, EZNcErr.bas in [Project] - [Adding standard module]

6.4 Macro Sample Program

Details of the macro sample program that uses EZSocketOLE interface that can call EZSocket custom interface by Visual C++.

6.4.1 Performance environment

The sample application performs in the following system configuration.

OS	WindowsNT 4.0, Windows2000 Windows95, 98 (At least one WindowsNT4.0 is necessary on the network.)
Compiler	Microsoft Visual C++ Ver6.0
Main body	EZSocketNc
Controller	MELDASMAGIC64, MELDAS600M, MELDAS600L
H/W	A personal computer that can operate the OS, compiler and controller mentioned above. It also has to have an Ethernet port.

6.4.2 Installation and uninstallation

How to install and uninstall the sample application is as follows.
Refer to each manual about installing OS, VC++, and about operation of the H/W.

(1) Installation
The sample application is created in the sample folder when installing EZSocket itself.
The sample application has its source code and executable file within the project name folder. The sample application includes the project work space file for Visual C++ Ver. 6.0. By opening the corresponding project work space file, it is possible to open the project by Visual C++.
The EZSocket interface macro is saved in the include folder under this file.

(2) Uninstallation
Delete the project name folder or sample folder to uninstall the sample application.

6.4.3 Executing sample program

Execute the sample application as follows.
The execution file is in the Debug or Release folder in the sample program folder. In the case of the macro sample program, execute MacSmp.exe.
After executing, the PC will ask you the NC's host name (or IP address) at the console screen, so input it. Then, the PC will ask you the type of the NC, so select one from among the list and input the #. Push the Enter key to confirm each input every time after you input. If normally executed, the NC's version will be displayed on the console screen.
Refer to the source program of the sample program for how to use the EZSocketOLE interface macro. For details, refer to the source program (EZSocketNcOle.h, EZSocketNcOle.cpp) of the EZSocketOLE interface macro.
The source code of the macro sample program is in the chapter 7.3

7. CONSOLE PROGRAM SAMPLE

7.1 The Console Program to Connect to CNC C70 (Via Ethernet)

```
//
// Simple sample program of the console application
//
//      Copyright(C) 2008 MITSUBISHI ELECTRIC CORPORATION
#include "stdafx.h"

#include "stdio.h"
#include <locale.h>

// EZSocket header file
#include "EZSocketNc.h"
#include "EZSocketNcStr.h"
#include "EZSocketNcDef.h"

#include "EasysocketDef.h"

int main(int argc, char* argv[])
{
        HRESULT                 hr = S_OK;
        LONG                    lRet = 0;
        EZNCST_OPEN             stOpen;
        LPOLESTR                lpwszBuffer = NULL;
        HANDLE                  hSampleFile = NULL;
        BYTE*                   pbData = NULL;
        const DWORD             dwLength = 256;
        DWORD                   dwNumRead = 0;
        DWORD                   dwWrittenSize = 0;

        memset(&stOpen, 0x00, sizeof(stOpen));

        // Initialize COM
        hr = CoInitialize(NULL);
        if( S_OK != hr ){
                wprintf(L"Failed in CoInitialize!\n");
                return 0;
        }

        setlocale( LC_ALL, "Japanese" );
        IEZNcCommunication3     *pIEZNcCom = NULL;      // Communication object
        IEZNcFile5              *pIEZNcFile = NULL;     // File object

        //
        // Creat EZNcCommunication object
        // For how to set the 1st argument of CLSIDFromProgID(), refer to note of *2 in
        "1.8.1 Programming procedure with VC++ (1)".
        CLSID clsid;
        CLSIDFromProgID( L"EZSocketNc.EZNcCommunication", &clsid );
        hr = CoCreateInstance(clsid,
                        NULL,
                        CLSCTX_INPROC_SERVER,
                        IID_IEZNcCommunication3,
                        (void **)&pIEZNcCom);
        if( S_OK != hr )
        {
                wprintf(L"EZSocket is not installed!\n");
                goto END;
        }

        //
        // Create EZNcFile object
        //
        if(pIEZNcCom->QueryInterface(IID_IEZNcFile5, (void**)&pIEZNcFile) != S_OK){
                wprintf(L"EZSocket is not installed!\n");
                goto END;
        }

        //
        // Set Open parameters
        //
        stOpen.lNetworkNumber           = 0x01;
        stOpen.lStationNumber           = 0x01;
```

```
stOpen.lUnitNumber                      = 0x00;
stOpen.lConnectUnitNumber               = 0x00;
stOpen.lIONumber                        = 0x3E1;
stOpen.lCpuType                         = CPU_Q17NNCCPU;
stOpen.lUnitType                        = UNIT_QJ71E71;
stOpen.lPacketType                      = PACKET_PLC1;
stOpen.lProtocolType                    = PROTOCOL_UDPIP;
stOpen.lPortNumber                      = 0x00;
stOpen.lBaudRate                        = 0x00;
stOpen.lDataBits                        = 0x00;
stOpen.lParity                          = 0x00;
stOpen.lStopBits                        = 0x00;
stOpen.lControl                         = 0x00;
stOpen.lpcwszHostAddress                = L"10.20.123.12";
stOpen.lCpuTimeOut                      = 0x00;
stOpen.lTimeOut                         = 1000;
stOpen.lSumCheck                        = FALSE;
stOpen.lSourceNetworkNumber             = 0x01;
stOpen.lSourceStationNumber             = 0x04;
stOpen.lDestinationPortNumber           = 5001;
stOpen.lDestinationIONumber             = 0x00;
stOpen.lConnectChannelNumber            = 0x00;
stOpen.lMultiDropChannelNumber          = 0x00;
stOpen.lThroughNetworkType              = 0x00;
stOpen.lIntelligentPreferenceBit        = 0x00;
stOpen.lDidPropertyBit                  = 0x01;
stOpen.lDsidPropertyBit                 = 0x01;

hr = pIEZNcCom->SetMelsecProtocol(&stOpen, &lRet);
if( S_OK != hr ){
        wprintf(L"Can't SetMelsecProtocol! Error Code = 0x%x\n",lRet);
        goto END;
}

// Open IEZNcCommunication3
hr = pIEZNcCom->Open2(EZNC_SYS_MELDASC70, 1, 20, &lRet);
if( S_OK != hr ){
        wprintf(L"Can't Open2! Error Code = 0x%x\n",lRet);
        goto END;
}

//
// Search file
//
hr = pIEZNcFile->FindDir(L"M01:\\PRG\\USER\\", 0x00, &lpwszBuffer, &lRet);
if( S_OK != hr ){
        wprintf(L"Can't FindDir! Error Code = 0x%x\n",lRet);
        goto END;
}

while(lRet >= 1){
        if(wcscmp(lpwszBuffer, L"10.PRG") == 0){
                break;
        }
        CoTaskMemFree(lpwszBuffer);
        lpwszBuffer = NULL;
        hr = pIEZNcFile->FindNextDir(&lpwszBuffer, &lRet);
        if( S_OK != hr ){
                wprintf(L"Can't FindNextDir! Error Code = 0x%x\n",lRet);
                goto END;
        }
}
if(lRet == 0){
        wprintf(L"File is not found.\n");
        pIEZNcFile->ResetDir(&lRet);
        goto END;
}

hr = pIEZNcFile->ResetDir(&lRet);
if( S_OK != hr ){
        wprintf(L"Can't ResetDir! Error Code = 0x%x\n",lRet);
        goto END;
}

//
// Read file
//
```

```
	hr = pIEZNcFile->OpenFile3(L"M01:\\PRG\\USER\\10.PRG", EZNC_FILE_READ, &lRet);
	if( S_OK != hr ){
		wprintf(L"Can't OpenFile3! Error Code = 0x%x\n",lRet);
		goto END;
	}

	hSampleFile = ::CreateFile("C:\\SAMPLE.PRG",
					GENERIC_WRITE,
					FILE_SHARE_READ,
					NULL,
					OPEN_ALWAYS,
					FILE_ATTRIBUTE_NORMAL,
					NULL);
	if(hSampleFile == INVALID_HANDLE_VALUE ){
		wprintf(L"Can't create file.\n");
		goto END;
	}
	do{
		hr = pIEZNcFile->ReadFile2(dwLength, &pbData, &dwNumRead, &lRet);
		if( S_OK != hr ){
			wprintf(L"Can't ReadFile2! Error Code = 0x%x\n",lRet);
			pIEZNcFile->AbortFile2(&lRet);
			::CloseHandle(hSampleFile);
			goto END;
		}
		if(dwNumRead != 0){
			::WriteFile(hSampleFile,
				(LPCVOID)pbData,
				dwNumRead,
				&dwWrittenSize,
				NULL);
			CoTaskMemFree(pbData);
			pbData = NULL;
		}
	}while(dwLength == dwNumRead);
	::CloseHandle(hSampleFile);

	hr = pIEZNcFile->CloseFile2(&lRet);
	if( S_OK != hr ){
		wprintf(L"Can't CloseFile2! Error Code = 0x%x\n",lRet);
		goto END;
	}

END:
	if(lpwszBuffer != NULL){
		CoTaskMemFree(lpwszBuffer);
		lpwszBuffer = NULL;
	}
	if(pbData != NULL){
		CoTaskMemFree(pbData);
		pbData = NULL;
	}

	// Close IEZNcCommunication3
	if(pIEZNcCom != NULL){
		pIEZNcCom->Close(&lRet);
	}

	// Release object
	if(pIEZNcFile != NULL){
		pIEZNcFile->Release();
		pIEZNcFile = NULL;
	}
	if(pIEZNcCom != NULL){
		pIEZNcCom->Release();
		pIEZNcCom = NULL;
	}

	// Release COM library
	CoUninitialize();

	return 0;
}
```

7.2 The Console Program which Runs on the Computer on which MELDASMAGIC64 is Mounted

```
//
// Simple sample program of the console application
//
// Copyright(C) 2000, 2008 MITSUBISHI ELECTRIC CORPORATION
#include <stdio.h>
#include <locale.h>

// EZSocket header file
#include "EZSocketNc.h"
#include "EZSocketNcDef.h"
#include "EZSocketNcErr.h"

void main()
{
        // Initialize COM
        CoInitialize(NULL);

        // Result of COM
        HRESULT hr;
        long lRet;

        setlocale( LC_ALL, "Japanese" );
        IEZNcCommunication3*        pIEZNcCom = NULL;           // Communication object
        IEZNcSystem*                pIEZNcSys = NULL; // NC system object

        //
        // Create EZNcCommunication object
        //For how to set the 1st argument of CLSIDFromProgID(), refer to *2 of "1.7.1Programming procedure with VC++ (1)".
        CLSID clsid;
        hr = CLSIDFromProgID(L"EZSocketNc.EZNcCommunication",&clsid);
        hr = CoCreateInstance(clsid,
                                NULL,
                                CLSCTX_INPROC_SERVER,
                                IID_IEZNcCommunication3,
                                (void**)&pIEZNcCom );
        if( S_OK != hr ){
                wprintf(L" EZSocket is not installed!\n");
                return;
        }

        //
        // Create EZNcSystem object
        //
        hr = pIEZNcCom->QueryInterface(IID_IEZNcSystem,(void**)&pIEZNcSys );
        if( S_OK != hr ){
                wprintf(L" EZSocket is not installed!\n");
                return;
        }

        // Open IEZNcCommunication3
       long lNcType = EZNC_SYS_MAGICBOARD64;
        long lMachineNo = 1;
       long lTimeOut = 1;
        hr = pIEZNcCom->Open2(lNcType,lMachineNo,lTimeOut,&lRet);
        if( S_OK != hr ){
                wprintf(L"Can't open! Error Code = 0x%x\n",lRet);
                return;
        }
```

```
        // Get the NC system's version
        LPOLESTR lpwszVersion;
        long lAxisNo = 1;
        hr=pIEZNcSys->GetVersion( lAxisNo, 0 , &lpwszVersion, &lRet );

        if( S_OK != hr ){
                wprintf(L"Can't Get Version! Error Code = 0x%x\n",lRet);
                return;
        }else{
            wprintf(L"Success! NC System versiton is %s\n",lpwszVersion);
                if( lpwszVersion )
                        CoTaskMemFree(lpwszVersion);
        }

        // Close IEZNcCommunication3
        pIEZNcCom->Close(&lRet);

        // Release the object
        pIEZNcSys->Release();
        pIEZNcCom->Release();

        // Release COM library
        CoUninitialize();

        return;
}
```

7.3 The Console Program which Runs MELDASMAGIC64 From the Remote Personal Computer Connected by Ethernet

```
//
// Simple sample program of the console application
//
//      Copyright(C) 2000, 2008 MITSUBISHI ELECTRIC CORPORATION
#define _WIN32_DCOM
#include <atlbase.h>
#include <atlimpl.cpp>

#include <stdio.h>
#include <locale.h>

// EZSocket header file
#include "EZSocketNc.h"
#include "EZSocketNcDef.h"
#include "EZSocketNcErr.h"

void main()
{
        // Initialize COM
        CoInitialize(NULL);

        // Result of COM
        HRESULT hr;
        long lRet;

        setlocale( LC_ALL, "Japanese" );
        MULTI_QI qi;
        qi.hr = 0;
        qi.pIID = &IID_IUnknown;
        qi.pItf = 0; // IUnknown;

        COSERVERINFO csi = { 0,L"10.20.123.12",NULL,0};

        IEZNcCommunication3*        pIEZNcCom = NULL;        // Communication object
        IEZNcSystem*                pIEZNcSys = NULL;        // NC system object
```

Designate the IP address or domain to connect

```
	//
	// Create EZNcCommunication object
	//For how to set the 1st argument of CLSIDFromProgID(), refer to *2 of "1.7.1Programming procedure with VC++ (1)".
	CLSID clsid;
	hr = CLSIDFromProgID(L"EZSocketNc.EZNcCommunication",&clsid);
	hr = CoCreateInstanceEx(clsid,
					NULL,
					CLSCTX_REMOTE_SERVER,

					&csi,
					1,
					&qi );
	if( S_OK != hr ){
		wprintf(L"EZSocket is not installed!\n");
		return;
	}

	//
	// Create EZNcSystem object
	//
	hr = qi.pItf->QueryInterface(IID_IEZNcCommunication3,(void**)&pIEZNcCom );
	hr = pIEZNcCom->QueryInterface(IID_IEZNcSystem,(void**)&pIEZNcSys );
	if( S_OK != hr ){
		wprintf(L"EZSocket is not installed!\n");
		return;
	}

	// Open IEZNcCommunication3
	long lNcType = EZNC_SYS_MAGICBOARD64;
	long lMachineNo = 1;
	long lTimeOut = 1;
	hr = pIEZNcCom->Open2(lNcType,lMachineNo,lTimeOut,&lRet);
	if( S_OK != hr ){
		wprintf(L"Can't open! Error Code = 0x%x\n",lRet);
		return;
	}

	// Get the NC system's version
	LPOLESTR lpwszVersion;
	long lAxisNo = 1;
	hr=pIEZNcSys->GetVersion( lAxisNo, 0 , &lpwszVersion, &lRet );
	if( S_OK != hr ){
		wprintf(L"Can't Get Version! Error Code = 0x%x\n",lRet);
		return;
	}else{
		wprintf(L"Success! NC System versiton is %s\n",lpwszVersion);
		if( lpwszVersion )
			CoTaskMemFree(lpwszVersion);
	}

	// Close IEZNcCommunication3
	pIEZNcCom->Close(&lRet);

	// Release the object
	pIEZNcSys->Release();
	pIEZNcCom->Release();
	qi.pItf->Release();

	// Release COM library
	CoUninitialize();

	return;
}
```

7.4 The Macro Sample Program that Uses EZSocketOLE Interface Macro

```
/************************************************************** MELCO ***************/
/*                                                                                    */
/*        <FILENAME>          MacSmp.cpp                                              */
/*        EZSocket(MELDAS) Sample program that uses OLE interface macro              */
/*                                .DATE                    2001-11-02                 */
/*                                                                                    */
/* Copyright (C) 2001 MITSUBISHI Electric Corporation All Rights Reserved             */
/*************************************************************************************/
#include "stdafx.h"
//Header file necessary for COM/DCOM
//Note: To use DCOM, it is necessary to add WIN32_DCOM at the head of stdafx.h
#include <atlbase.h>
#include <atlimpl.cpp>

#include "MacSmp.h"

// EZSocket(MELDAS) header file
#include "EZSocketNc.h"
#include "EZSocketNcDef.h"
#include "EZSocketNcErr.h"
#include "EZSocketNcOle.h"              // Utilizing the macro

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// The only application object
CWinApp theApp;
using namespace std;
int _tmain(int argc, TCHAR* argv[], TCHAR* envp[])
{
        int nRetCode = 0;

        // Initialize MFC and output error in failure in initialization
        if (!AfxWinInit(::GetModuleHandle(NULL), NULL, ::GetCommandLine(), 0)) {
            // TODO: Change error codes upon necessity
            cerr << _T("Fatal Error: MFC initialization failed") << endl;
            nRetCode = 1;
        }else{
            // Initialize COM
            CoInitialize(NULL);

            HRESULT hr;
            long lRet;
            setlocale( LC_ALL, "Japanese" );

            IEZNcCommunication* pIEZNcCom = NULL;      // Communication object
            IEZNcSystem*        pIEZNcSys = NULL;      // NC system object

            // Input of the NC's host name
            wchar_t lpszHostName[256];
            wprintf(L"Input the CNC Controller's host name(or IP address) :");
            wscanf(L"%s",lpszHostName );

            // Create EZNcCommunication object my macro
            hr = CreateEZNcCommunication(lpszHostName, pIEZNcCom );
            if( S_OK == hr ){
                // Select the NC's type
                long lNcType;
                wprintf(L"Select the CNC type No.[1:MELDAS600M ,2:MELDAS600L, 3:MAGIC64] :");
                wscanf(L"%d",&lNcType);
                if( lNcType == 1 ) lNcType = EZNC_SYS_MELDAS6X5M;
                else if( lNcType == 2 ) lNcType = EZNC_SYS_MELDAS6X5L;
                else lNcType = EZNC_SYS_MAGICBOARD64;
```

```
            // Open EZNcCommunication
            long lMachineNo = 1;
            long lTimeOut = 1;
            pIEZNcCom->Open(lNcType,lMachineNo,lTimeOut,&lRet);
            if( S_OK != lRet ){
                wprintf(L"Can't open! Error Code = 0x%x\n",lRet);
                nRetCode = 1;
            }else{
                // Create EZNcSystem object by macro
                hr = CreateEZNcSystem( pIEZNcCom , pIEZNcSys );
                if( S_OK != hr ){
                        wprintf(L"Can't Create EZNcSystem!\n");
                        nRetCode = 1;
                }else{
                        // Get the NC's version
                        LPOLESTR lpwszVersion;
                        long lAxisNo = 1;

                        pIEZNcSys->GetVersion( lAxisNo, 0 , &lpwszVersion, &lRet );
                        if( S_OK != lRet ){
                            wprintf(L"Can't Get Version! Error Code = 0x%x\n",lRet);
                            nRetCode = 1;
                        }else{
                            wprintf(L"Success! NC System versiton is %s\n",lpwszVersion);
                            if( lpwszVersion ) CoTaskMemFree(lpwszVersion);
                        }

                        // Release EZNcSystem object
                        pIEZNcSys->Release();
                        pIEZNcSys = NULL;
                }
                // Close EZNcCommunication
                pIEZNcCom->Close(&lRet);
            }
            // Release EZNcCommunication object
            pIEZNcCom->Release();
            pIEZNcCom = NULL;
        }
        // Release COM library
        CoUninitialize();
    }
    return nRetCode;
```

8. RESCTRICTION

8.1 Restriction in Performance on EZSocket Server

In the case of Magic64, due to the restrictions of the OS specifications, the same ISA bus cannot be shared between the EZSocket application installed as the NT service process and the EZSocket application installed as the usual user process. If you try a connection from a EZSocket product installed as the user process to a FA device via the ISA bus used by the EZSocket application installed as the NT service process, a failure may occur.

Index

A

Adding tool # in the tool life management group.. 2-138
Adding tool life management group # 2-134

B

Batch-reading of data..... 2-188
Batch-writing of data 2-189

C

Changing tool # of tool life management..... 2-139
Changing tool life management group # 2-135
Closing file 2-101
Closing file compulsorily 2-102
Closing line 2-11
Closing machining program..... 2-107
Closing machining program compulsorily..... 2-108

D

Deleting all data setting 2-187
Deleting all device setting 2-179
Deleting file 2-91
Deleting tool # of tool life management..... 2-141
Deleting tool life management group # 2-136

F

File Copy 2-89
Finding directory..... 2-85
Finding the next directory 2-87
Finishing the directory search..... 2-88

G

Getting alarm information..... 2-27
Getting common variable name..... 2-114
Getting common variable null data..... 2-116
Getting modal communication condition 2-23
Getting NC System # and name 2-24
Getting NC system information 2-26
Getting the absolute position monitor information 2-72
Getting the ATC ready tool # 2-158
Getting the ATC user PLC interface 2-165
Getting the automatic operation time..... 2-52
Getting the automatic start time..... 2-54
Getting the auxiliary axis diagnosis information ... 2-76
Getting the auxiliary axis monitor information 2-74
Getting the auxiliary axis version information 2-77
Getting the control parameter for ATC tool register 2-155
Getting the current position 2-30
Getting the cutting mode 2-81
Getting the data from workpiece offset file 2-205
Getting the date/time..... 2-48
Getting the drive free capacity 2-94
Getting the drive information 2-93
Getting the external elapsed time..... 2-56
Getting the feedrate 2-33
Getting the feedrate command value..... 2-39
Getting the G-code modal command value 2-36
Getting the invalid status..... 2-79
Getting the local variable null data 2-119
Getting the M/S/T/B command modal value..... 2-40
Getting the machine position 2-29
Getting the macro sub program calling level.....2-118
Getting the manual interruption amount..... 2-34
Getting the next travel distance..... 2-32
Getting the number of common variable sets.....2-113
Getting the number of pots of each ATC magazine 2-157
Getting the number of tool offset sets 2-120
Getting the operation command status..... 2-80
Getting the operation status 2-84
Getting the parameters 2-167
Getting the part system..... 2-13
Getting the power ON time 2-50
Getting the power supply diagnosis information ... 2-65
Getting the power supply version information..... 2-64
Getting the program # 2-43
Getting the program information..... 2-47
Getting the program position 2-35
Getting the reference surface level 2-129
Getting the remaining distance..... 2-31
Getting the remaining dwell time 2-78
Getting the servo axis status..... 2-82
Getting the servo diagnosis information..... 2-62
Getting the servo information 2-61
Getting the spare tool exchange data of tool life management file..... 2-201
Getting the spare tool for tool life management... 2-151
Getting the spindle diagnosis information 2-70
Getting the spindle version information..... 2-69
Getting the subprogram level..... 2-46
Getting the tool # of each ATC magazine pot..... 2-160
Getting the tool # of the ATC extended magazine pot 2-163
Getting the tool # of the ATC magazine pot..... 2-159
Getting the tool life management data 2-142
Getting the tool life management data of tool life management file..... 2-191
Getting the tool life management data of tool life management file2.....2-196
Getting the tool life management group # list..... 2-133
Getting the tool life management type 2-131

Getting the tool list in the tool life management group 2-137
Getting the tool offset #..... 2-38
Getting the tool offset type 2-121
Getting the tool offset value..... 2-122
Getting the total number of the ATC magazine pot sets 2-156
Getting the workpiece coordinate offset..... 2-126
Getting the workpiece coordinate position..... 2-28

I

IEZNcATC2
GetMGNAux..... 2-165
GetMGNControl 2-155
GetMGNPot 2-159
GetMGNPot3 2-160
GetMGNPotEx 2-163
GetMGNReady 2-158
GetMGNSize..... 2-156
GetMGNSize2..... 2-157
SetMGNAux 2-166
SetMGNPot 2-161
SetMGNPot3 2-162
SetMGNPotEx 2-164
IEZNcAxisMonitor
GetAbsPositionMonitor..... 2-72
GetAuxAxisDiagnosis 2-76
GetAuxAxisMonitor 2-74
GetAuxAxisVersion 2-77
GetDowelTime 2-78
GetPowerDiagnosis 2-65
GetPowerVersion 2-64
GetServoDiagnosis 2-62
GetServoMinitor..... 2-58
GetServoVersion 2-61
GetSpindleDiagnosis..... 2-70
GetSpindleMonitor..... 2-67
GetSpindleVersion..... 2-69
IEZNcCommand2
GetCommand2..... 2-40
GetFeedCommand..... 2-39
GetGCodeCommand..... 2-36
GetToolCommand..... 2-38
SetCommand2 2-41
IEZNcCommonVariable2
CommonVRead.....2-111
CommonVWrite 2-112
GetCVNullData 2-116
GetName..... 2-114
GetSize..... 2-113
SetName 2-115
IEZNcCommunication3
Close..... 2-11
GetHead..... 2-13
GetModalCondition 2-23
Open2 2-9
SetHead 2-12
SetMelsecProtocol 2-15
SetModalCondition 2-22
SetTCPIPProtocol 2-14
IEZNcDevice
DeleteDeviceAll..... 2-179
ReadDevice 2-180
SetDevice 2-174
WriteDevice 2-181
IEZNcFile5
AbortFile2..... 2-102
AbortNCFile2 2-108
CloseFile2 2-101
CloseNCFile2 2-107
Copy2 2-89
Delete2..... 2-91
FindDir 2-85
FindNextDir 2-87
GetDriveInformation 2-93
GetDriveSize 2-94
OpenFile3 2-95
OpenNCFile2..... 2-105
ReadFile2 2-103
ReadNCFile2 2-109
Rename2..... 2-92
ResetDir..... 2-88
WriteFile..... 2-104
WriteNCFile 2-110
IEZNcGeneric2
DeleteDataAll..... 2-187
ReadBlockData..... 2-188
ReadData..... 2-182
SetData..... 2-185
WriteBlockData..... 2-189
WriteData 2-184
IEZNcLocalVariable2
GetLVNullData 2-119
GetMacroLevel 2-118
LocalVRead 2-117
IEZNcOperation
Run 2-172
Search..... 2-171
Stop..... 2-173
IEZNcParameter2
GetParameterData..... 2-167
SetParameterData 2-169
IEZNcPosition
GetCurrentPosition..... 2-30
GetDistance..... 2-31
GetFeedSpeed 2-33
GetMachinePosition..... 2-29
GetManualOverlap 2-34
GetNextDistance..... 2-32
GetProgramPosition 2-35

GetWorkPosition 2-28
IEZNcProgram2
CurrentBlockRead..... 2-42
GetBlockNumber..... 2-45
GetInformation..... 2-47
GetProgramNumber2..... 2-43
GetSequenceNumber 2-44
GetSubProLevel 2-46
IEZNcRunStatus
GetAxisStatus 2-82
GetCommandStatus..... 2-80
GetCuttingMode..... 2-81
GetInvalidStatus..... 2-79
GetRunStatus..... 2-84
IEZNcSubFunction2
ChangeInit..... 2-190
GetSpareToolOfFile..... 2-201
GetToolLifeValueOfFile..... 2-191
GetToolLifeValueOfFile2..... 2-196
GetToolWorkOffsetOfFile..... 2-205
SetSpareToolOfFile 2-203
SetToolLifeValueOfFile 2-193
SetToolLifeValueOfFile2 2-199
SetToolWorkOffsetOfFile 2-207
IEZNcSystem
GetAlarm 2-27
GetSystemInformation..... 2-26
GetVersion 2-24
IEZNcTime
GetAliveTime..... 2-50
GetClockData 2-48
GetEstimateTime 2-56
GetRunTime 2-52
GetStartTime..... 2-54
SetAliveTime 2-51
SetClockData..... 2-49
SetEstimateTime..... 2-57
SetRunTime..... 2-53
SetStartTime 2-55
IEZNcTool3
AddToolLifeGroup 2-134
AddToolLifeToolNo 2-138
ChangeToolLifeGroup 2-135
ChangeToolLifeToolNo 2-139
DeleteToolLifeGroup 2-136
DeleteToolLifeToolNo 2-141
GetOffset..... 2-122
GetSpareTool 2-151
GetSurface 2-129
GetToolLifeGroupList 2-133
GetToolLifeToolNoList 2-137
GetToolLifeType2 2-131
GetToolLifeValue 2-142
GetToolSetSize..... 2-120
GetToolWorkOffset 2-126
GetType 2-121
SetOffset..... 2-124
SetSpareTool 2-153
SetSurface 2-130
SetToolLifeType2..... 2-132
SetToolLifeValue 2-145
SetToolLifeValue2 2-148
SetToolWorkOffset 2-127
Initializing sub function 2-190

O

Opening file 2-95
Opening line 2-9
Opening machining program..... 2-105

R

Reading common variable 2-111
Reading file 2-103
Reading machining program 2-109
Reading the block #..... 2-45
Reading the current block 2-42
Reading the device 2-180
Reading the generic data 2-182
Reading the local variable 2-117
Reading the sequence # 2-44
Renaming file 2-92
Respective setting of tool life management data .. 2-145

S

Search 2-171
Servo monitor 2-58
Setting automatic operation time 2-53
Setting common variable name 2-115
Setting data 2-185
Setting data to workpiece offset file 2-207
Setting date and time 2-49
Setting device 2-174
Setting external elapsed time..... 2-57
Setting manual command (M/S/T/B)..... 2-41
Setting Melsec communication..... 2-15
Setting modal communication condition..... 2-22
Setting parameters 2-169
Setting part system..... 2-12
Setting power ON time 2-51
Setting reference surface level 2-130
Setting spare tool for tool life management..... 2-153
Setting TCP/IP communication 2-14
Setting the ATC user PLC interface..... 2-166
Setting the automatic start time..... 2-55
Setting the spare tool exchange data of tool life management file..... 2-203
Setting the tool # of the ATC magazine pot 2-161
Setting the tool life management data of tool life management file..... 2-193, 2-199

Setting tool # of the ATC extended magazine pot . 2-164
Setting tool # of the pot of each ATC magazine 2-162
Setting tool life management data 2-148
Setting tool life management type 2-132
Setting tool offset value 2-124
Setting workpiece coordinate offset..... 2-127
Spindle monitor..... 2-67
Starting PLC program 2-172
Stopping PLC program 2-173

W

Writing common variable 2-112
Writing device 2-181
Writing file 2-104
Writing generic data 2-184
Writing machining program 2-110

Revision History

Date of revision	Manual No.	Revision details
Jun. 2006	IB(NA)1500195-A	First edition created.
Jun. 2007	IB(NA)1500195-B	・Amended according to the specifications of EZSocket ver.2.80. ・Mistakes were corrected.
May. 2008	IB(NA)1500195-C	・Amended according to the specifications of EZSocket ver.2.86. ・Mistakes were corrected.
Oct. 2008	IB(NA)1500195-D	・Amended according to the specifications of EZSocket ver.2.87. ・Mistakes were corrected.
Apr.2009	IB(NA)1500195-E	Amended according to the specifications of EZSocket ver.2.88 1. Corrections relating to CNC C70 (1) Added the methods that are supported by Mitsubishi CNC C70 Series. (2) Added the function of communicating via QnUDE. 2. Correction common to all the models (1) Added an example of releasing areas to ReadBlockData().

Global Service Network

AMERICA

MITSUBISHI ELECTRIC AUTOMATION INC. (AMERICA FA CENTER)

Central Region Service Center
500 CORPORATE WOODS PARKWAY, VERNON HILLS, IL., 60061, U.S.A.
TEL: +1-847-478-2500 / FAX: +1-847-478-2650

Western Michigan Service Satellite
ALLEGAN, MICHIGAN., 49010, U.S.A.
TEL: +1-847-478-2500 / FAX: +1-269-673-4092

Eastern Michigan Service Satellite
TEL: +1-847-478-2500 / FAX: +1-847-478-2650

Ohio Service Satellite
LIMA, OHIO, 45801, U.S.A.
TEL: +1-847-478-2500 / FAX: +1-847-478-2650

Minnesota Service Satellite
RICHFIELD, MINNESOTA, 55423, U.S.A.
TEL: +1-847-478-2500 / FAX: +1-847-478-2650

Western Region Service Center
5665 PLAZA DRIVE, CYPRESS, CALIFORNIA, 90630, U.S.A.
TEL: +1-714-220-4796 / FAX: +1-714-229-3818

North CA Service Satellite
PORTLAND, OREGON, 97086, U.S.A.
TEL: +1-714-220-4796 / FAX: +1-714-229-3818

Eastern Region Service Center
200 COTTONTAIL LANE
SOMERSET, NEW JERSEY, 08873, U.S.A.
TEL: +1-732-560-4500 / FAX: +1-732-560-4531

Western Pennsylvania Service Satellite
ERIE, PENNSYLVANIA, 16510, U.S.A.
TEL: +1-814-897-7820 / FAX: +1-814-987-7820

Southern Region Service Center
2810 PREMIERE PARKWAY SUITE 400, DULUTH, GEORGIA, 30097, U.S.A.
TEL: +1-678-258-4500 / FAX: +1-678-258-4519

Northern Texas Service Satellite
1000, NOLEN DRIVE SUITE 200, GRAPEVINE, TEXAS, 76051, U.S.A.
TEL: +1-817-251-7468 / FAX: +1-817-416-5000

Southern Texas Service Satellite
FRIENDSWOOD, TEXAS, 77546, U.S.A
TEL: +1-832-573-0787 / FAX: +1-678-573-8290

Central Florida Service Satellite
SATELITE BEACH, FLORIDA, 32937, U.S.A.
TEL : +1-321-610-4436 / FAX : +1-321-610-4437

Canadian Region Service Center
4299 14TH AVENUE MARKHAM, ONTARIO, L3R OJ2, CANADA
TEL: +1-905-475-7728 / FAX: +1-905-475-7935

Mexico City Service Center
MARIANO ESCOBEDO 69 TLALNEPANTLA, 54030
EDO. DE MEXICO
TEL: +52-55-9171-7662 / FAX: +52-55-9171-7649

Monterrey Service Satellite
ARGENTINA 3900, FRACC. LAS TORRES, MONTERREY, N.L., 64720, MEXICO
TEL: +52-81-8365-4171 / FAX: +52-81-8365-4171

Brazil Service Center
ACESSO JOSE SARTORELLI, KM 2.1 CEP 18550-000, BOITUVA-SP, BRAZIL
TEL: +55-15-3363-9900 / FAX: +55-15-3363-9911

Brazilian's Sites Service Center
CITIES OF PORTO ALEGRE AND CAXIAS DO SUL BRAZIL
CITIES OF SANTA CATARINA AND PARANA STATES
TEL: +55-15-3363-9927

EUROPE

MITSUBISHI ELECTRIC EUROPE B.V. (EUROPE FA CENTER)
GOTHAER STRASSE 10, 40880 RATINGEN, GERMANY
TEL: +49-2102-486-0 / FAX: +49-2102-486-5910

Germany Service Center
KURZE STRASSE. 40, 70794 FILDERSTADT-BONLANDEN, GERMANY
TEL: + 49-711-3270-010 / FAX: +49-711-3270-0141

France Service Center
25, BOULEVARD DES BOUVETS, 92741 NANTERRE CEDEX FRANCE
TEL: +33-1-41-02-83-13 / FAX: +33-1-49-01-07-25

France (Lyon) Service Satellite
120, ALLEE JACQUES MONOD 69800 SAINT PRIEST
TEL: +33-1-41-02-83-13 / FAX: +33-1-49-01-07-25

Italy Service Center
VIALE COLLEONI 7-PALAZZO SIRIO CENTRO DIREZIONALE COLLEONI,
20041 AGRATE BRIANZA MILANO ITALY
TEL: +39-039-60531-342 / FAX: +39-039-6053-206

Italy (Padova) Service Sattelite
VIA SAVELLI 24- 35129 PADOVA ITALY
TEL: +39-039-60531-342 / FAX: +39-039-6053-206

U.K. Service Center
TRAVELLERS LANE, HATFIELD, HERTFORDSHIRE, AL10 8XB, U.K.
TEL: +44-1707-27-6100 / FAX: +44-1707-27-8992

Spain Service Center
CTRA. DE RUBI, 76-80-APDO. 420
08190 SAINT CUGAT DEL VALLES, BARCELONA SPAIN
TEL: +34-935-65-2236 / FAX: +34-935-89-1579

Poland Service Center
UL.KRAKOWSKA 50, 32-083 BALICE, POLAND
TEL: +48-12-630-4700 / FAX: +48-12-630-4727

Poland (Wroclaw) Service Center
UL KOBIERZYCKA 23, 52-315 WROCLAW, POLAND
TEL: +48-71-333-77-53 / FAX: +48-71-333-77-53

Turkey Service Center
DARULACEZE CAD. FAMAS IS MERKEZI A BLOK No.43 KAT 2 80270 OKMEYDANI
ISTANBUL, TURKEY
TEL: +90-212-320-1640 / FAX: +90-212-320-1649

Czech Republic Service Center
TECHNOLOGICKA 374/6, 708 00 OSTRAVA-PUSTKOVEC, CZECH REPUBLIC
TEL: +420-59-5691-185 / FAX: +420-59-5691-199

Russia Service Center
213, B.NOVODMITROVSKAYA STR., 14/2, 127015 MOSCOW, RUSSIA
TEL: +7-495-748-0191 / FAX: +7-495-748-0192

Sweden Service Center
STRANDKULLEN, 718 91 FROVI, SWEDEN
TEL: +46-581-700-20 / FAX: +46-581-700-75

Bulgaria Service Center
4 A. LYAPCHEV BOUL., 1756 - SOFIA, BULGARIA
TEL: +359-2-8176000 / FAX: +359-2-9744061

Ukraine (Kharkov) Service Center
APTEKARSKIY PEREULOK 9-A, OFFICE 3, 61001 KHARKOV, UKRAINE
TEL: +38-57-732-7744 / FAX: +38-57-731-8721

Ukraine (Kiev) Service Center
4-B, M. RASKOVOYI STR., 02660 KIEV, UKRAINE
TEL: +38-044-494-3355 / FAX: +38-044-494-3366

Belarus Service Center
703, OKTYABRSKAYA STR., 16/5, 220030 MINSK, BELARUS
TEL: +375-17-210-4626 / FAX: +375-17-227-5830

ASEAN

MITSUBISHI ELECTRIC ASIA PTE. LTD. (ASEAN FA CENTER)

Singapore Service Center
307 ALEXANDRA ROAD #05-01/02 MITSUBISHI ELECTRIC BUILDING SINGAPORE 159943
TEL: +65-6473-2308 / FAX: +65-6476-7439

Indonesia Service Center
WISMA NUSANTARA 14TH FLOOR JL. M.H. THAMRIN 59, JAKARTA 10350 INDONESIA
TEL: +62-21-3917-144 / FAX: +62-21-3917-164

Malaysia (KL) Service Center
60, JALAN USJ 10 /1B 47620 UEP SUBANG JAYA SELANGOR DARUL EHSAN, MALAYSIA
TEL: +60-3-5631-7605 / FAX: +60-3-5631-7636

Malaysia (Johor Baru) Service Center
No.16, JALAN SHAH BANDAR 1, TAMAN UNGKU TUN AMINAH, 81300 SKUDAI, JOHOR MALAYSIA
TEL: +60-7-557-8218 / FAX: +60-7-557-3404

Vietnam Service Center-1
47-49 HOANG SA ST. DAKAO WARD, DIST. 1, HO CHI MINH CITY, VIETNAM
TEL: +84-8-910-4763 / FAX: +84-8-910-2593

Vietnam Service Center-2
THUAN KIEN PLAZA 190 HONG BANG ROAD. TOWER C. SUITE 3002. DIST. 5,
HO CHI MINH CITY, VIETNAM
TEL: +84-8-240-3587 / FAX: +84-8-726-7968

Vietnam (Hanoi) Service Center
5TH FL., 93B KIM LIEN ST., PHUONG LIEN WARD, DONG DA DIST.
TEL: +84-8-573-7646 / FAX: +84-4-573-7650

Philippines Service Center
UNIT No.411, ALABAMG CORPORATE CENTER KM 25. WEST SERVICE ROAD
SOUTH SUPERHIGHWAY, ALABAMG MUNTINLUPA METRO MANILA, PHILIPPINES 1771
TEL: +63-2-807-2416 / FAX: +63-2-807-2417

MITSUBISHI ELECTRIC AUTOMATION (THAILAND) CO., LTD. (THAILAND FA CENTER)
BANG-CHAN INDUSTRIAL ESTATE No.111 SOI SERITHAI 54
T.KANNAYAO, A.KANNAYAO, BANGKOK 10230, THAILAND
TEL: +66-2906-8255 / FAX: +66-2906-3239

Thailand Service Center
898/19, 20, 21, 22 S.V. CITY BUILDING OFFICE TOWER 1 FLOOR 7
RAMA RD BANGPONGPANG, YANNAWA, BANGKOK 10120. THAILAND
TEL: +66-2-682-6522 / FAX: +66-2-682-9750

INDIA

MITSUBISHI ELECTRIC ASIA PVT LTD.
FIRST & SECOND FLOOR, AVR BASE, MUNICIPAL No.BC-308,
HENNURE BANASWADI ROAD, HRBR RING ROAD, BANGALORE-560 043, INDIA
TEL: +91-80-4020-1600 / FAX: +91-80-4020-1699

India (Pune) Service Center
EL-3, J BLOCK, M.I.D.C., BHOSARI PUNE 411026, INDIA
TEL: +91-20-2710-2000 / FAX: +91-20-2710-2185

India (Bangalore) Service Center
S 615, 6TH FLOOR, MANIPAL CENTER, BANGALORE 560001, INDIA
TEL: +91-80-509-2119 / FAX: +91-80-532-0480

India (Delhi) Service Center
1197, SECTOR 15 PART-2, OFF DELHI-JAIPUR HIGHWAY BEHIND 32nd MILESTONE
GURGAON 122001, INDIA
TEL: +91-98-1024-8895

OCEANIA

MITSUBISHI ELECTRIC AUSTRALIA LTD.

Oceania Service Center
348 VICTORIA ROAD, RYDALMERE, N.S.W. 2116, AUSTRALIA
TEL: +61-2-9684-7269 / FAX: +61-2-9684-7245

CHINA

MITSUBISHI ELECTRIC AUTOMATION (SHANGHAI) LTD. (CHINA FA CENTER)

China (Shanghai) Service Center
4/F ZHI FU PLAZA, NO. 80 XIN CHANG ROAD,
SHANGHAI 200003, CHINA
TEL: +86-21-2322-3030 / FAX: +86-21-2322-3000

China (Beijing) Service Center
9/F, OFFICE TOWER1, HENDERSON CENTRE, 18 JIANGUOMENNEI AVENUE
DONGCHENG DISTRICT, BEIJING, CHINA 100005
TEL: +86-10-6518-8830 / FAX: +86-10-6518-8030

China (Tianjin) Service Center
B-2-801-802, YOUYI BUILDING. 50 YOUYI ROAD, HEXI DISTRICT,
TIANJIN, CHINA 300061
TEL: +86-22-2813-1015 / FAX: +86-22-2813-1017

China (Chengdu) Service Center
BLOCK B-1, 23F, CHUAN XIN MANSION, 18 SECTION 2,
RENMIN ROAD SOUTH, CHENGDU, SICHUAN, CHINA 610016
TEL: +86-28-8619-9730 / FAX: +86-28-8619-9805

China (Shenzhen) Service Center
ROOM 2512-2516, GREAT CHINA INTERNATIOANL EXCHANGE SQUARE, JINTIAN RD.S.,
FUTIAN DISTRICT, SHENZHEN, CHINA 518034
TEL: +86-755-2399-8272 / FAX: +86-755-8218-4776

KOREA

MITSUBISHI ELECTRIC AUTOMATION KOREA CO., LTD. (KOREA FA CENTER)

Korea Service Center
1480-6, GAYANG-DONG, GANGSEO-GU SEOUL 157-200, KOREA
TEL: +82-2-3660-9607 / FAX: +82-2-3663-0475

Korea Busan Service Satellite
#405 BUSAN INDUSTRIAL SUPPLIES MARKET BLDG, 578 KWAEBOP-DONG, SASANG-GU,
BUSAN 617-726, KOREA
TEL: +82-51-319-3747 / FAX: +82-51-319-3768

TAIWAN

MITSUBISHI ELECTRIC TAIWAN CO., LTD. (TAIWAN FA CENTER)

TAIWAN (Taichung) Service Center
No.8-1, GONG YEH 16TH RD., TAICHUNG INDUSTRIAL PARK TAICHUNG CITY, TAIWAN R.O.C
TEL: +886-4-2359-0688 / FAX: +886-4-2359-0689

TAIWAN (Taipei) Service Center
3TH. FLOOR, No.122 WUKUNG 2ND RD., WU-KU HSIANG, TAIPEI HSIEN, TAIWAN R.O.C
TEL: +886-2-2299-2205 / FAX: +886-2-2298-1909

TAIWAN (Tainan) Service Center
2F (C), 1-1, CHUNGHWA-RD., YONGKANG CITY, TAINAN HSIEN, TAIWAN R.O.C
TEL: +886-6-313-9600 / FAX: +886-6-313-7713

Notice

Every effort has been made to keep up with software and hardware revisions in the contents described in this manual. However, please understand that in some unavoidable cases simultaneous revision is not possible.
Please contact your Mitsubishi Electric dealer with any questions or comments regarding the use of this product.

Duplication Prohibited

This manual may not be reproduced in any form, in part or in whole, without written permission from Mitsubishi Electric Corporation.

© 2006-2009 MITSUBISHI ELECTRIC CORPORATION
ALL RIGHTS RESERVED.

MITSUBISHI CNC

MITSUBISHI ELECTRIC CORPORATION
HEAD OFFICE : TOKYO BUILDING,2-7-3 MARUNOUCHI,CHIYODA-KU,TOKYO 100-8310,JAPAN

MODEL	EZSocket
MODEL CODE	008—423
Manual No.	IB-1500195(ENG)

IB(NA)1500195(ENG)-E(0904)MEE

Specifications subject to change without notice.