Changes for the Better

MITSUBISHI ELECTRIC

MITSUBISHI Communication Software for CNC

# FCSB1224W000

# Reference Manual

# INTRODUCTION

Thank you for purchasing the Mitsubishi CNC communication software FCSB1224W000. This user's reference manual describes how to use the OLE/COM interface of FCSB1224W000.

Read this manual before use to get familiar with and correctly use the functions of FCSB1224W000.

# Precautions for Safety

(Read carefully before use.)

Before using the product, read the user's reference manual and other related manuals. Pay careful attention to safety when using the product.
The safety instructions in this manual are intended for this product. Do not use this product until you have a full knowledge of general and safety information and precautions about the computerized numerical controller.
In this manual, the safety instruction levels are classified into "WARNING" and "CAUTION".

⚠ DANGER — When there is a great risk that the user could be subject to fatalities or serious injuries if handling is mistaken.

⚠ CAUTION — When the user could be subject to medium or slight injuries or when physical damage could occur if handling is mistaken.

Note that even items ranked as ⚠ CAUTION, may lead to major results depending on the situation. In any case, important information that must always be observed is described.
Keep this manual in a safe place for future reference.

## [Mechanical precautions] ⚠ DANGER

- When connecting the product with the computerized numerical controller, consider the risk of external power supply failure and computer malfunction, and install the external safety circuit as fail-safe of the entire system.
- There is a risk of accident due to output error or malfunction.
- Writing to the computerized numerical controller will directly be reflected in machine control.
- Input error of setup or other parameter may cause accidental operation.
- Check all things before execution.

## [Startup and maintenance precautions] ⚠ CAUTION

- Operation error may cause machine damage or accident.
- Some functions may be different or unavailable depending on the version of the computerized numerical controller.

# Trademarks

MELDAS, MELSEC, EZSocket, EZMotion, iQ Platform, MELSOFT, GOT, CC-Link, CC-Link/LT and CC-Link IE are either trademarks or registered trademarks of Mitsubishi Electric Corporation in Japan and/or other countries.

Ethernet is a registered trademark of Xerox Corporation in the United States and/or other countries.
Microsoft®, Windows, Visual C++® and Visual Basic® are either trademarks or registered trademarks of Microsoft Corporation in the United States and/or other countries.
CompactFlash and CF are either trademarks or registered trademarks of SanDisk Corporation in the United States and/or other countries.
SD Logos are trademarks or registered trademarks of SD-3C, LLC in the United States, other countries or both.

Other company and product names that appear in this manual are trademarks or registered trademarks of the respective companies.

# CONTENTS

## 1. OUTLINE

## 2. I/F DETAILED SPECIFICATIONS

## 3. ERROR CODE LIST

## 4. API OPERATING PROCEDURE

## 5. APPLICATION INSTALLATION PROCEDURE

## 6.SAMPLE APPLICATION

## 7. CONSOLE PROGRAM SAMPLE

# 1. OUTLINE

The Mitsubishi CNC communication software FCSB1224W000 is designed to help development of an application with Windows interface for Mitsubishi computerized numerical controller CNCM700/M800 series and CNC C70 series. (Hereinafter referred to as the product.)

The product can accelerate development by eliminating necessity to know about internal processing of the computerized numerical controller and enabling use of the common OLE interface on the Mitsubishi computerized numerical controller.

## 1.1 Features

- Functions of Mitsubishi CNC M700/M800 series and CNC C70 series can be used on the Windows application with VC++, VB or VBA macro language.
- Communication and other complex processing with Mitsubishi CNC M700/M800 series and CNC C70 series will be conducted by the product so that the user can focus on development of the value-added Windows application.
- As the product will be upgraded in accordance with new models in the future, upgrading of the user-created application will also be easy.

## 1.2 Applicable Models

The product is applicable to the following models. Check compatibility before use.

- Mitsubishi CNC M700 series (M700/M700V series, M70/M70V series and E70) (hereinafter referred to as M700)
- Mitsubishi CNC M800 series (M800/M80 series) (hereinafter referred to as M800)
- Mitsubishi CNC C70 (hereinafter referred to as C70)

## 1.3 Connection Configuration

This section explains about connection configuration between the Mitsubishi computerized numerical controller and personal computer. Prepare the computerized numerical controller, personal computer, cable and other necessary equipment for communication using this product. For connection of equipment, see the instruction manual of the computerized numerical controller used.

### 1.3.1 Connection with M700 series

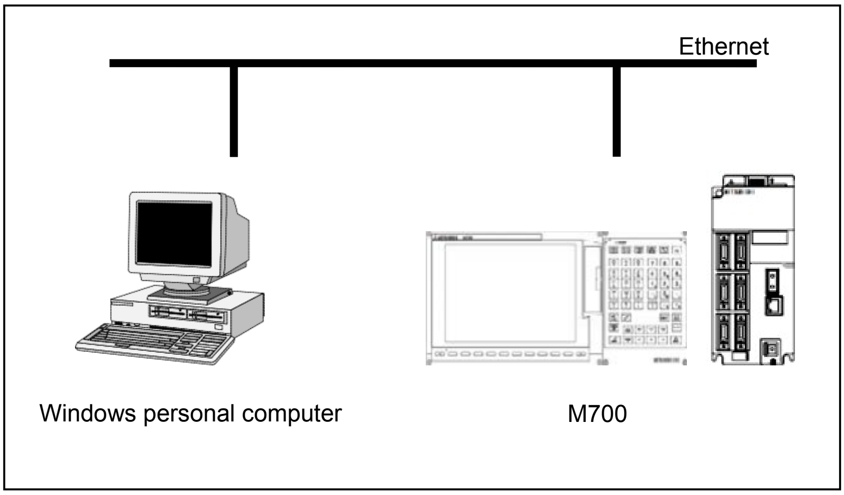


Figure 1-1 Connection with M700 series

### 1.3.2 Connection with M800 series

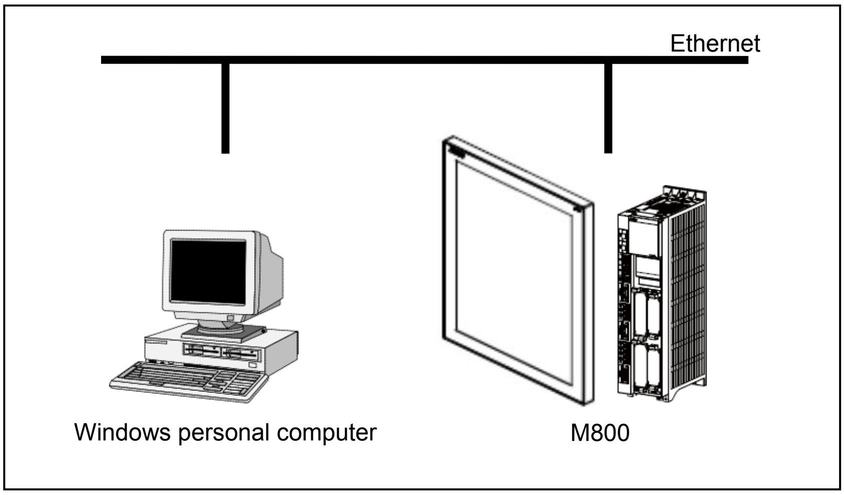


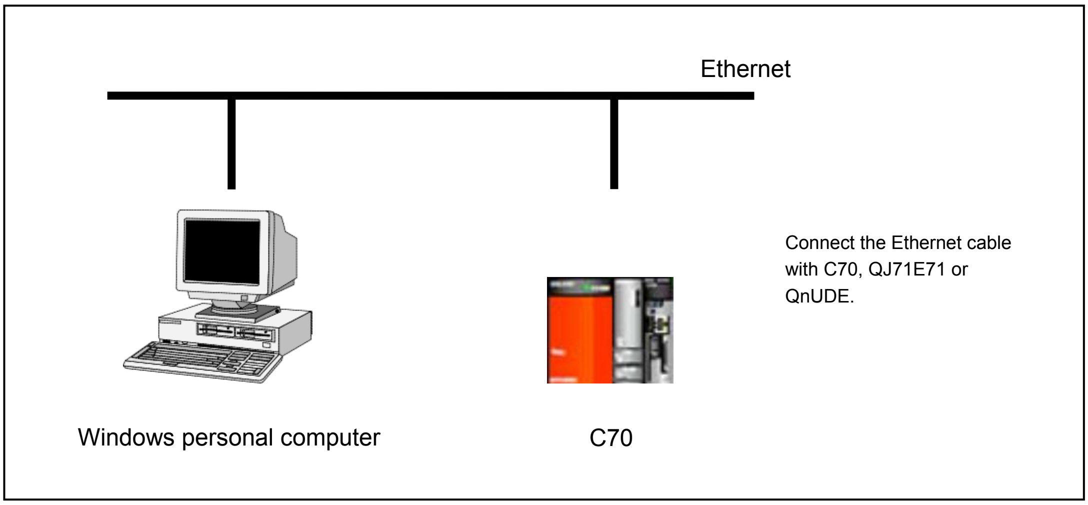
Figure 1-2 Connection with M800 series

### 1.3.3 Connection with C70

See the connection path between C70 and personal computer.

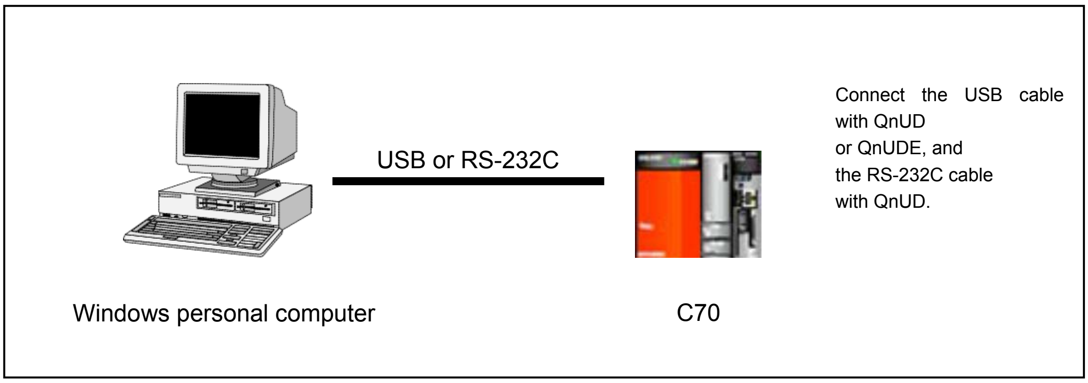
Table 1-1 C70 Connection path

| Model | Connection path | Connection module | Figure |
|---|---|---|---|
| C70 | Ethernet connection | C70 | Figure 1-3 |
| | Ethernet connection via QJ71E71 | QJ71E71 | Figure 1-3 |
| | Ethernet connection via QnUDE | QnUDE | Figure 1-3 |
| | USB connection via QnUD | QnUD | Figure 1-4 |
| | RS-232C connection via QnUD | QnUD | Figure 1-4 |
| | USB connection via QnUDE | QnUDE | Figure 1-4 |
| | GOT (bus connection) transparent | Base unit | Figure 1-5 |
| | GOT (Ethernet) transparent | C70<br>QJ71E71<br>QnUDE | Figure 1-6 |



Connect the Ethernet cable with C70, QJ71E71 or QnUDE.

Figure 1-3 Ethernet connection with C70



Connect the USB cable with QnUD or QnUDE, and the RS-232C cable with QnUD.

Figure 1-4 QnUD connection with C70

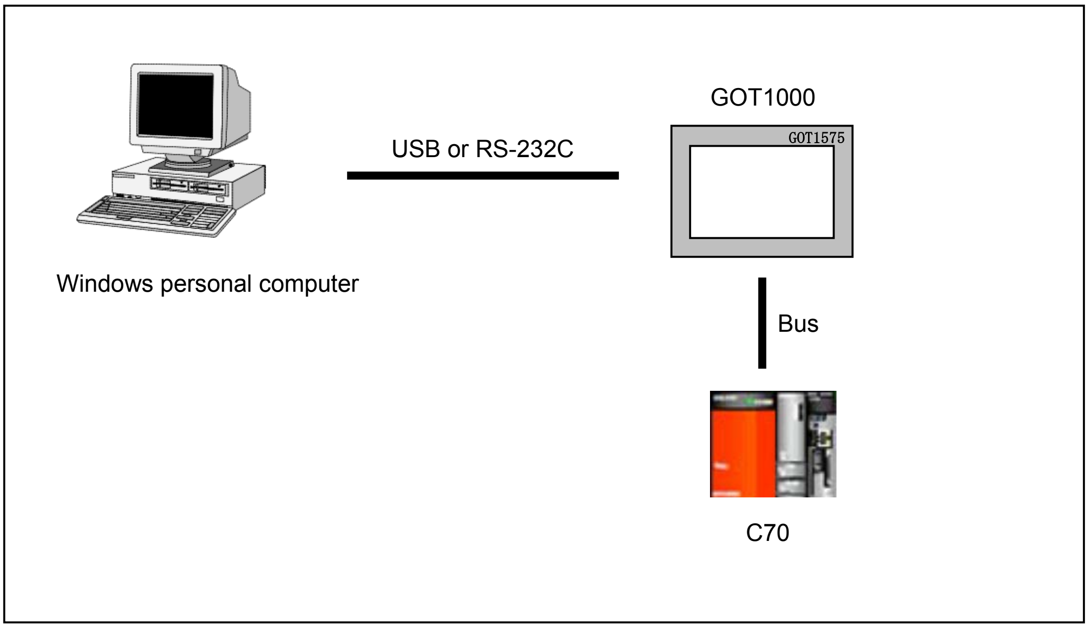


Figure 1-5 C70 and GOT (bus connection) transparent

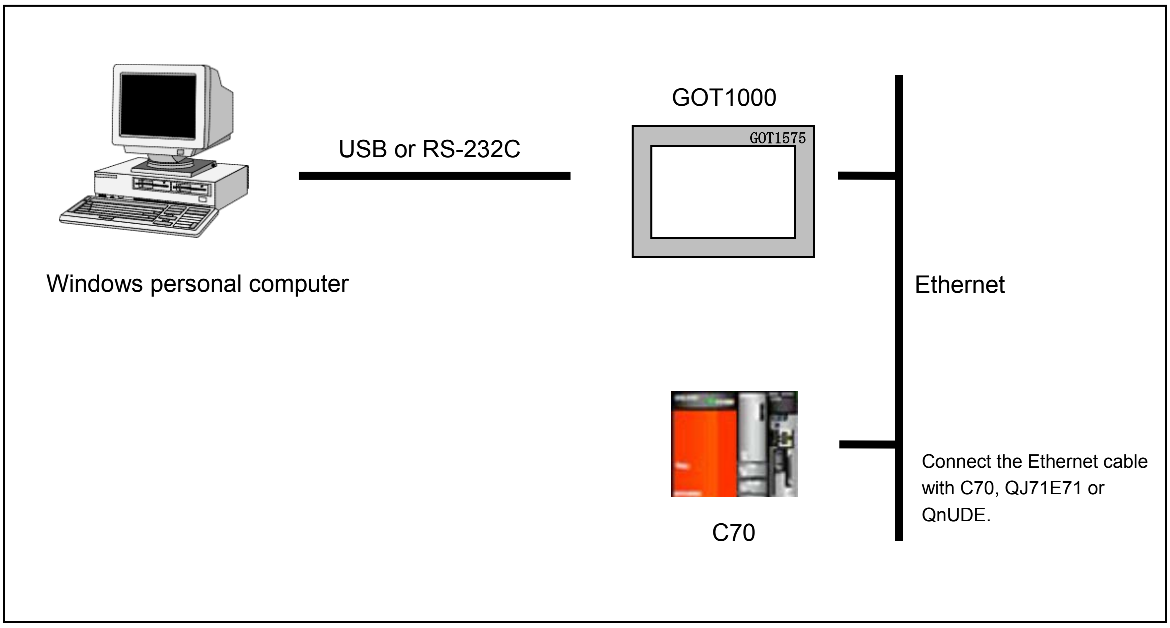


Figure 1-6 C70 and GOT (Ethernet) transparent

## 1.4 Development Environment

Environmental requirements for application development with this product are as below.

Table 1-2 Development and operating environment

| Item | Specifications |
|---|---|
| Personal computer | Personal Computer AT compatible machine (x86 processor)<br>Personal Computer AT compatible machine (x64 processor) |
| CPU (*2) | - |
| OS | Microsoft Windows 7 Home Basic SP1 or later, Japanese/English<br>Microsoft Windows 7 Home Premium SP1 or later, Japanese/English<br>Microsoft Windows 7 Professional SP1 or later, Japanese/English<br>Microsoft Windows 7 Ultimate SP1 or later, Japanese/English<br>Microsoft Windows 7 Enterprise SP1 or later, Japanese/English<br>Microsoft Windows 7 Home Premium x64 SP1 or later, Japanese/English<br>Microsoft Windows 7 Professional x64 SP1 or later, Japanese/English<br>Microsoft Windows 7 Ultimate x64 SP1 or later, Japanese/English<br>Microsoft Windows 7 Enterprise x64 SP1 or later, Japanese/English (*3) |
| Required memory (*2) | - |
| Disc space (*2) | - |
| Peripheral equipment (*2) | - |
| Development language (*2) | Microsoft Visual C++.NET 2003, 2005, 2008, 2010 (*1)<br>Microsoft Visual C++ Ver.5.0, Ver.6.0<br>Microsoft Visual Basic Ver.5.0, Ver.6.0<br>Microsoft Visual Basic for Applications Ver.5.0 (Excel 97 VBA equivalent), Ver.6.0 (Excel 2000 VBA equivalent) |
| Note | (*1) Development with native code (VC++) only. Development with managed code is not supported.<br>(*2) See the Windows operating environment recommended by Microsoft Corporation.<br>(*3) As the product is 32 bit module, it will run on the Windows 32-bit On Windows 64-bit (WOW64) subsystem if executed on x64 platform. It does not support 64 bit native operation. |

## 1.5 Installation

Dynamic link library (DLL) of the product is necessary to use its functions.
See the "release note" for how to install the product.
When installting the product on x64 platform, specify the destination folder as below.

C:\Program Files (x86)\EZSocket

## 1.6 Preparation for Use

To create an application using this product with VC++, VB or VBA, the following appropriate include files or modules are required. The table below shows the default folders of when the installation is made in the C drive from DVD-ROM.

Table 1-4 Include files according to development language

| | VC++ | VB or VBA |
|---|---|---|
| Installation destination folder | %ProgramFiles%\EZSocket\EZSocket Nc\include\Vc | %ProgramFiles%\EZSocket\EZSocket Nc\include\Vb |
| File | EZSocketNc.h<br>EZSocketNcStr.h<br>EZSocketNc_i.c<br>EZSocketNcDef.h<br>EZSocketNcErr.h<br>EZSocketCommonErr.h | EZNcDef.bas<br>EZNcErr.bas<br>EZComErr.bas |

To use the product with C70, the MELSEC programmable controller load module needs to be installed beforehand. See the release note for how to install it.

## 1.7 Interface

The product provides two types of interface as DLL inprocess server: custom interface and automation interface. The two types of interface have similar data access functions.
The custom interface works well with VC++ and the automation interface works well with VB and VBA.
The interface can be selected according to the development language.
As the interfaces of the product are based on Microsoft Component Object Model (**COM**), general knowledge about **COM** will be necessary to use the interfaces from the application. Note that general explanation about **COM** is not described in this manual.

### 1.7.1 Custom interface

See the custom interface list in Table 1-5.

Table 1-5 Custom interface list

| Component | Interface | Category |
|---|---|---|
| EZNcCommunication | | |
| | IEZNcCommunication3 | Communication |
| | IEZNcSystem | NC System |
| | IEZNcPosition | Position |
| | IEZNcCommand2 | Command |
| | IEZNcProgram2 | Program |
| | IEZNcTime | Time |
| | IEZNcAxisMonitor | Axis monitor |
| | IEZNcRunStatus | Operation status |
| | IEZNcFile6 | File |
| | IEZNcCommonVariable2 | Common variable |
| | IEZNcLocalVariable2 | Local variable |
| | IEZNcTool3 | Tool |
| | IEZNcATC3 | ATC |
| | IEZNcParameter3 | Parameter |
| | IEZNcOperation | Operation |
| | IEZNcDevice | Programmable controller device |
| EZNcSubFunction | | |
| | IEZNcSubFunction3 | Subfunction |

(Note 1) Though the interface name may change due to version upgrade, the former interface can still be used since the new interface inherits its former version.

Example) IEZNcFile5 → IEZNcFile6

Note that the former interface is for backward compatibility and the new interface is required to use the latest version of product.

(Note 2) C70 does not support automation interface.

1.7.2 Automation interface

See the automation interface list in Table 1-6.

They are handy for use with VB since all functions are contained in a single automation interface.

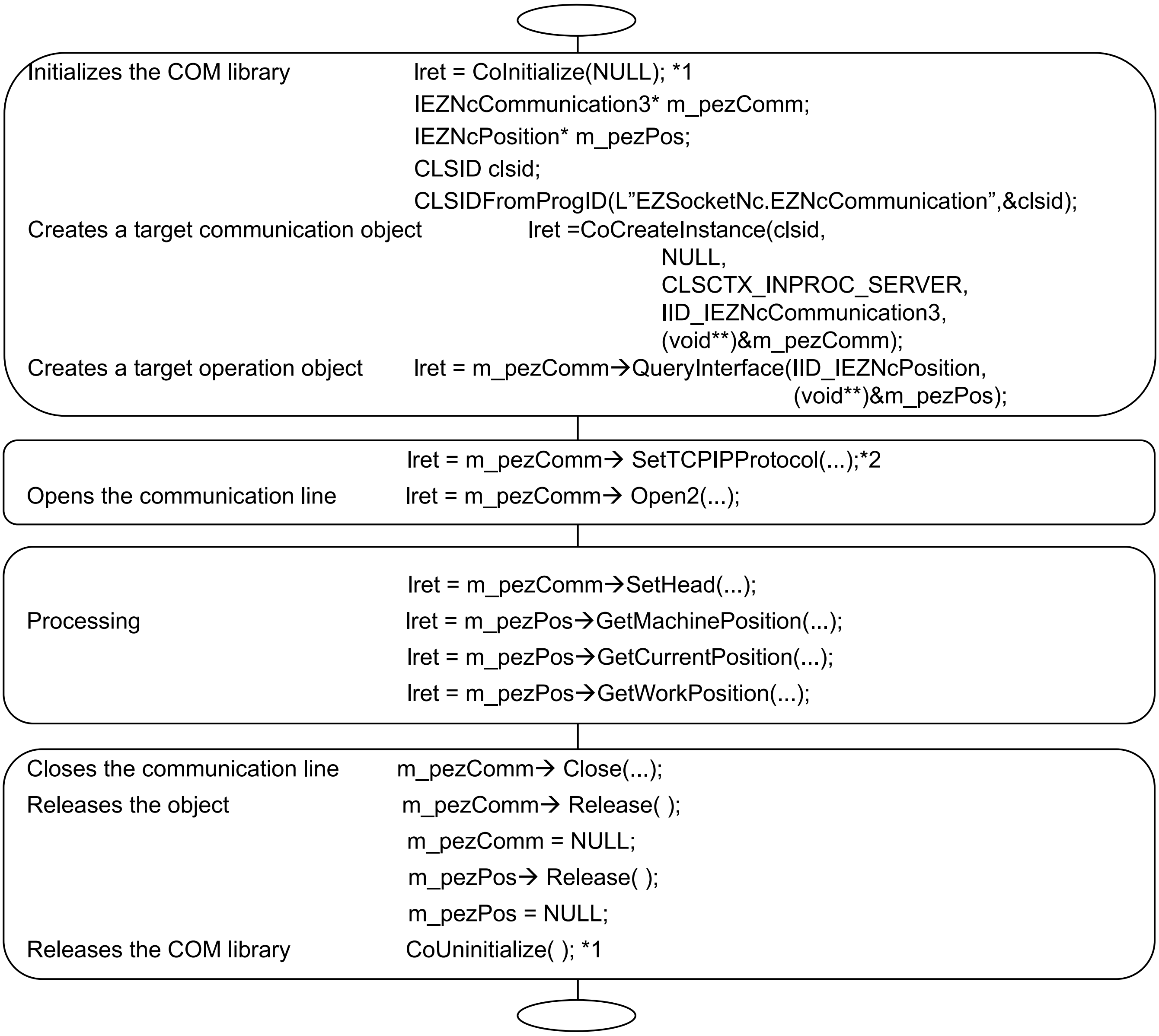
Table 1-6 Automation interface list

| Component | Interface | Category |
|---|---|---|
| DispEZNcCommunication | | |
| | IDispEZNcCommunication | Communication |
| | | NC System |
| | | Position |
| | | Command |
| | | Program |
| | | Time |
| | | Axis monitor |
| | | Operation status |
| | | File |
| | | Common variable |
| | | Local variable |
| | | Tool |
| | | ATC |
| | | Parameter |
| | | Programmable controller device |
| | | Operation |
| DispEZNcSubFunction | | |
| | IDispEZNcSubFunction | Subfunction |

## 1.8 Program Flow

### 1.8.1 VC++ program flow

This section explains the program flow outline to create an application for M700/M800 series or C70 using the custom interface with VC++.



```
Initializes the COM library            lret = CoInitialize(NULL); *1
                                       IEZNcCommunication3* m_pezComm;
                                       IEZNcPosition* m_pezPos;
                                       CLSID clsid;
                                       CLSIDFromProgID(L"EZSocketNc.EZNcCommunication",&clsid);
Creates a target communication object          lret =CoCreateInstance(clsid,
                                                              NULL,
                                                              CLSCTX_INPROC_SERVER,
                                                              IID_IEZNcCommunication3,
                                                              (void**)&m_pezComm);
Creates a target operation object      lret = m_pezComm→QueryInterface(IID_IEZNcPosition,
                                                                        (void**)&m_pezPos);
```

```
                                       lret = m_pezComm→ SetTCPIPProtocol(...);*2
Opens the communication line           lret = m_pezComm→ Open2(...);
```

```
                                       lret = m_pezComm→SetHead(...);
Processing                             lret = m_pezPos→GetMachinePosition(...);
                                       lret = m_pezPos→GetCurrentPosition(...);
                                       lret = m_pezPos→GetWorkPosition(...);
```

```
Closes the communication line          m_pezComm→ Close(...);
Releases the object                    m_pezComm→ Release( );
                                       m_pezComm = NULL;
                                       m_pezPos→ Release( );
                                       m_pezPos = NULL;
Releases the COM library               CoUninitialize( ); *1
```

*1 In the thread using this product, call the COM library function CoInitialize() before using this product and CoUninitialize( ) after using this product. When finishing the use of the objects of this product, call Release() to release the objects (decrement the reference count).

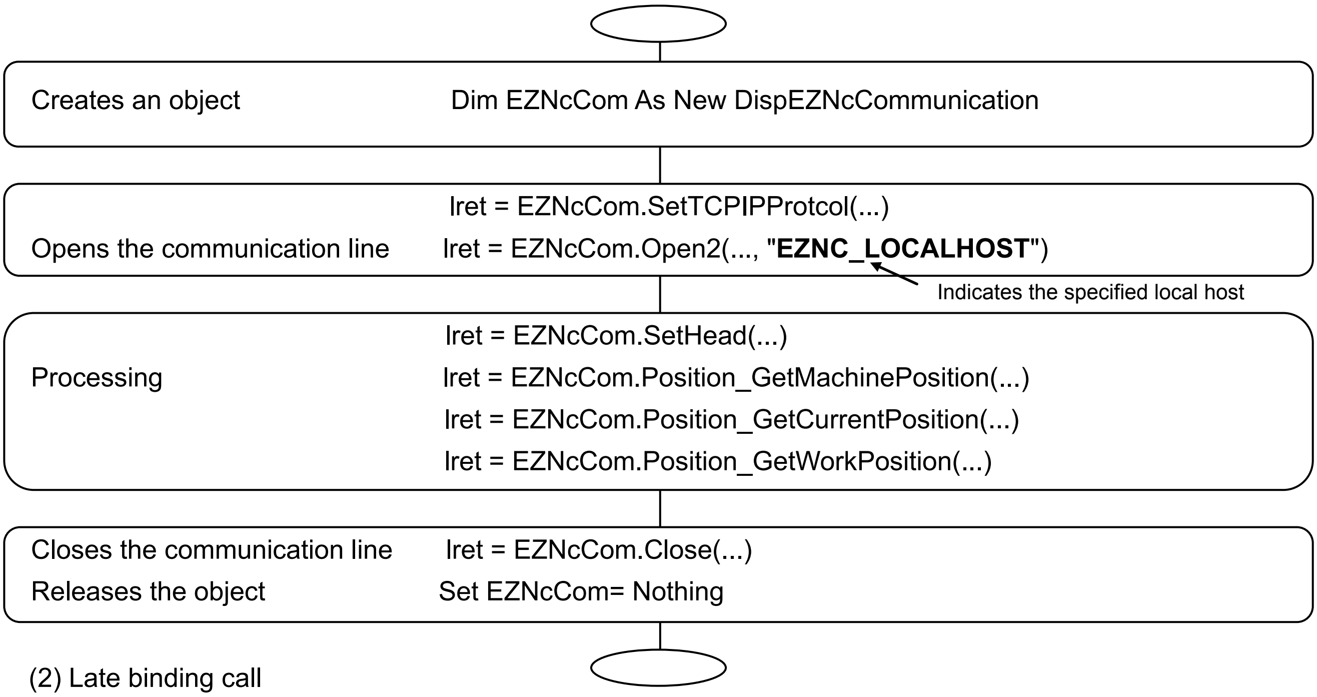
*2 When creating an M700/M800 series application, call SetTCPIPProtocol before Open. When creating a C70 application, call SetMelsecProtocol instead of SetTCPIPProtocol.

1.8.2 VB program flow

This section explains the program flow outline to create an application for M700/M800 series using the automation interface with VB.

(1) Early binding call

Early binding requires reference of type library of the automation interface to be set in advance.



Creates an object
```
Dim EZNcCom As New DispEZNcCommunication
```

Opens the communication line
```
Iret = EZNcCom.SetTCPIPProtcol(...)
Iret = EZNcCom.Open2(..., "EZNC_LOCALHOST")
```

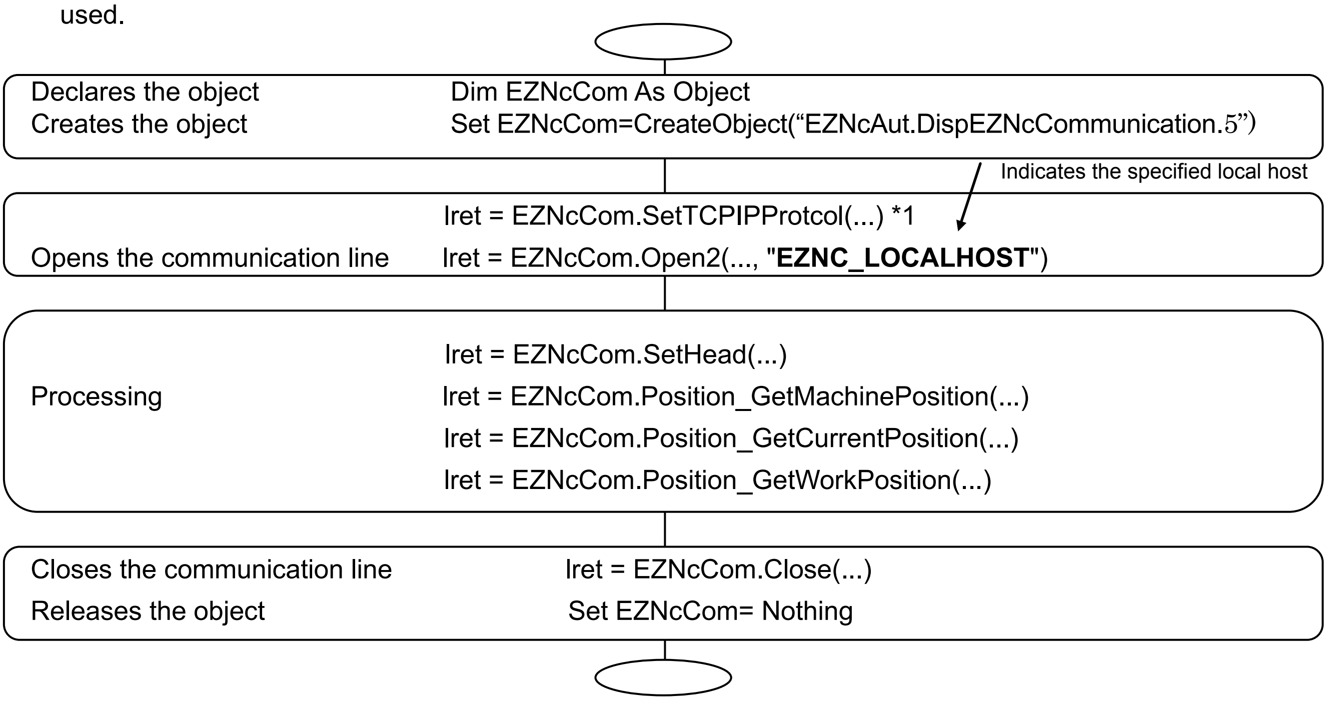
Processing
```
Iret = EZNcCom.SetHead(...)
Iret = EZNcCom.Position_GetMachinePosition(...)
Iret = EZNcCom.Position_GetCurrentPosition(...)
Iret = EZNcCom.Position_GetWorkPosition(...)
```

Closes the communication line
```
Iret = EZNcCom.Close(...)
```

Releases the object
```
Set EZNcCom= Nothing
```

(2) Late binding call

Late binding does not require reference setting. Note that the object browser function with VB cannot be used.



Declares the object
```
Dim EZNcCom As Object
```

Creates the object
```
Set EZNcCom=CreateObject(“EZNcAut.DispEZNcCommunication.5”)
```

Opens the communication line
```
Iret = EZNcCom.SetTCPIPProtcol(...) *1
Iret = EZNcCom.Open2(..., "EZNC_LOCALHOST")
```

Processing
```
Iret = EZNcCom.SetHead(...)
Iret = EZNcCom.Position_GetMachinePosition(...)
Iret = EZNcCom.Position_GetCurrentPosition(...)
Iret = EZNcCom.Position_GetWorkPosition(...)
```

Closes the communication line
```
Iret = EZNcCom.Close(...)
```

Releases the object
```
Set EZNcCom= Nothing
```

# 2. I/F DETAILED SPECIFICATIONS

## 2.1 Common Items

(1) Character code

All character string parameters in the interface of this product use UNICODE.

(2) Character string handling

With the method for handling character string data, if returning the character string data to the application, memory is allocated on the product side. Character string data memory that is no longer needed is freed up on the application side. If it is not freed up, a memory leak may occur.

To develop applications with custom interfaces for use with VC++, use **CoTaskMemFree()** to explicitly free up character string area memory.

To develop applications with automation interfaces for use with VC++, use **SysFreeString()** to explicitly free up character string area memory.

(3) Handling of common error codes

The method return value will be either for successful completion (**S_OK**) or failure (**S_FALSE**) of the method.

A detailed method error is returned as an argument.

Commonly used error messages include the following.

**EZ_ERR_NOT_OPEN**: Communication lines are not open.
**EZ_ERR_DOUBLE_OPEN**: Double open error
**EZ_ERR_DATA_TYPE**: Invalid argument data type
**EZ_ERR_DATA_RANGE**: Invalid argument data range
**EZ_ERR_NOT_SUPPORT**: Not supported
**EZ_ERR_NULLPTR**: Argument is **NULL** pointer

(4) Error handling when calling an I/F incompatible with the model

If the I/F of a function not supported by the model is called, **EZ_ERR_NOT_SUPPORT** is output as an unsupported error code.

(5) Notations used in Chapter 2.3 and later in this document

The meaning of notations used on pages describing individual method detailed specifications in Chapter 2.3 and later are as follows.

**"□ Argument" section:** Describes the argument specifications of the method.
**"□ Return value" section:** Describes the return values of the method.
**"□ Function" section:** Describes the general function of the method.
**"□ Reference" section:** Describes the methods related to the method.
**"□ Specification" section:** Indicates that specification of the system number (including PLC axis systems) and axis numbers is necessary when executing the method. If used without specifying, note that operation will not be guaranteed.

Required specification details are abbreviated with the following marks.

System: Set the part system number. Use SetHead() for the part system number setting.
PLC axis: Set the PLC axis system number. Use SetHead() for PLC axis system number setting.
Axis number: Set the axis number.

In addition, even if the System mark is listed, sometimes system specification is not required according to the argument value, as with the method in Chapter 2.12.1. For more information, see the supplementary information described in the "□ Specification" section on an individual method's page.

(6) Restrictions on methods that require part system setting

For disabled systems, if a method that requires part system setting is executed, the method's execution result will be null. Check beforehand whether the set system is an enabled system.

In addition, system 1 will be set after a line opens.

(7) File name specification

In this product, the Mitsubishi CNC (*1) is regarded as a single drive, and the various data on the NC control unit (machining program, tool offset, etc.) are treated as a file. To access an NC control unit file, set the file name according to the following form unless otherwise noted.

Drive name + ":" + \directory name + \file name

Make sure to set the file name with an absolute path.*2

Also, the drive name should correspond to an NC control unit number as follows.

| NC control unit number | Drive name (NC memory) |
|---|---|
| 01 | M01 |
| 02 | M02 |
| 03 | M03 |
| : | : |
| : | : |
| FF | MFF |

*1: Described in the following sections of this manual as "NC control unit".

*2: Set the file name with upper-case characters.
In English-version operating systems, set by changing "¥" to a backslash.

(8) Restrictions on calls from multiple threads of the C70

With the C70, if the number of threads the product uses exceeds 1000, error codes may no longer be output correctly.

Keep in mind the number of threads the product uses when creating a C70 application.

## 2.2 Method List

Table Interface list ○: Supported, , -: Not supported

| Chapter number | Function class | Interface (operation) | Function | Target | | |
|---|---|---|---|---|---|---|
| | | | | M700 series | M800 series | C70*1 |
| 2.3.1 | IEZNcCommunication3 | Open2 | Open NC control unit system line | ○ | ○ | ○ |
| 2.3.2 | (Communication) | Close | Close NC control unit system line | ○ | ○ | ○ |
| 2.3.3 | | SetHead | Set system number | ○ | ○ | ○ |
| 2.3.4 | | GetHead | Get system number | ○ | ○ | ○ |
| 2.3.5 | | SetTCPIPProtocol | Set TCPIP communication settings | ○ | ○ | - |
| 2.3.6 | | SetMelsecProtocol | Set MELSEC communication settings | - | - | ○ |
| 2.4.1 | IEZNcSystem | GetVersion | Get system number, name, and control S/W version | ○ | ○ | ○ |
| 2.4.2 | (NC System) | GetSystemInformation | Get NC system information | ○ | ○ | ○ |
| 2.4.3 | | GetAlarm | Get alarm information | ○ | - | ○ |
| 2.4.4 | | GetAlarm2 | Get alarm information | ○ | ○ | ○ |
| 2.5.1 | IEZNcPosition | GetWorkPosition | Get workpiece coordinate position (skip ON-compliant) | ○ | ○ | ○ |
| 2.5.2 | (Position) | GetWorkPosition2 | Get workpiece coordinate position (skip ON-compliant) | ○ | ○ | ○ |
| 2.5.3 | | GetMachinePosition | Get machine position (skip ON-compliant) | ○ | ○ | ○ |
| 2.5.4 | | GetMachinePosition2 | Get machine position (skip ON-compliant) | ○ | ○ | ○ |
| 2.5.5 | | GetCurrentPosition | Get current position | ○ | ○ | ○ |
| 2.5.6 | | GetDistance | Get command remaining distance (skip ON-compliant) | ○ | ○ | ○ |
| 2.5.7 | | GetDistance2 | Get command remaining distance (skip ON-compliant) | ○ | ○ | ○ |
| 2.5.8 | | GetNextDistance | Get next travel distance | ○ | ○ | ○ |
| 2.5.9 | | GetFeedSpeed | Get feed speed | ○ | ○ | ○ |
| 2.5.10 | | GetTCPSpeed | Get tip speed | ○ | ○ | - |
| 2.5.11 | | GetManualOverlap | Get manual interrupt amount | ○ | ○ | ○ |
| 2.5.12 | | GetManualOverlap2 | Get manual interrupt amount | ○ | ○ | ○ |
| 2.5.13 | | GetProgramPosition | Get program position | ○ | ○ | ○ |
| 2.5.14 | | GetProgramPosition3 | Get program position | ○ | ○ | ○ |
| 2.5.15 | | GetTCPMachinePosition | Get tip machine position | ○ | ○ | - |
| 2.5.16 | | GetTCPWorkPosition | Get tip workpiece position | ○ | ○ | - |
| 2.5.17 | | GetFeedbackPosition | Get feedback position | ○ | ○ | - |
| 2.5.18 | | GetTableCoordinationPosition | Get table coordinate system counter | ○ | ○ | - |
| 2.5.19 | | GetWorkInstallationPosition | Get workpiece installation coordinate system counter | ○ | ○ | - |
| 2.5.20 | | GetInclinedSurfacePosition | Get inclined surface coordinate system counter | ○ | ○ | - |
| 2.6.1 | IEZNcCommand2 | GetGCodeCommand | Get G code modal command value | ○ | ○ | ○ |
| 2.6.2 | (Command) | GetToolCommand | Get tool compensation number | ○ | ○ | ○ |
| 2.6.3 | | GetFeedCommand | Get feed speed command value | ○ | ○ | ○ |
| 2.6.4 | | GetCommand2 | Get M/S/T/B function command modal value | ○ | ○ | ○ |
| 2.6.5 | | SetCommand2 | Set manual numerical value command settings (M, S, T, B) | ○ | ○ | ○ |
| 2.7.1 | IEZNcProgram2 | CurrentBlockRead | Read current block | ○ | ○ | ○ |
| 2.7.2 | (Program end) | GetProgramNumber2 | Get program number (main, sub) | ○ | ○ | ○ |
| 2.7.3 | | GetSequenceNumber | Get sequence number (main, sub) | ○ | ○ | ○ |
| 2.7.4 | | GetBlockNumber | Get block number (main, sub) | ○ | ○ | ○ |
| 2.7.5 | | GetSubProLevel | Get subprogram call level | ○ | ○ | ○ |
| 2.7.6 | | GetInformation | Get user machining program information | ○ | ○ | ○ |
| 2.7.7 | | GetCurrentBlockByByte | Get number of bytes from start of program | ○ | ○ | - |
| 2.8.1 | IEZNcTime | GetClockData | Get date and time | ○ | ○ | ○ |
| 2.8.2 | (Time) | SetClockData | Set date and time settings | ○ | ○ | ○ |
| 2.8.3 | | GetAliveTime | Get power-on time | ○ | ○ | ○ |
| 2.8.4 | | SetAliveTime | Set power-on time settings | ○ | ○ | ○ |
| 2.8.5 | | GetRunTime | Get automatic operation time | ○ | ○ | ○ |
| 2.8.6 | | SetRunTime | Set automatic operation time settings | ○ | ○ | ○ |
| 2.8.7 | | GetStartTime | Get automatic start time | ○ | ○ | ○ |
| 2.8.8 | | SetStartTime | Set automatic start time settings | ○ | ○ | ○ |
| 2.8.9 | | GetEstimateTime | Get external integration time (1, 2) | ○ | ○ | ○ |
| 2.8.10 | | SetEstimateTime | Set external integration time settings (1, 2) | ○ | ○ | ○ |

| hapter number | Function class | Interface (operation) | Function | Target | | |
|---|---|---|---|---|---|---|
| | | | | M700 series | M800 series | C70[*1] |
| 2.9.1 | IEZNcAxisMonitor<br>(Axis monitor) | GetServoMonitor | Get servo monitor | ○ | ○ | ○ |
| 2.9.2 | | GetServoVersion | Get servo version information | ○ | ○ | ○ |
| 2.9.3 | | GetServoDiagnosis | Get servo diagnostics information | ○ | ○ | ○ |
| 2.9.4 | | GetPowerVersion | Get power supply version information | ○ | ○ | ○ |
| 2.9.5 | | GetPowerDiagnosis | Get power supply diagnostics information | ○ | ○ | ○ |
| 2.9.6 | | GetSpindleMonitor | Monitor spindle | ○ | ○ | ○ |
| 2.9.7 | | GetSpindleVersion | Get spindle version information | ○ | ○ | ○ |
| 2.9.8 | | GetSpindleDiagnosis | Get spindle diagnostics information | ○ | ○ | ○ |
| 2.9.9 | | GetAbsPositionMonitor | Get absolute position monitor information | ○ | ○ | ○ |
| 2.9.10 | | GetAuxAxisMonitor | Get auxiliary axis monitor information | ○ | ○ | - |
| 2.9.11 | | GetAuxAxisDiagnosis | Get auxiliary axis diagnostics information | ○ | ○ | - |
| 2.9.12 | | GetAuxAxisVersion | Get auxiliary axis version information | ○ | ○ | - |
| 2.9.13 | | GetDowelTime | Get remaining dwell time | ○ | ○ | ○ |
| 2.9.14 | | GetPowerConsumption | Get current power consumption | - | ○ | - |
| 2.9.15 | | GetIntegralPower | Get integral power | - | ○ | - |
| 2.10.1 | IEZNcRunStatus<br>(Operation status) | GetInvalidStatus | Get disabled status | ○ | ○ | ○ |
| 2.10.2 | | GetCommandStatus | Get operation command status | ○ | ○ | ○ |
| 2.10.3 | | GetCuttingMode | Get cutting mode status | ○ | ○ | ○ |
| 2.10.4 | | GetAxisStatus | Get axis status | ○ | ○ | ○ |
| 2.10.5 | | GetRunStatus | Get operation status | ○ | ○ | ○ |
| 2.11.1 | IEZNcFile6<br>(File) | FindDir2 | Search directory | ○ | ○ | ○ |
| 2.11.2 | | FindNextDir2 | Search next directory | ○ | ○ | ○ |
| 2.11.3 | | ResetDir | Terminate directory search | ○ | ○ | ○ |
| 2.11.4 | | Copy2 | Copy file | ○ | ○ | ○ |
| 2.11.5 | | Delete2 | Delete file | ○ | ○ | ○ |
| 2.11.6 | | Rename2 | Change file name | ○ | ○ | ○ |
| 2.11.7 | | GetDriveInformation | Get drive information | ○ | ○ | ○ |
| 2.11.8 | | GetDriveSize | Get free drive space | ○ | ○ | ○ |
| 2.11.9 | | GetDriveSize2 | Get free drive space | - | ○ | - |
| 2.11.10 | | OpenFile3 | Open file | ○ | ○ | ○ |
| 2.11.11 | | CloseFile2 | Close file | ○ | ○ | ○ |
| 2.11.12 | | AbortFile2 | Force close file | ○ | ○ | ○ |
| 2.11.13 | | ReadFile2 | Read file | ○ | ○ | ○ |
| 2.11.14 | | WriteFile | Write file | ○ | ○ | ○ |
| 2.11.15 | | OpenNCFile2 | Open NC program file | ○ | ○ | - |
| 2.11.16 | | CloseNCFile2 | Close NC program file | ○ | ○ | - |
| 2.11.17 | | AbortNCFile2 | Force close NC program file | ○ | ○ | - |
| 2.11.18 | | ReadNCFile2 | Write NC program file | ○ | ○ | - |
| 2.11.19 | | WriteNCFile | Read NC program file | ○ | ○ | - |
| 2.12.1 | IEZNcCommonVariable2<br>(Common variable) | CommonVRead | Read common variables (#100, #500) | ○ | ○ | ○ |
| 2.12.2 | | CommonVWrite | Write common variables (#100, #500) | ○ | ○ | ○ |
| 2.12.3 | | GetSize | Get number of sets for common variables (#100, #500) | ○ | ○ | ○ |
| 2.12.4 | | GetName | Get names of common variables (#500 to 519) | ○ | ○ | ○ |
| 2.12.5 | | SetName | Set name settings for common variables (#500 to 519) | ○ | ○ | ○ |
| 2.12.6 | | GetCVNullData | Get value when no numerical value is set | ○ | ○ | ○ |
| 2.13.1 | IEZNcLocalVariable2<br>(Local variable) | LocalVRead | Read local variable | ○ | ○ | ○ |
| 2.13.2 | | GetMacroLevel | Get macro subprogram execution level (level 0 to 4) | ○ | ○ | ○ |
| 2.13.3 | | GetLVNullData | Get value when no numerical value is set | ○ | ○ | ○ |

| Chapter number | Function class | Interface (operation) | Function | Target | | |
|---|---|---|---|---|---|---|
| | | | | M700 series | M800 series | C70*1 |
| 2.14.1 | IEZNcTool3 (Tool) | GetToolSetSize | Get number of sets for tool offset | ○ | ○ | ○ |
| 2.14.2 | | GetType | Get tool offset type | ○ | ○ | ○ |
| 2.14.3 | | GetOffset | Get tool offset data | ○ | ○ | ○ |
| 2.14.4 | | GetOffset2 | Get tool offset data | ○ | ○ | ○ |
| 2.14.5 | | SetOffset | Set tool offset data settings | ○ | ○ | ○ |
| 2.14.6 | | GetToolWorkOffset | Get workpiece coordinate system offset (#54 to 60) | ○ | ○ | ○ |
| 2.14.7 | | GetToolWorkOffset2 | Get workpiece coordinate system offset (#54 to 60) | ○ | ○ | ○ |
| 2.14.8 | | SetToolWorkOffset | Set workpiece coordinate system offset settings (#54 to 60) | ○ | ○ | ○ |
| 2.14.9 | | SetToolWorkOffset2 | Set workpiece coordinate system offset settings | ○ | ○ | ○ |
| 2.14.10 | | GetSurface | Get reference surface height | ○ | ○ | ○ |
| 2.14.11 | | GetSurface2 | Get reference surface height | ○ | ○ | ○ |
| 2.14.12 | | SetSurface | Set reference surface height settings | ○ | ○ | ○ |
| 2.14.13 | | GetToolLifeType2 | Get tool life control type 2 | ○ | ○ | ○ |
| 2.14.14 | | SetToolLifeType2 | Set tool life control type settings 2 | ○ | ○ | ○ |
| 2.14.15 | | GetToolLifeGroupList | Get tool life control group number list | ○ | ○ | ○ |
| 2.14.16 | | ChangeToolLifeGroup | Change tool life control group number | ○ | ○ | ○ |
| 2.14.17 | | DeleteToolLifeGroup | Delete tool life control group number | ○ | ○ | ○ |
| 2.14.18 | | GetToolLifeToolNoList | Get list of tools within tool life control group | ○ | ○ | ○ |
| 2.14.19 | | AddToolLifeToolNo | Add tool number to tool life control group | ○ | ○ | ○ |
| 2.14.20 | | ChangeToolLifeToolNo | Change tool life control tool number | ○ | ○ | ○ |
| 2.14.21 | | DeleteToolLifeToolNo | Delete tool life control tool number | ○ | ○ | ○ |
| 2.14.22 | | GetToolLifeValue | Get tool life control data | ○ | ○ | ○ |
| 2.14.23 | | SetToolLifeValue | Set individual tool life control data settings | ○ | ○ | ○ |
| 2.14.24 | | SetToolLifeValue2 | Set tool life control data settings | ○ | ○ | ○ |
| 2.15.1 | IEZNcATC3 (ATC) | GetMGNControl | Get ATC tool registration control parameter | ○ | ○ | ○ |
| 2.15.2 | | GetMGNSize | Get total number of sets of magazine pots for ATC tool registration | ○ | ○ | ○ |
| 2.15.3 | | GetMGNSize2 | Get number of sets of pots for each magazine for ATC tool registration | ○ | ○ | ○ |
| 2.15.4 | | GetMGNReady2 | Get tool number for ATC tool registration | ○ | ○ | ○ |
| 2.15.5 | | GetMGNPot | Get tool number for magazine pot for ATC tool registration | ○ | ○ | ○ |
| 2.15.6 | | GetMGNPot3 | Get tool number for each magazine pot for ATC tool registration | ○ | ○ | ○ |
| 2.15.7 | | SetMGNPot | Set tool number settings for magazine pots for ATC tool registration | ○ | ○ | ○ |
| 2.15.8 | | SetMGNPot3 | Set tool number settings for each magazine pot for ATC tool registration | ○ | ○ | ○ |
| 2.15.9 | | GetMGNAux | Get user programmable controller interface for ATC tool registration | ○ | ○ | ○ |
| 2.15.10 | | SetMGNAux | Set user programmable controller interface settings for ATC tool registration | ○ | ○ | ○ |
| 2.16.1 | IEZNcParameter3 (Parameter) | GetParameterData2 | Get parameters | ○ | - | ○ |
| 2.16.2 | | GetParameterData3 | Get parameters | ○ | ○ | ○ |
| 2.16.3 | | SetParameterData2 | Set parameter settings | ○ | - | ○ |
| 2.16.4 | | SetParameterData3 | Set parameter settings | ○ | ○ | ○ |
| 2.17.1 | IEZNcOperation (Operation) | Search | Operation search | ○ | ○ | ○ |
| 2.17.2 | | Run | Start programmable controller program | ○ | ○ | ○ |
| 2.17.3 | | Stop | Stop programmable controller program | ○ | ○ | ○ |
| 2.18.1 | IEZNcDevice (Device) | SetDevice | Set device settings | ○ | ○ | - |
| 2.18.2 | | DeleteDeviceAll | Delete all device settings | ○ | ○ | - |
| 2.18.3 | | ReadDevice | Read device | ○ | ○ | - |
| 2.18.4 | | WriteDevice | Write device | ○ | ○ | - |
| 2.18.5 | | ReadBlockDevice | Batch read devices | ○ | ○ | - |
| 2.18.6 | | WriteBlockDevice | Batch write devices | ○ | ○ | - |
| 2.19.1 | IEZNcSubFunction3 (Subfunction) | ChangeInit2 | Initialize subfunction | ○ | ○ | ○ |
| 2.19.2 | | GetToolWorkOffsetOfFile | Get workpiece coordinate system offset data from workpiece offset file | ○ | ○ | ○ |
| 2.19.3 | | SetToolWorkOffsetOfFile | Set workpiece coordinate system offset data settings for workpiece offset file | ○ | ○ | ○ |

*1) C70 does not support automation interface.

**C70** **M700** **M800**

## 2.3.1 IEZNcCommunication3::Open2 Open line

□ **Custom call procedure**

```
HRESULT      Open2 (
                     LONG lSystemType,              // (I) NC system type
                     LONG lMachine,                 // (I) NC control unit
                     LONG lTimeOut,                 // (I) Communication time-out value
                     LONG* plRet                    // (O) Error code
                     )
```

□ **Automation call procedure**

```
             Open2 (
                     lSystemType As LONG            // (I) NC system type
                     lMachine As LONG               // (I) NC control unit
                     lTimeOut As LONG               // (I) Communication time-out value
                     lpcwszHostName As STRING       // (I) Host name
                     ) As LONG                      // (O) Error code
```

□ **Argument**

*lSystemType*: Sets the NC system type.

| Value | Meaning |
|---|---|
| **EZNC_SYS_MELDASC70** | With the C70, performs a line connection. |
| **EZNC_SYS_MELDAS700M** | With M700 M systems, performs a line connection. |
| **EZNC_SYS_MELDAS700L** | With M700 L system, performs a line connection. |
| **EZNC_SYS_MELDAS800M** | With M800 M systems, performs a line connection. |
| **EZNC_SYS_MELDAS800L** | With M800 L system, performs a line connection. |

*lMachine*: Sets the NC control unit number. When connecting to multiple Mitsubishi CNCs, specify the different NC control unit numbers for each Mitsubishi CNC. Value: 1 to 255

*lTimeOut*: Sets the communication time-out value. However, with C70, this value is disabled. The C70 communication time-out is set by *lTimeOut* in **SetMelsecProtocol()**.

| Value | Meaning |
|---|---|
| **1 to 3000** | Time-out value (unit: 100 ms)<br>(M700/M800 series is 10 or more, and if a time-out error occurs, increase the value.) |

*lpcwszHostName*: Sets the NC system host name to connect. The IP address can also be specified.
When connecting to a local host, set **"EZNC_LOCALHOST"** as the character string.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZ_ERR_DATA_TYPE**: Invalid argument data type
**EZ_ERR_DATA_RANGE**: Invalid argument data range
**EZNC_SYSFUNC_IOCTL_ADDR**: Invalid NC control unit number
**EZNC_SYSFUNC_IOCTL_NOTOPEN**: Device is not open
**EZNC_SYSFUNC_IOCTL_DATA**: Invalid communication parameter data range
**EZNC_COMM_NOTSETUP_PROTOCOL**: TCP/IP communication has not been set (M700/M800 series only)
**EZNC_COMM_NOTMODULE**: No submodule
**EZNC_COMM_CREATEPC**: EZSocketPc objects cannot be created (C70 only)
**EZNC_COMM_CANNOT_OPEN**：When connecting to local host with automation call, host name EZNC_LOCALHOST has not been set.

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Executes connection of the communication line for an **IEZNcCommunication3** object.

For the M700/M800 series, before open, make sure to execute **SetTCPIPProtocol()** to configure the communication settings. Also, for the C70, before open, make sure to execute **SetMelsecProtocol()** to configure the communication settings. If it is not executed, an error will occur when **Open2()** is executed.

(Note 1) The existing interface IEZNcCommunication2::Open (() can be used as a backward compatible model. Use this method when newly starting use of EZSocket.

□ **Reference**

**Close(), SetTCPIPProtocol(), SetMelsecProtocol()**

□ **Specification**

## 2.3.2 IEZNcCommunication3::Close Close line

□ **Custom call procedure**

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

□ **Automation call procedure**

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

□ **Argument**

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_SYSFUNC_IOCTL_NOTOPEN**: Device is not open

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Failure |

□ **Function**

The communication lines of **IEZNcCommunication3** objects connected by **Open2( )** will be disconnected.
For the M700/M800 series, even if the line is closed, the content set by **SetTCPIPProtocol**() is maintained, so **Open2**() can be performed again.
For the C70, even if the line is closed, the content set by **SetMelsecProtocol**() is maintained, so **Open2**() can be performed again.

□ **Reference**

**Open2(), SetTCPIPProtocol(), SetMelsecProtocol()**

□ **Specification**

## 2.3.3 IEZNcCommunication3::SetHead Set part system number

□ **Custom call procedure**

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

□ **Automation call procedure**

```
            SetHead(
                  lHead As LONG                  // (I) Part system number.
                  ) As LONG                      // (O) Error code
```

|  |  |  |
|---|---|---|
| □ **Argument** | *lHead*: Sets the part system number. or the PLC axis system. Value: The range is determined by Mitsubishi CNC specifications (optional) and the machine manufacturer setting values (parameter). A value of 0 means "Not setting". When setting the PLC axis system, sets **EZNC_PLCAXIS**.<br>*plRet*: Returns an error code. (Upon automation, the return value is used.)<br>**S_OK**: Normal termination | |
| □ **Return value** | Value | Meaning |
| | **S_OK**<br>**S_FALSE** | Normal termination<br>Failure |
| □ **Function** | Set the NC axis / PLC axis system number.<br>Set the part system number before executing a method that requires system number. setting.<br>The set part system remains enabled until it is changed.<br>System 1 is set after a line opens. | |
| □ **Reference** | **GetHead()** | |
| □ **Specifica-tion** | | |

## 2.3.4 IEZNcCommunication3::GetHead Get part system number

□ **Custom call procedure**

```
HRESULT      GetHead(
                   LONG* plHead,                    // (O) Part system number.
                   LONG* plRet                      // (O) Error code
                   )
```

□ **Automation call procedure**

```
             GetHead(
                   plHead As LONG*                  // (O) Part system number.
                   ) As LONG                        // (O) Error code
```

| | |
|---|---|
| □ **Argument** | *plHead*: Returns the part system number. or the PLC axis system. Value: The range is determined by Mitsubishi CNC specifications (optional) and the machine manufacturer setting values (parameter). A value of 0 means "Not setting".<br>The PLC axis system returns **EZNC_PLCAXIS**.<br><br>*plRet*: Returns an error code. (Upon automation, the return value is used.)<br>**S_OK**: Normal termination |

| □ **Return value** | Value | Meaning |
|---|---|---|
| | **S_OK** | Normal termination |
| | **S_FALSE** | Failure |

| | |
|---|---|
| □ **Function** | Gets the NC axis / PLC axis system number. The PLC axis system number gets **EZNC_PLCAXIS**. |
| □ **Reference** | **SetHead()** |
| □ **Specifica-tion** | |

## 2.3.5 IEZNcCommunication3::SetTCP/IPProtocol Set TCP/IP communication protocol

□ **Custom call procedure**

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

□ **Automation call procedure**

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

□ **Argument**

*lpcwszIPAddress*: Sets the IP address of the M700/M800 series connection destination. (Example: "192.168.1.1")

*lPort*: Sets the M700/M800 series connection destination port number.
For the port number, check the settings on the Mitsubishi CNC side. For the M700/M800 series, the port number becomes 683.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_COMM_ALREADYOPENED**: Cannot be set because communication is already in progress
**EZ_ERR_DATA_RANGE**: Invalid IP address or port number
**EZ_ERR_NOT_SUPPORT**: Not supported

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Sets TCP/IP communication protocol.
For the M700/M800 series, call before performing **Open2()**. If it is not called, an error will occur when **Open2()** is executed.
The setting details are retained until the object is released by **Release()**.
Temporarily, until **Close()** is performed if **Open2()** is performed, re-setting with **SetTCPIPProtocol()** cannot be done. An error will occur.
This function is not supported with C70. (EZ_ERR_NOT_SUPPORT is returned to plRet.)

□ **Reference**

**Open2(), Close()**

□ **Specifica-tion**

## 2.3.6 IEZNcCommunication3::SetMelsecProtocol Set Melsec communication settings

□ **Custom call procedure**

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

□ **Argument**

*pstOpen*: A pointer referring to the **EZNCST_OPEN** structure which sets the parameters for the open. Refer to the following for members of the structure of **EZNCST_OPEN**.

(Note) The structure is created with the assumption of connection types other than Ethernet communication and serial communication. According to the following structure member descriptions, set 0 for unneeded parameters.

*lNetworkNumber*: Sets the network number with MELSECNET/10(H). Sets "0x00" when a host station is set. When a Qn multi-drop connection (serial communication, via CC-Link module) is set, the following applies.

| Value | Meaning |
|---|---|
| **0x00** | Sets a host network |
| **0x01** | Sets another network for multi-drop destination |

*lStationNumber*: Sets the station number with MELSECNET and CC-Link. "0xFF" is set when a host station is set. If accessing the CPU of the CPU board and the AF board, sets a host station. When a Qn multi-drop connection (serial communication, via CC-Link module) is set, the following applies.

| Value | Meaning |
|---|---|
| **0x00** | Set a host network |
| **0x01** | Set another network for multi-drop destination |

*lUnitNumber*: Sets sthe module number of the computer link (serial communication) module or the station number for a Qn communication system intelligent special module. However, sets "0x00" with the setting of a QnA series host station (module mounted to the host station CPU). This is disabled with computer link (serial communication) communication and a Qn intelligent special target. With a multi-drop link, the module number of the target destination computer link (serial communication) module station is set.

*lConnectUnitNumber*: Sets the module number of the computer link (serial communication) module or the Ethernet module for QnA/Qn. With a multi-drop link, the module number of the request source computer link (serial communication) module station is set. However, with a multi-drop via a CPU direct connection, the module number of the request source station is not required. ("0x00".) Sets "0x00" when not a multi-drop link. Sets a relay destination station number when using a QnA or Qn Ethernet module. ("0x00" is fixed when accessing within a host network.) If accessing other networks via MNET/10, sets the station number set in the parameter of the connected Ethernet module.

*lIONumber*: Sets the module I/O number. This parameter sets the target actual input/output No. (start I/O number ÷ 16) of the serial communication module and intelligent special module with a multi-drop link or intelligent special module access. **(With a multi-drop link, the station to pass through: request source station I/O number is set.)** Sets 0x3E0 to 0x3FF if accessing other stations via the host station CPU or network.

| Value | Target |
|---|---|
| **0 to 1FFh** | Communication system intelligent special module (start I/O number ÷ 16) |
| **200 to 3CFh** | Reserve |
| **3D0h** | Control system CPU module |
| **3D1h** | Standby system CPU module |
| **3D2h** | System A CPU module |
| **3D3h** | System B CPU module |
| **3D4h** | CPU module in other system |
| **3D5h to 3DBh** | Reserve |
| **3DCh** | System A communication peripheral equipment 1 |
| **3DDh** | System B communication peripheral equipment 1 |
| **3DEh** | System A communication peripheral equipment 2 |
| **3DFh** | System B communication peripheral equipment 2 |
| **3E0 to 3E3h** | Individual CPU module with multiple CPUs (Module 1 to 4) |
| **3F0h** | Global request with multiple CPUs |
| **3FCh** | Card next to CPU |
| **3FDh** | Peripheral equipment 2 |
| **3FEh** | Peripheral equipment 1 |
| **3FFh** | CPU module (including LM) |

*lCpuType*: Sets the target CPU (NC module) that performs communication.

| Value | Target |
|---|---|
| **CPU_Q17NNCCPU** | Q173NCCPU (C70) |

*lUnitType*: Sets the module connected to a physical port on the computer.

| Value | Meaning |
|---|---|
| **UNIT_ACPU** | Direct to ACPU-RS422 port |
| **UNIT_QCPU** | Direct to QnACPU-RS422 port |
| **UNIT_QNCPU** | Direct to QnCPU (Q mode) RS232C port |
| **UNIT_QNCPU_A** | Direct to QnCPU (A mode) RS232C |
| **UNIT_QNUSB** | Direct to QnCPU (Q mode) USB port |
| **UNIT_QNUSB_A** | Direct to QnCPU (A mode) USB port |
| **UNIT_QNMOTION** | Direct to Q motion -RS232C port |
| **UNIT_QNMOTIONUSB** | Direct to Q motion USB port |
| **UNIT_FXCPU** | Direct to FXCPU-RS422 port |
| **UNIT_C24** | Direct to C24 module for A |
| **UNIT_UC24** | Direct to UC24 module for A |
| **UNIT_QC24** | Direct to QC24 module for QnA |
| **UNIT_QJ71C24** | Direct to C24 module for Q |
| **UNIT_FXENET_ADP** | Connection to Ethernet adapter for FX |
| **UNIT_FX232BD** | Connection to FXCPU computer link (RS232C) |
| **UNIT_FX485BD** | Connection to FXCPU computer link (RS485) |
| **UNIT_E71** | Connection to Ethernet LAN for A |
| **UNIT_QE71** | Connection to Ethernet LAN for QnA |
| **UNIT_QJ71E71** | Connection to Ethernet LAN for Q |
| **UNIT_G4ACPU** | Direct to AJ65BT-G4 (-S3) module (ACPU access) |
| **UNIT_G4QCPU** | Direct to AJ65BT-G4 (-S3) module (QnA access) |
| **UNIT_G4QNCPU** | Direct to AJ65BT-G4-S3 module (Qn access) |
| **UNIT_MNET2BOARD** | Connection to MNET2 board |
| **UNIT_MNET10BOARD** | Connection to MNET/10 board |
| **UNIT_MNETHBOARD** | Connection to MNET/H board |
| **UNIT_MNETGBOARD** | Connection to CC-Link IE controller network board |
| **UNIT_CCLINKBOARD** | Connection to CC-Link board |
| **UNIT_MSPANUBOARD** | Connection to CPU board |
| **UNIT_AFBOARD** | Connection to AF board |
| **UNIT_EMEDBOARD** | Connection to EmEd board |
| **UNIT_SIMULATOR** | Connection to simulator (LLT) |
| **UNIT_QBF** | Connection to personal computer CPU for Q |
| **UNIT_SSCBOARD** | Connection to SSC net board |
| **UNIT_A900GOT** | Connection to GOT900 series / 1000 series |
| **UNIT_OTHER** | Generic connection |
| **UNIT_MNETGBOARD** | Connection to CC-Link IE controller network board |
| **UNIT_QNETHER** | Connection to QnCPU (Q mode) Ethernet port |
| **UNIT_QNETHER_DIRECT** | Direct connection to QnCPU (Q mode) Ethernet port |
| **UNIT_GOT_QJ71E71** | Connection to QJ71E71 module through GOT1000 series |
| **UNIT_GOT_QNETHER** | Connection to QnCPU Ethernet port through GOT1000 series |

*lPacketType*: Sets the computer link or Ethernet packet transmission format. The following format is set for this parameter.

| Value | Meaning |
|---|---|
| **PACKET_BINARY1** | Dedicated protocol format (when AJ71E71/AJ71QE71 is set) |
| **PACKET_ASCII1** | Dedicated protocol format (when AJ71(U)C24, AJ71E71/AJ71QE71 is set) |
| **PACKET_PLC1** | CPU protocol format (when AJ71E7/AJ71QE71 or other than the above is set) |

*lProtocolType*: Sets the communication protocol type of the module (board) to connect. Select connection through a serial port + modem with communication via AJAJ71QC24N/QJ71C24/LJ71C24 + modem. (If directly connecting to AJ71QC24N/QJ71C24/LJ71C24, select connection through a serial port.) Select connection through a shared memory server only with a simulator connection, and select connection through a Q bus only with a personal computer CPU connection.

| Value | Meaning |
|---|---|
| **PROTOCOL_MNET2** | Through MNET II board |
| **PROTOCOL_MNET10** | Through MNET/10 board |
| **PROTOCOL_MNETH** | Through MNET/10H and MNET/25H board |
| **PROTOCOL_MNETG** | Through CC-Link IE controller network board |
| **PROTOCOL_CCIEF** | Through CC-Link IE field network board |
| **PROTOCOL_EMED** | Through EmEd board |
| **PROTOCOL_SERIAL** | Through serial port |
| **PROTOCOL_USB** | Through USB port |
| **PROTOCOL_TCPIP** | Through TCP/IP |
| **PROTOCOL_UDPIP** | Through UDP/IP |
| **PROTOCOL_SHAREDMEMORY** | Through shared memory server |
| **PROTOCOL_CCLINK** | Through CC-Link board |
| **PROTOCOL_MSPANU** | Through CPU board |
| **PROTOCOL_AF** | Through AF board |
| **PROTOCOL_SSC** | Through SSC board |
| **PROTOCOL_TEL** | Through Q6TEL, A6TEL |
| **PROTOCOL_SERIALMODEM** | Through serial port + modem |
| **PROTOCOL_QBF** | Through Q bus |
| **PROTOCOL_USBGOT** | Through GOT1000 USB port |

*lPortNumber*: Sets the port number for connection between the physical port on the computer and the module set by *lUnitType*. For the connectable ports for the connection module, refer to the connectable ports remarks. However, an arbitrary value is set as the request source (personal computer) port number with an Ethernet connection. If "=0" is set as the port number, the MNET/10 routing method will be an automatic response method. In addition, set a fixed value of "5001" when not selecting an automatic response method via QE71 or when not setting TCP/IP for E71/QE71.

| 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 |

However, when an Ethernet connection is made, the following is applicable.

| Model | Protocol | | Port number |
|---|---|---|---|
| QJ71E71<br>AJ71QE71<br>(UDP) | UDP | Method other than automatic response | 5001 is fixed |
| | | Automatic response method | 0: Automatically assigns open ports in the personal computer |
| | | | Other than 0: Create sockets using specified port |
| | TCP | - | Fixed to 0: Automatically assigns open ports in the personal computer |
| AJ71QE71<br>(TCP)<br>AJ71E71 | UDP | Sets by matching the port number set in the sequence | |
| | TCP | If sets by the sequence: Set by matching the set port number | |
| | | If not set by the sequence: Arbitrarily set an open port in the personal computer | |

*lBaudRate*: Sets the baud rate with serial communication. This parameter can be one of the following values.

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

*lDataBits*: Sets the number of sent and received byte data bits (6 to 8).

*lParity*: Sets the parity bit. This parameter can be one of the following values.
It becomes effective only during serial communication.

| Value | Meaning |
|---|---|
| **EVENPARITY** | Even number |
| **ODDPARITY** | Odd number |
| **MARKPARITY** | Mark |
| **NOPARITY** | No parity |

*lStopBits*: Sets the number of stop bits used. This parameter can be one of the following values. It becomes effective only during serial communication.

| Value | Meaning |
|---|---|
| **ONESTOPBIT** | 1 stop bit |
| **ONE5STOPBITS** | 1.5 stop bits |
| **TWOSTOPBITS** | 2 stop bits |

*lControl*: Sets the control signal. This parameter can be one of the following values.
It becomes effective only during serial communication.

| 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*:Sets the connected host name (IP address) as a UNICODE character string with Ethernet communication. Set NULL when Ethernet is not set.

*lCpuTimeOut*: Sets the CPU monitoring timer with Ethernet communication. The unit is *250 ms. (The default is 4.)

*lTimeOut*: Sets the communication time-out value. The unit is ms. (The default is 1000 ms.)
* The time-out starts counting from when data communication ends.

*lSumCheck*: Sets whether a sum check is enabled or disabled. One of the following values is set for this parameter. It is enabled only when connecting through a computer link module or an A-series Ethernet module (TCP/IP).

| Value | Meaning |
|---|---|
| **TRUE** | There is a sum check |
| **FALSE** | There is no sum check |

*lSourceNetworkNumber*: Sets the request source network number when Ethernet for QnA and Qn (via AJ71QE71 and QJ71E71) is set

Sest the same network No. (network No. specified by the network parameter) as the Ethernet for a QnA or Qn connection.

*lSourceStationNumber*: Sets the request source station number (station number on the personal computer side) when an Ethernet for QnA and Qn (via AJ71QE71 and QJ71E71) is set

Sets the station number so as not to overlap with the QE71 station numbers set in the same Ethernet loop.

*lDestinationPortNumber*: Sets the port number of the target destination module when Ethernet is set. Sets the relay destination port number when accessing other networks. The following applies when other than an automatic response method, E71, or QE71 (TCP/IP).

- QnA (AJ71QE71) (UDP/IP) : "5001" is fixed
- Qn (QJ71E71) (TCP/IP) : "5002" is fixed
  If the target is a Q redundant CPU, any port number can also be set.
- Qn (QJ71E71) (UDP/IP) : “5001"is fixed
- Qn (Ethernet port) (UDP/IP) : “5006" is fixed
- Qn (Ethernet port) (TCP/IP) : “5007" is fixed
- Qn (Ethernet port direct communication) : “5008" is fixed

*lDestinationIONumber*: Sets the actual input/output No. (start I/O number ÷ 16) of the last access target station with a Qn multi-drop connection (via CC-Link, serial communication). (When the target is an intelligent special module.) If the target is a CPU module, set between 0x3F0 and 0x3FF. (Refer to "*lIONumber*".)

*lConnectChannelNumber*: Sets the connected channel No. (Ch1/Ch2) when connection of the serial communication module for Qn is set. Because this is used for system reservation, do not set anything. (Specify 0x00.)

*lMultiDropChannelNumber*: Sets the multi-drop connection channel No. (Ch1/Ch2) with a Qn multidrop connection. This will be invalid if any other connection is set.

| Value | Meaning |
|---|---|
| **0x01** | Connection to channel 1 |
| **0x02** | Connection to channel 2 |

*lThroughNetworkType*: Sets whether to include MNET/10 mode in the networks to pass through when accessing other stations via MELSECNET/10H.

| Value | Meaning |
|---|---|
| **0x00** | Does not include MNET/10 mode |
| **0x01** | Includes MNET/10 mode |

*lIntelligentPreferenceBit*: Sets whether to connect through the multi-drop link destination network with a Qn multi-drop connection (via CC-Link, serial communication). (This is to distinguish the host network module.)

| Value | Meaning |
|---|---|
| **0x00** | Multi-drop does not access other destination networks |
| **0x01** | Multi-drop accesses other destination networks |

*lDidPropertyBit*: With Q series host station intelligent special access (intelligent special module mounted to the CPU of the host station), by disabling the following settings, it is not necessary to set "*lUnitNumber*". (Set only by the module I/O number of "*lIONumber*".)

| Value | Meaning |
|---|---|
| **0x00** | Enables module number |
| **0x01** | Disables module number |

*lDsidPropertyBit*: With a Qn multi-drop connection, as the following are disabled, there is no need to set "*lDestinationIONumber*". However, if the following settings are disabled, make sure to enable "*lDidPropertyBit*". (Set by "*lUnitNumber*".)

| Value | Meaning |
|---|---|
| **0x00** | Enables the I/O number of the last access target station |
| **0x01** | Disables the I/O number of the last access target station |

*plRet*: Returns an error code.

**S_OK**: Normal termination

**EZNC_COMM_ALREADYOPENED**: Cannot be set because communication is already in progress

**EZ_ERR_DATA_TYPE**: Invalid argument data type

**EZ_ERR_DATA_RANGE**: Invalid argument data range

**EZ_ERR_NOT_SUPPORT**: Not supported

**□ Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

**□ Function**

Configures MELSEC communication settings. This is valid only for the C70.
This function is not supported with the M700/M800 Series. (EZ_ERR_NOT_SUPPORT is returned to plRet.)
Call before executing **Open2()**. If it is not called, an error will occur when **Open2()** is executed.
The setting details are retained until the object is released by **Release()**.
Temporarily, until **Close()** is performed if **Open2()** is performed, re-setting with **SetMelsecProcotol()** cannot be done. An error will occur.
If an error occurs in **SetMelsecProcotol()**, the setting before the error occurred is maintained.
The setting contents that resulted in the error are disabled.
SetsNULL to the pointer setting argument such as "lpcwszHostAddress" when not used.

(Note) This method does not support automation interfaces. Use is limited to a custom interface.

□ **Structure**

```
typedef struct EZNcStOpen{
    LONG      lNetworkNumber;                 // Network number
    LONG      lStationNumber;                 // Station number
    LONG      lUnitNumber;                    // Module number
    LONG      lConnectUnitNumber;             // Module number
    LONG      lIONumber;                      // Module I/O number
    LONG      lCpuType;                       // Target CPU
    LONG      lUnitType;                      // Connected module
    LONG      lPacketType;                    // Packet transmission type
    LONG      lProtocolType;                  // Communication protocol type
    LONG      lPortNumber;                    // Connection port number
    LONG      lBaudRate;                      // Baud rate
    LONG      lDataBits;                      // Number of byte data bits
    LONG      lParity;                        // Parity bit
    LONG      lStopBits;                      // Stop bit
    LONG      lControl;                       // Control signal
    WCHAR *   lpcwszHostAddress;              // Connection host name (IP address)
    LONG      lCpuTimeOut;                    // CPU monitoring timer
    LONG      lTimeOut;                       // Time-out value
    LONG      lSumCheck;                      // Sum check
    LONG      lSourceNetworkNumber;           // Request source network number
    LONG      lSourceStationNumber;           // Request source station number
                                                (personal computer side station number)
    LONG      lDestinationPortNumber;         // Port number
    LONG      lDestinationIONumber;           // Actual input/output No.
    LONG      lConnectChannelNumber;          // Channel No.
    LONG      lMultiDropChannelNumber;        // Multi-drop connection channel No.
    LONG      lThroughNetworkType;            // MNET/10 mode
    LONG      lIntelligentPreferenceBit;      // Via multi-drop link destination
    LONG      lDidPropertyBit;                // Intelligent special module setting
    LONG      lDsidPropertyBit;               // Multi-drop connection setting
} EZNCST_OPEN;
```

□ **Reference**

**Open2(), Close()**

□ **Specification**

**C70** **M700** **M800**

## 2.4.1 IEZNcSystem::GetVersion Get NC system S/W number, name

□ **Custom call procedure**

```
HRESULT      GetVersion(
                      LONG lAxisNo,                 // (I) Axis number
                      LONG lIndex,                  // (I) Parameter number
                      LPOLESTR* lppwszBuffer,       // (O) NC system S/W number, name
                      LONG* plRet                   // (O) Error code
                       )
```

□ **Automation call procedure**

```
             System_GetVersion(
                      lAxisNo As LONG               // (I) Axis number
                      lIndex As LONG                // (I) Parameter number ber
                      lppwszBuffer As STRING*       // (O) NC system S/W number, name
                      ) As LONG                     // (O) Error code
```

□ **Argument**

*lAxisNo*: Sets the axis number (From Axis 1 = from **1**)

*lIndex*: Sets the parameter number. Refer to the table below.

*lppwszBuffer*: Returns the system S/W number, name, and control S/W version as a **UNICODE** character string.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZ_ERR_DATA_TYPE**: Invalid argument data type
**EZNC_DATA_READ_DATASIZE**: Application does not fit into prepared buffer
**EZNC_DATA_READ_READ**: Data is not readable

| *lIndex* | Description | Data range |
|---|---|---|
| **0** | NC system S/W number, name, and PLC version | Depends on the system specifications. |
| **1** | Control unit, extension unit | Depends on the system specifications. |
| **2** | RIO unit, terminal RIO unit<br>Axis setting is necessary only for C70. | Depends on the system specifications. |

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Failure |

| | |
|---|---|
| □<br>**Function** | Gets the various NC system S/W version information as a **UNICODE** character string.<br>0: Gets the system S/W number, name, and PLC version of the NC system.<br>The format of the character string data becomes as follows.<br>NC system S/W number\tNC system name\tprogrammable controller system number\0<br>A **TAB code** is inserted between the NC system number and the NC system S/W name.<br>The end of the data becomes a **NULL** code.<br>Output example: "BND-2005W000-A0    MITSUBISHI CNC 830WM"<br>If there is no item, a **TAB** code will follow. If a termination item does not exist, a **NULL** code will follow the **TAB** code.<br><br>1: Gets the control unit and extension unit versions.<br>The format of the character string data becomes as follows.<br>Control unit number\tExtension unit number\0<br>A **TAB** code is inserted between the control unit number and the extension unit number.<br>The end of the data becomes a **NULL** code.<br><br>2: Gets the RIO unit and terminal RIO unit versions.<br>The format of the character string data becomes as follows.<br>RIO unit number\tTerminal RIO unit number\0<br>M700 has 24 items with RIO unit 1\t RIO unit 2\t…\0.<br>M700/M800 series have up to 32 items (*) with RIO unit 1¥t RIO unit 2¥t5 ··· ¥0.<br>* Confirm the number of RIO unit with MTB.<br>A **TAB** code is inserted between the RIO unit number and the terminal RIO unit number.<br>The end of the data becomes a **NULL** code.<br>As the character string area memory is allocated in this product, clients using VC++ need to release the character string area memory explicitly with **CoTaskMemFree()**. |
| □<br>**Reference** | |
| □<br>**Specification** | |

## 2.4.2 IEZNcSystem::GetSystemInformation Get NC system information

□ **Custom call procedure**

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

□ **Automation call procedure**

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

□ **Argument**

*lType*: Sets the NC system information type. Refer to the table below.

*plSystem*: Returns the NC system information.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZ_ERR_DATA_TYPE**: Invalid argument data type
**EZNC_DATA_READ_READ**: Data is not readable

| *lType* | Description | Data range |
|---|---|---|
| 0 | Part system enabled/disabled | 0: Part system disabled<br>1: Part system enabled |
| 1 | Number of axes each part system | This will be different for NC systems with 1 [Axis] or more. |

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets information regarding the NC system.

□ **Reference**

□ **Specification**

System

## 2.4.3 IEZNcSystem::GetAlarm Get alarm information

□ **Custom call procedure**

```
HRESULT     GetAlarm(
                 LONG lMessageNumber,        // (I) Number of messages to get
                 LONG lAlarmType,            // (I) Alarm type to get
                 LPOLESTR* lppwszBuffer,     // (O) Message character string
                 LONG* plRet                 // (O) Error code
                 )
```

□ **Automation call procedure**

```
            System_GetAlarm(
                 lMessageNumber As LONG      // (I) Number of messages to get
                 lAlarmType As LONG          // (I) Alarm type to get
                 lppwszBuffer As STRING*     // (O) Message character string
                 ) As LONG                   // (O) Error code
```

□ **Argument**

*lMessageNumber*: Sets the number of messages to get. Value: **1 to 10 (maximum)**

*lAlarmType*: Sets the alarm type to get.

| Value | Meaning |
|---|---|
| **M_ALM_NC_ALARM** | NC alarm |
| **M_ALM_STOP_CODE** | Stop code |
| **M_ALM_PLC_ALARM** | PLC alarm message |
| **M_ALM_OPE_MSG** | Operator message |
| **M_ALM_ALL_ALARM** | No alarm type distinction |

*lppwszBuffer*: Gets the alarm message as a **UNICODE** character string.
The message format includes **CR, LF** codes to distinguish between messages. In addition, **NULL** is inserted at the end of the message.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_OPE_CURRALM_ADDR**: Invalid part systems, axes settings
**EZNC_OPE_CURRALM_ALMTYPE**: Invalid alarm type
**EZNC_OPE_CURRALM_DATAERR**: Error in communication data between NC and personal computer
**EZNC_OPE_CURRALM_DATASIZE**: Application does not fit into prepared buffer
**EZNC_OPE_CURRALM_NOS**: Invalid number of got messages
**EZ_ERR_NOT_SUPPORT**：Not supported

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets the alarm message of the alarm currently generated in the setting NC control unit. The language of the alarm message adheres to an NC parameter (#1043 lang).
Messages are retrieved in descending order of importance.
As the character string area memory is allocated in this product, clients using VC++ need to release the character string area memory explicitly with **CoTaskMemFree()**.
This function is not supported with the M800 Series. (EZ_ERR_NOT_SUPPORT is returned to plRet.)

□ **Reference**

□ **Specification**

System (All systems when 0)

## 2.4.4 IEZNcSystem2::GetAlarm2 Get alarm information

□ **Custom call procedure**

```
HRESULT      GetAlarm2(
                 LONG lMessageNumber,          // (I) Number of messages to get
                 LONG lAlarmType,              // (I) Alarm type to get
                 LPOLESTR* lppwszBuffer,       // (O) Message character string
                 LONG* plRet                   // (O) Error code
                 )
```

□ **Automation call procedure**

```
             System_GetAlarm2(
                 lMessageNumber As LONG        // (I) Number of messages to get
                 lAlarmType As LONG            // (I) Alarm type to get
                 lppwszBuffer As STRING*       // (O) Message character string
                 ) As LONG                     // (O) Error code
```

□ **Argument**

*lMessageNumber*: Sets the number of messages to get. Value: **1 to 10 (maximum)**

*lAlarmType*: Sets the alarm type to get.

| Value | Meaning |
|---|---|
| **M_ALM_NC_ALARM** | NC alarm |
| **M_ALM_STOP_CODE** | Stop code |
| **M_ALM_PLC_ALARM** | PLC alarm message |
| **M_ALM_OPE_MSG** | Operator message |
| **M_ALM_ALL_ALARM** | No alarm type distinction |

*lppwszBuffer*: Gets the alarm message as a **UNICODE** character string.
The message format includes **CR, LF** codes to distinguish between messages. In addition, **NULL** is inserted at the end of the message.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_OPE_CURRALM_ADDR**: Invalid system, spindle specification
**EZNC_OPE_CURRALM_ALMTYPE**: Invalid alarm type
**EZNC_OPE_CURRALM_DATAERR**: Error in communication data between NC and personal computer
**EZNC_OPE_CURRALM_DATASIZE**: Application does not fit into prepared buffer
**EZNC_OPE_CURRALM_NOS**: Invalid number of got messages

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets the alarm message of the alarm currently generated in the setting NC control unit. The language of the alarm message adheres to an NC parameter (#1043 lang).
Messages are retrieved in descending order of importance.
As the character string area memory is allocated in this product, clients using VC++ need to release the character string area memory explicitly with **CoTaskMemFree()**.

□ **Reference**

□ **Specification**

System (All systems when 0)

**C70** **M700** **M800**

## 2.5.1 IEZNcPosition::GetWorkPosition Get workpiece coordinate position

□ **Custom call procedure**

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

□ **Automation call procedure**

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

□ **Argument**

*lAxisNo*: Sets the axis number (From Axis 1 = from **1**)

*pdPosition*: Returns the workpiece coordinate position of the set axis number of the set part system.
Data range: -99,999.999 to 99,999.999 [mm]
*lSkipOn* : Sets the skip on flag.

| Value | Meaning |
|---|---|
| **1** | Skip is on |
| **0** | Normal |

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZ_ERR_DATA_TYPE**: Invalid argument data type
**EZNC_DATA_READ_ADDR**: Invalid part system, axis No. setting
**EZNC_DATA_READ_READ**: Data is not readable

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets the workpiece coordinate position of the set system/axis number
If **1** is set for the skip on flag, the workpiece coordinate position at the time the skip on signal is input will be got.

□ **Reference**

□ **Specification**

System  Axis number

**C70** **M700** **M800**

## 2.5.2 IEZNcPosition::GetWorkPosition2 — Get workpiece coordinate position

□ **Custom call procedure**

```
HRESULT    GetWorkPosition2(
                   LONG lAxisNo,              // (I) Axis number
                   DOUBLE* pdPosition,        // (O) Workpiece coordinate position
                   LONG lSkipOn,              // (I) Skip on flag
                   LONG* plRet                // (O) Error code
                   )
```

□ **Automation call procedure**

```
           Position_GetWorkPosition2(
                   lAxisNo As LONG            // (I) Axis number
                   pdPosition As DOUBLE*      // (O) Workpiece coordinate position
                   lSkipOn As LONG            // (I) Skip on flag
                   ) As LONG                  // (O) Error code
```

□ **Argument**

*lAxisNo*: Sets the axis number (From Axis 1 = from **1**)

*pdPosition*: Returns the workpiece coordinate position of the set axis number of the set part system.
Data range: -99,999.999 to 99,999.999 [mm]
This will vary according to the NC system specifications and parameters.

*lSkipOn* : Sets the skip on flag.

| Value | Meaning |
|---|---|
| **1** | Skip is on |
| **0** | Normal |

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZ_ERR_DATA_TYPE**: Invalid argument data type
**EZNC_DATA_READ_ADDR**: Invalid part system, axis No. setting
**EZNC_DATA_READ_READ**: Data is not readable
**EZ_ERR_NOT_SUPPORT**: Not supported

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets the workpiece coordinate position of the set system/axis number.
If **1** is set for the skip on flag, the workpiece coordinate position at the time the skip on signal is input will be got.

□ **Reference**

□ **Specification**

System | Axis number

## 2.5.3 IEZNcPosition::GetMachinePosition Get machine position

□ **Custom call procedure**

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

□ **Automation call procedure**

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

□ **Argument**

*lAxisNo*: Sets the axis. (From Axis 1 = from **1**)

*pdPosition*: Returns the machine position of the set axis number of the set part system.
Data range: -99,999.999 to 99,999.999 [mm]
This will vary according to the NC system specifications and parameters.

*lSkipOn* : Sets the skip on flag.

| Value | Meaning |
|---|---|
| **1** | Skip is on |
| **0** | Normal |

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_ADDR**: Invalid part system, axis number setting
**EZNC_DATA_READ_READ**: Data is not readable

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets the machine coordinate position for the set system/axis number (coordinate position in a basic machine coordinate system).
If **1** is set for the skip on flag, the machine coordinate position at the time the skip on signal is input will be got.

□ **Reference**

□ **Specification**

System  Axis number

## 2.5.4 IEZNcPosition::GetMachinePosition2 Get machine position

□ **Custom call procedure**

```
HRESULT    GetMachinePosition2(
                  LONG lAxisNo,              // (I) Axis number
                  DOUBLE* pdPosition,        // (O) Machine position
                  LONG lSkipOn,              // (I) Skip on flag
                  LONG* plRet                // (O) Error code
                  )
```

□ **Automation call procedure**

```
           Position_GetMachinePosition2(
                  lAxisNo As LONG            // (I) Axis number
                  pdPosition As DOUBLE*      // (O) Machine position
                  lSkipOn As LONG            // (I) Skip on flag
                  ) As LONG                  // (O) Error code
```

□ **Argument**

*lAxisNo*: Sets the axis. (From Axis 1 = from **1**)

*pdPosition*: Returns the machine position of the set axis number of the set part system.
Data range: -99,999.999 to 99,999.999 [mm]
This will vary according to the NC system specifications and parameters.

*lSkipOn* : Sets the skip on flag.

| Value | Meaning |
|---|---|
| **1** | Skip is on |
| **0** | Normal |

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_ADDR**: Invalid part system, axis number setting
**EZNC_DATA_READ_READ**: Data is not readable

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets the machine coordinate position for the set system/axis number (coordinate position in a basic machine coordinate system).
If **1** is set for the skip on flag, the machine coordinate position at the time the skip on signal is input will be got.

□ **Reference**

□ **Specification**

System  Axis number

## 2.5.5 IEZNcPosition::GetCurrentPosition Get relative position

□ **Custom call procedure**

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

□ **Automation call procedure**

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

□ **Argument**

*lAxisNo*: Sets the axis No. (From Axis 1 = from **1**)

*pdPosition*: Returns the relative position from the position at a completion of the dog type zero point return or from the preset position configured by G92/origin set/counter set.
Data range: -99,999.999 to 99,999.999 [mm]
This will vary according to the NC system specifications and parameters.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_ADDR**: Invalid part system, axis number setting
**EZNC_DATA_READ_READ**: Data is not readable

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets the relative position to the position at a completion of the dog type zero point return or to the preset position configured by G92/origin set/counter set of the set system/axis number.

□ **Reference**

□ **Specification**

System  Axis number

## 2.5.6 IEZNcPosition::GetDistance Get remaining command

□ **Custom call procedure**

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

□ **Automation call procedure**

```
           Position_GetDistance(
                lAxisNo As LONG                // (I) Axis number
                pdDistance As DOUBLE*          // (O) Rmaining command
                lSkipOn As LONG                // (I) Skip on flag
                ) As LONG                      // (O) Error code
```

□ **Argument**

*lAxisNo*: Sets the axis number (From Axis 1 = from **1**)

*pdDistance*: Returns the remaining command of the travel command being executed in the set axis No. of the set system.
Data range: -99,999.999 to 99,999.999 [mm]
This will vary according to the NC system specifications and parameters.

*lSkipOn* : Sets the skip on flag.

| Value | Meaning |
|---|---|
| **1** | Skip is on |
| **0** | Normal |

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZ_ERR_DATA_TYPE**: Invalid argument data type
**EZNC_DATA_READ_ADDR**: Invalid part system, axis number setting
**EZNC_DATA_READ_READ**: Data is not readable

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets the remaining command of the travel command being executed for the set part system/axis number.
If **1** is set for the skip on flag, the remaining command of the travel command at the time the skip on signal is input will be got.

□ **Reference**

□ **Specification**

System Axis number

## 2.5.7 IEZNcPosition::GetDistance Get remaining command

□ **Custom call procedure**

```
HRESULT      GetDistance2(
                       LONG lAxisNo,                  // (I) Axis number
                       DOUBLE* pdDistance,            // (O) Remaining command
                       LONG lSkipOn,                  // (I) Skip on flag
                       LONG* plRet                    // (O) Error code
                       )
```

□ **Automation call procedure**

```
             Position_GetDistance2(
                       lAxisNo As LONG                // (I) Axis number
                       pdDistance As DOUBLE*          // (O) Remaining Command r
                       lSkipOn As LONG                // (I) Skip on flag
                       ) As LONG                      // (O) Error code
```

□ **Argument**

*lAxisNo*: Sets the axis number. (From Axis 1 = from **1**)

*pdDistance*: Returns the remaining command of the travel command being executed in the set axis number. of the set system.
Data range: -99,999.999 to 99,999.999 [mm]
This will vary according to the NC system specifications and parameters.

*lSkipOn* : Sets the skip on flag.

| Value | Meaning |
|---|---|
| **1** | Skip is on |
| **0** | Normal |

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZ_ERR_DATA_TYPE**: Invalid argument data type
**EZNC_DATA_READ_ADDR**: Invalid part system, axis number setting
**EZNC_DATA_READ_READ**: Data is not readable

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets the remaining command of the travel command being executed for the set system/axis number.
If **1** is set for the skip on flag, the remaining command of the travel command at the time the skip on signal is input will be got.

□ **Reference**

□ **Specification**

System  Axis number

## 2.5.8 IEZNcPosition::GetNextDistance Get next command

□ **Custom call procedure**

```
HRESULT       GetNextDistance(
                      LONG lAxisNo,                  // (I) Axis No. setting
                      DOUBLE* pdDistance,            // (O) Next command
                      LONG* plRet                    // (O) Error code
                      )
```

□ **Automation call procedure**

```
              Position_GetNextDistance(
                      lAxisNo As LONG                // (I) Axis No. setting
                      pdDistance As DOUBLE*          // (O) Next command
                      ) As LONG                      // (O) Error code
```

□ **Argument**

*lAxisNo*: Sets the axis No. (From Axis 1 = from **1**)

*pdDistance*: Returns the travel command of the next block for the set axis No. of the set part system.
Data range: -99,999.999 to 99,999.999 [mm]
This will vary according to the NC system specifications and parameters.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_ADDR**: Invalid part system, axis No. setting
**EZNC_DATA_READ_READ**: Data is not readable

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets the travel command of the next block for the set system/axis No.

□ **Reference**

□ **Specification**

System Axis number

C70 M700 M800

## 2.5.9 IEZNcPosition::GetFeedSpeed Get feed speed

**□ Custom call procedure**

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

**□ Automation call procedure**

```
             Position_GetFeedSpeed(
                     lFeedType As LONG                        // (I) Feed speed type
                     pdSpeed As DOUBLE*                       // (O) Feed speed
                     ) As LONG                                // (O) Error code
```

**□ Argument**

*lFeedType*: Sets the type of feed speed to get.

| Value | Meaning |
|---|---|
| **0** | F command feed speed (FA) |
| **1** | Manual effective feed speed (FM) |
| **2** | Synchronization feed speed (FS) |
| **3** | Automatic effective feed speed (Fc) |
| **4** | Screw lead (FE) |

*pdSpeed*: Returns the feed speed of the specified system.

| Data range: | FA | : 0.000 to 1000000.000 [mm/min] |
|---|---|---|
| | FM | : 0.000 to 1000000.000 [mm/min] |
| | FS | : 0.000 to 1000.0000000 [mm/rev] |
| | FC | : 0.000 to 1000000.000 [mm/min] |
| | FE | : 0.000 to 1000.0000000 [mm] |

| Feed speed type | Data range | |
|---|---|---|
| | **M700/M800 series** | **C70** |
| FA | 0.000 to 1,000,000.000 [mm/min] | 0.000 to 1,000,000.000 [mm/min] |
| FM | 0.000 to 1,000,000.000 [mm/min] | 0.000 to 1,000,000.000 [mm/min] |
| FS | 0.000 to 1,000.0000000 [mm/rev] | 0.000 to 1,000.0000000 [mm/rev] |
| Fc | 0.000 to 1,000,000.000 [mm/min] | 0.000 to 1,000,000.000 [mm/min] |
| FE | 0.000 to 1,000.0000000 [mm]<br>(Unit will vary) | 0.000 to 1,000.0000000 [mm] |

The number of digits is determined by the NC type, options, and MTB setting values (parameters).

*plRet*: Returns an error code. (Upon automation, the return value is used.)

**S_OK**: Normal termination
**EZ_ERR_DATA_TYPE**: Invalid argument data type
**EZNC_DATA_READ_ADDR**: Invalid part system, axis No. setting
**EZNC_DATA_READ_READ**: Data is not readable

**□ Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

**□ Function**

Gets the feed speed of the set system.

**□ Reference**

**□ Specification**

System

## 2.5.10 IEZNcPosition::GetTCPSpeed Get tip speed

□ **Custom call procedure**

```
HRESULT     GetTCPSpeed (
                 DOUBLE* pdPosition,          // (O) Tip speed
                 LONG* plRet                  // (O) Error code
                 )
```

□ **Automation call procedure**

```
            Position_ GetTCPSpeed (
                 pdPosition As DOUBLE*        // (O) Tip speed
                 ) As LONG                    // (O) Error code
```

□ **Argument**

*pdPosition*: Returns the tip speed of the set system.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_ADDR**: Invalid system specification
**EZNC_DATA_READ_READ**: Data is not readable
**EZ_ERR_NOT_SUPPORT**: Not supported

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets the tip speed of the set part system.
If **1** is set for the skip on flag, the workpiece coordinate position at the time the skip on signal is input will be got.
This is valid only for the M700/M800 series.
This function is not supported with C70.

□ **Reference**

□ **Specification**

System

## 2.5.11 IEZNcPosition::GetManualOverlap Get manual interrupt amount

□ **Custom call procedure**

```
HRESULT     GetManualOverlap(
                    LONG lAxisNo,             // (I) Axis No. setting
                    LONG lType,               // (I) Type
                    DOUBLE* pdLength,         // (O) Manual interrupt amount
                    LONG* plRet               // (O) Error code
                    )
```

□ **Automation call procedure**

```
            Position_GetManualOverlap(
                    lAxisNo As LONG           // (I) Axis No. setting
                    lType As LONG             // (I) Type
                    pdLength As DOUBLE*       // (O) Manual interrupt amount
                    ) As LONG                 // (O) Error code
```

□ **Argument**

*lAxisNo*: Sets the axis No. (From Axis 1 = from **1**)

*lType*: Sets the manual interrupt amount type to get.

| Value | Meaning |
|---|---|
| **0** | If getting the manual interrupt amount while the manual ABS switch is off |
| **1** | If getting the manual interrupt amount while the manual ABS switch is on |

*pdLength*: Returns the manual interrupt amount for the set axis No. of the set part system.
Data range: -99,999.999 to 99,999.999 [mm]
This will vary according to the NC system specifications and parameters.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZ_ERR_DATA_TYPE**: Invalid argument data type
**EZNC_DATA_READ_ADDR**: Invalid part system, axis No. setting
**EZNC_DATA_READ_READ**: Data is not readable

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets the manual interrupt amount of the set part system/axis.

□ **Reference**

□ **Specification**

System Axis number

## 2.5.12 IEZNcPosition::GetManualOverlap2 Get manual interrupt amount

□ **Custom call procedure**

```
HRESULT   GetManualOverlap2(
                    LONG lAxisNo,              // (I) Axis No.
                    LONG lType,                // (I) Type
                    DOUBLE* pdLength,          // (O) Manual interrupt amount
                    LONG* plRet                // (O) Error code
                    )
```

□ **Automation call procedure**

```
          Position_GetManualOverlap2(
                    lAxisNo As LONG            // (I) Axis No.
                    lType As LONG              // (I) Type
                    pdLength As DOUBLE*        // (O) Manual interrupt amount
                    ) As LONG                  // (O) Error code
```

□ **Argument**

*lAxisNo*: Sets the axis No. (From Axis 1 = from **1**)

*lType*: Sets the manual interrupt amount type to get.

| Value | Meaning |
|---|---|
| **0** | If geting the manual interrupt amount while the manual ABS switch is off |
| **1** | If getting the manual interrupt amount while the manual ABS switch is on |

*pdLength*: Returns the manual interrupt amount for the set axis No. of the set part system.
Data range: -99,999.999 to 99,999.999 [mm]
This will vary according to the NC system specifications and parameters.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZ_ERR_DATA_TYPE**: Invalid argument data type
**EZNC_DATA_READ_ADDR**: Invalid part system, axis specification
**EZNC_DATA_READ_READ**: Data is not readable
**EZ_ERR_NOT_SUPPORT**: Not supported

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets the manual interrupt amount of the set part axis of the set part system.

□ **Reference**

□ **Specification**

System  Axis number

## 2.5.13 IEZNcPosition::GetProgramPosition Get program position

□ **Custom call procedure**

```
HRESULT      GetProgramPosition(
                      LONG lAxisNo,                 // (I) Axis No.
                      DOUBLE* pdPosition,           // (O) Program position
                      LONG* plRet                   // (O) Error code
                      )
```

□ **Automation call procedure**

```
             Position_GetProgramPosition(
                      lAxisNo As LONG               // (I) Axis No.
                      pdPosition As DOUBLE*         // (O) Program position
                      ) As LONG                     // (O) Error code
```

□ **Argument**

*lAxisNo*: Sets the axis No. (From Axis 1 = from **1**)

*pdPosition*: Returns the program position.
Data range: -99,999.999 to 99,999.999 [mm]
This will vary according to the NC system specifications and parameters.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_ADDR**: Invalid part system, axis No. setting
**EZNC_DATA_READ_READ**: Data is not readable

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets program position.

□ **Reference**

□ **Specification**

System Axis number

## 2.5.14 IEZNcPosition::GetProgramPosition3 Get program position

□ **Custom call procedure**

```
HRESULT         GetProgramPosition3(
                        LONG lAxisNo,                   // (I) Axis No.
                        DOUBLE* pdPosition,             // (O) Program position
                        LONG* plRet                     // (O) Error code
                        )
```

□ **Automation call procedure**

```
                Position_GetProgramPosition3(
                        lAxisNo As LONG                 // (I) Axis No.
                        pdPosition As DOUBLE*           // (O) Program position
                        ) As LONG                       // (O) Error code
```

□ **Argument**

*lAxisNo*: Sets the axis No. (From Axis 1 = from **1**)

*pdPosition*: Returns the program position.
Data range: -99,999.999 to 99,999.999 [mm]
This will vary according to the NC system specifications and parameters.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_ADDR**: Invalid part system, axis No.setting
**EZNC_DATA_READ_READ**: Data is not readable

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets program position.

□ **Reference**

□ **Specification**

System Axis number

## 2.5.15 IEZNcPosition::GetTCPMachinePosition Get tip machine position

□ **Custom call procedure**

```
HRESULT     GetTCPMachinePosition (
                    LONG lAxisNo,              // (I) Axis No.
                    DOUBLE* pdPosition,        // (O) Tip machine position
                    LONG* plRet                // (O) Error code
                    )
```

□ **Automation call procedure**

```
            Position_ GetTCPMachinePosition (
                    lAxisNo As LONG            // (I) Axis No.
                    pdPosition As DOUBLE*      // (O) Tip machine position
                    ) As LONG                  // (O) Error code
```

□ **Argument**

*lAxisNo*: Set the axis No. (From Axis 1 = from **1**)

*pdPosition*: Returns the tip machine position of the set axis No. of the set part system.
Data range: -99,999.999 to 99,999.999 [mm]
This will vary according to the NC system specifications and parameters.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_ADDR**: Invalid part system, axis No. setting
**EZNC_DATA_READ_READ**: Data is not readable
**EZ_ERR_NOT_SUPPORT**: Not supported

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets the tip machine position.
This is valid only for the M700/M800 series.
This function is not supported with C70.

□ **Reference**

□ **Specification**

System Axis number

## 2.5.16 IEZNcPosition::GetTCPWorkPosition Get tip workpiece position

□ **Custom call procedure**

```
HRESULT     GetTCPWorkPosition (
                    LONG lAxisNo,                // (I) Axis No.
                    DOUBLE* pdPosition,          // (O) Tip workpiece position
                    LONG* plRet                  // (O) Error code
                    )
```

□ **Automation call procedure**

```
            Position_ GetTCPWorkPosition (
                    lAxisNo As LONG              // (I) Axis No.
                    pdPosition As DOUBLE*        // (O) Tip workpiece position
                    ) As LONG                    // (O) Error code
```

□ **Argument**

*lAxisNo*: Sets the axis No. (From Axis 1 = from **1**)

*pdPosition*: Returns the tip workpiece position of the set axis No. of the set part system.
Data range: -99,999.999 to 99,999.999 [mm]
This will vary according to the NC system specifications and parameters.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_ADDR**: Invalid part system, axis No. setting
**EZNC_DATA_READ_READ**: Data is not readable
**EZ_ERR_NOT_SUPPORT**: Not supported

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets the tip workpiece position.
This is valid only for the M700/M800 series.
This function is not supported with C70.

□ **Reference**

□ **Specification**

System Axis number

## 2.5.17 IEZNcPosition::GetFeedbackPosition Get feedback position

□ **Custom call procedure**

```
HRESULT     GetFeedbackPosition (
                    LONG lAxisNo,                   // (I) Axis No.
                    DOUBLE* pdPosition,             // (O) Feedback position
                    LONG* plRet                     // (O) Error code
                    )
```

□ **Automation call procedure**

```
            Position_ GetFeedbackPosition (
                    lAxisNo As LONG                 // (I) Axis No.
                    pdPosition As DOUBLE*           // (O) Feedback position
                    ) As LONG                       // (O) Error code
```

□ **Argument**

*lAxisNo*: Sets the axis No. (From Axis 1 = from **1**)

*pdPosition*: Returns the feedback position of the set axis of the set part system.
Data range: -99,999.999 to 99,999.999 [mm]
This will vary according to the NC system specifications and parameters.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_ADDR**: Invalid part system, axis No. setting
**EZNC_DATA_READ_READ**: Data is not readable
**EZ_ERR_NOT_SUPPORT**: Not supported

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets the feedback position.
This is valid only for the M700/M800 series.
This function is not supported with C70.

□ **Reference**

□ **Specification**

System  Axis number

## 2.5.18 IEZNcPosition::GetTableCoordinationPosition Get table coordinate system counter

□ **Custom call procedure**

```
HRESULT    GetTableCoordinationPosition (
                      LONG lAxisNo,             // (I) Axis No.
                      DOUBLE* pdPosition,       // (O) Table coordinate system counter
                      LONG* plRet               // (O) Error code
                      )
```

□ **Automation call procedure**

```
           Position_ GetTableCoordinationPosition (
                      lAxisNo As LONG           // (I) Axis No.
                      pdPosition As DOUBLE*     // (O) Table coordinate system counter
                      ) As LONG                 // (O) Error code
```

□ **Argument**

*lAxisNo*: Sets the axis No. (From Axis 1 = from **1**)

*pdPosition*: Returns the table coordinate system counter of the set axis of the set part system.
Data range: -99,999.999 to 99,999.999 [mm]
This will vary according to the NC system specifications and parameters.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_ADDR**: Invalid part system, axis No. setting
**EZNC_DATA_READ_READ**: Data is not readable
**EZ_ERR_NOT_SUPPORT**: Not supported

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets the table coordinate system counter.
This is valid only for the M700/M800 series.
This function is not supported with C70.

□ **Reference**

□ **Specification**

System Axis number

## 2.5.19 IEZNcPosition::GetWorkInstallationPosition Get workpiece installation coordinate system counter

□ **Custom call procedure**

```
HRESULT     GetWorkInstallationPosition (
                   LONG lAxisNo,           // (I) Axis No.
                   DOUBLE* pdPosition,     // (O) Workpiece installation coordinate system counter
                   LONG* plRet             // (O) Error code
                   )
```

□ **Automation call procedure**

```
            Position_ GetWorkInstallationPosition (
                   lAxisNo As LONG         // (I) Axis No.
                   pdPosition As DOUBLE* // (O) Workpiece installation coordinate system counter
                   ) As LONG               // (O) Error code
```

□ **Argument**

*lAxisNo*: Sets the axis No. (From Axis 1 = from **1**)

*pdPosition*: Returns the workpiece installation coordinate system counter of the set axis of the set system.
Data range: -99,999.999 to 99,999.999 [mm]
This will vary according to the NC system specifications and parameters.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_ADDR**: Invalid part system, axis No. setting
**EZNC_DATA_READ_READ**: Data is not readable
**EZ_ERR_NOT_SUPPORT**: Not supported

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets the workpiece installation coordinate system counter.
This is valid only for the M700/M800 series.
This function is not supported with C70.

□ **Reference**

□ **Specification**

System Axis number

## 2.5.20 IEZNcPosition::GetInclinedSurfacePosition Get inclined surface coordinate system counter

□ **Custom call procedure**

```
HRESULT       GetInclinedSurfacePosition (
                        LONG lAxisNo,          // (I) Axis No.
                        DOUBLE* pdPosition,    // (O) Inclined surface coordinate system counter
                        LONG* plRet            // (O) Error code
                        )
```

□ **Automation call procedure**

```
              Position_ GetInclinedSurfacePosition (
                        lAxisNo As LONG        // (I) Axis No.
                        pdPosition As DOUBLE*    // (O) Inclined surface coordinate system counter
                        ) As LONG              // (O) Error code
```

□ **Argument**

*lAxisNo*: Sets the axis No. (From Axis 1 = from **1**)

*pdPosition*: Returns the inclined surface coordinate system counter of the set axis No. of the set part system.
Data range: -99,999.999 to 99,999.999 [mm]
This will vary according to the NC system specifications and parameters.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_ADDR**: Invalid part system, axis No. setting
**EZNC_DATA_READ_READ**: Data is not readable
**EZ_ERR_NOT_SUPPORT**: Not supported

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets the inclined surface coordinate system counter.
This is valid only for the M700/M800 series.
This function is not supported with C70.

□ **Reference**

□ **Specification**

System Axis number

C70 M700 M800

## 2.6.1 IEZNcCommand2::GetGCodeCommand Get G code modal command value

□ **Custom call procedure**

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

□ **Automation call procedure**

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

□ **Argument**

*lType*: Sets the G code modal command value type to get.
The following describes examples for the M700/M800 series M system. The content will differ depending on the type and the settings.

| Value | Meaning |
|---|---|
| **1** | Group 1 (Interpolation mode)<br>G00, G01, G02, G03, G33, G02.1, G03.1, G02.3, G03.3, G02.4,G03.4,<br>G062 command modal |
| **2** | Group 2 (Plane selection) G17, G18, G19 command modal |
| **3** | Group 3 (Absolute) G90, (incremental) G91 command modal |
| **4** | Group 4 (Chuck barrier) G22, G23 command modal |
| **5** | Group 5 (Feed mode) G93, G94, G95 command modal |
| **6** | Group 6 (Inch) G20, (millimeter) G21 command modal |
| **7** | Group 7 (Radial compensation mode) G40, G41, G42, G41.2, G42.2 command modal |
| **8** | Group 8 (Length compensation mode)<br>G43, G44, G43.1, G43.4, G43.5, G49 command modal |
| **9** | Group 9 (Fixed cycle mode)<br>G70, G71, G72, G73, G74, G75, G76, G77, G78, G79, G80,<br>G81, G82, G83, G84, G85, G86, G87, G88, G89 command modal |
| **10** | Group 10 (Initial point return) G98, (R point return) G99 command modal |
| **11** | Group 11 G50, G51 command modal |
| **12** | Group 12 (Workpiece coordinate system modal)<br>G54, G54.1, G55, G56, G57, G58, G59 command modal |
| **13** | Group 13 (Cutting mode)<br>G61, G61.1, G61.2, G62, G63, G63.1, G63.2, G64 command modal |
| **14** | Group 14 (Modal call) G66, G66.1, G67 command modal |
| **15** | Group 15 (Normal control) G40.1, G41.1, G42.1 command modal<br>(only for M700/M800 series M system) |
| **16** | Group 16 (Coordinate rotation) G68, G68.2, G68.3, G69<br>(only for M700/M800 series M system) |
| **17** | Group 17 (Constant surface speed control) G96, G97 command modal |
| **18** | Group 18 (Polar coordinate command) G15, G16 command modal |
| **19** | Group 19 (G command mirror image) G50.1, G51.1 command modal |
| **20** | Group 20 (Spindle selection) G43.1, G44.1, G47.1 command modal |
| **21** | Group 21 (Cylindrical interpolation / polar coordinate interpolation)<br>G07.1, G107, G12.1, G112, G13.1, G113<br>(only for M700/M800 series M system) |

*pdValue* : Returns the current G code modal command value of the set part system.

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

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZ_ERR_DATA_TYPE**: Invalid argument data type
**EZNC_DATA_READ_ADDR**: Invalid system
**EZNC_DATA_READ_READ**: Data is not readable

| □ **Return value** | Value | Meaning |
|---|---|---|
| | **S_OK** | Normal termination |
| | **S_FALSE** | Communication failure |

□ **Function**

Gets the G code command modal value of the set part system. Refer to the "Programming Manual" of each model for the individual model's G command and group list. Also, depending on the model, different operation than with the original G code command may be incorporated by using the G code macro call. Check after referring to the instructions issued by MTB

□ **Reference**

□ **Specifica-tion**

System

C70 | M700 | M800

## 2.6.2 IEZNcCommand2::GetToolCommand — Get tool compensation number

□ **Custom call procedure**

```
HRESULT     GetToolCommand(
                LONG lAxisNo,                        // (I) Axis No.
                LONG lType,                          // (I) Type
                LONG* plValue,                       // (O) Tool compensation number
                LONG* plRet                          // (O) Error code
                )
```

□ **Automation call procedure**

```
            Command_GetToolCommand(
                lAxisNo As LONG                      // (I) Axis No.
                lType As LONG                        // (I) Type
                plValue As LONG*                     // (O) Tool compensation number
                ) As LONG                            // (O) Error code
```

□ **Argument**

*lAxisNo*: Sets the axis when getting the length compensation number.
(From Axis 1 = from **1**)

*lType*: Sets the tool compensation type to get.

| Value | Meaning |
|---|---|
| **0** | D command value of the shape compensation number |
| **1** | D command value of the wear compensation number |
| **2** | H command value of the length compensation number<br>(axis specification necessary) |

*plValue* : Returns the shape/wear compensation number of the tool for the set part system and the tool length compensation number of the set axis No. in the set part system.
Data range: 1 to 200 (range depends on the number of tool offset sets)
Value meaning: 1 = D1, 1 = H1

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZ_ERR_DATA_TYPE**: Invalid argument data type
**EZNC_DATA_READ_ADDR**: Invalid part system, axis No. setting
**EZNC_DATA_READ_READ**: Data is not readable
**EZ_ERR_NOT_SUPPORT**: Not supported

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets the shape/wear compensation number of the tool for the set part system and the tool length compensation number of the set axis No. in the set part system.

□ **Reference**

□ **Specification**

System | Axis number

**C70** **M700** **M800**

## 2.6.3 IEZNcCommand2::GetFeedCommand Get feed speed command value

□ **Custom call procedure**

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

□ **Automation call procedure**

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

□ **Argument**

*lType*: Sets the command value type to get.

| Value | Meaning |
|---|---|
| **0** | F command feed speed (FA) |
| **1** | Manual effective feed speed (FM) |
| **2** | Synchronization feed speed (FS) |
| **3** | Automatic effective feed speed (FC) |
| **4** | Screw lead (FE) |
| **5** | Tip speed (TCP) (M700/M800 series only) |

*pdValue* : Returns the current feed speed command value of the set part system.

| Feed speed type | Data range | |
|---|---|---|
| | **M700/M800 series** | **C70** |
| FA | 0.000 to 10,000,000.000 [mm/min] | 0.000 to 1,000,000.000 [mm/min] |
| FM | 0.000 to 1,000,000.000 [mm/min] | 0.000 to 1,000,000.000 [mm/min] |
| FS | 0.000 to 1,000,000.000000 [mm/rev] | 0.000 to 1,000,000.000 [mm/rev] |
| FC | 0.000 to 100,000.000 [mm/min] | 0.000 to 1,000,000.000 [mm/min] |
| FE | 0.000 to 100,000.000 [mm/rev]<br>(Unit will vary) | 0.000 to 999.9999 [mm/rev] |

The total number of digits is determined by the NC model, options, and MTB setting values (parameters).

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZ_ERR_DATA_TYPE**: Invalid argument data type
**EZNC_DATA_READ_ADDR**: Invalid system
**EZNC_DATA_READ_READ**: Data is not readable

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets the current feed speed command value of the set part system.

□ **Reference**

□ **Specification**

System

## 2.6.4 IEZNcCommand2::GetCommand2 Get M/S/T/B function command modal value

□ **Custom call procedure**

```
HRESULT     GetCommand2(
                    LONG lType,              // (I) Command type
                    LONG lIndex              // (I) Command number
                    LONG* plValue,           // (O) Command value
                    LONG* plRet              // (O) Error code
                    )
```

□ **Automation call procedure**

```
            Command_GetCommand2(
                    lType As LONG            // (I) Command type
                    lIndex As LONG           // (I) Command number
                    plValue As LONG*         // (O) Command value
                    ) As LONG                // (O) Error code
```

□ **Argument**

*lType*: Sets the command value type to get.

| Value | Meaning |
|---|---|
| **EZNC_M** | M command (sub function M command value) |
| **EZNC_S** | S command (spindle rotation speed S command value) |
| **EZNC_T** | T command (tool change T command value) |
| **EZNC_B** | B command (second sub function command value (specification of index table position, etc.)) |

*lIndex*: Set the command number.
Example) When lType = EZNC_M and lIndex= 1, the M command will be 1.

| Model / Command | C70 | M700 series | M800 series |
|---|---|---|---|
| M | 1 to 4 | 1 to 4 | 1 to 4 |
| S | 1 to 7 | 1 to maximum number of spindles* | 1 to maximum number of spindles* |
| T | 1 to 4 | 1 | 1 |
| B | 1 to 4 | 1 to 4 | 1 to 4 |

*For the maximum number of spindles, refer to the product catalog for each Mitsubishi CNC.

*plValue* : Returns the current command value of the set part system.
Data range: 0 to 99,999999 (maximum)

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZ_ERR_DATA_TYPE**: Invalid argument data type
**EZNC_DATA_READ_READ**: Data is not readable
**EZNC_DATA_READ_ADDR**: Invalid part system setting
**EZNC_DATA_READ_SUBSECT**: Invalid subsection number
**EZ_ERR_NOT_SUPPORT**: Not supported

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets the current command modal value for the M/S/T/B function of the set part system.

□ **Reference**

**SetCommad2( )**

□ **Specification**

System (PLC axis system cannot set)

| C70 | M700 | M800 |
|---|---|---|

## 2.6.5 IEZNcCommand2::SetCommand2 Set manual numerical value command value settings for M/S/T/B functions

□ **Custom call procedure**

```
HRESULT     SetCommand2(
                   LONG lType,                          // (I) Type
                   LONG lIndex                          // (I) Command number
                   LONG lValue,                         // (I) Command value
                   LONG* plRet                          // (O) Error code
                   )
```

□ **Automation call procedure**

```
            Command_SetCommand2(
                   lType As LONG                        // (I) Type
                   lIndex As LONG                       // (I) Command number
                   lValue As LONG                       // (I) Command value
                   ) As LONG                            // (O) Error code
```

□ **Argument**

*lType*: Sets the command value type to get.

| Value | Meaning |
|---|---|
| **EZNC_M** | M command (sub function M command value) |
| **EZNC_S** | S command (spindle rotation speed S command value) |
| **EZNC_T** | T command (tool change T command value) |
| **EZNC_B** | B command (second sub function command value (specification of index table position, etc.)) |

*lIndex*: Sets the set number.
Example) When lType = EZNC_M and lIndex= 1, the M command will be 1.

| Model / Command | C70 | M700 series | M800 series |
|---|---|---|---|
| M | 1 to 4 | 1 to 4 | 1 to 4 |
| S | 1 to 7 | 1 to maximum number of spindles* | 1 to maximum number of spindles* |
| T | 1 to 4 | 1 | 1 |
| B | 1 to 4 | 1 to 4 | 1 to 4 |

*For the maximum number of spindles, refer to the product catalog for each Mitsubishi CNC.

*lValue*: Sets the command value of the set part system or the axis No.
Data range: 0 to 99999999 (maximum)

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZ_ERR_DATA_TYPE**: Invalid argument data type
**EZNC_DATA_WRITE_WRITE**: Data is not writable
**EZNC_DATA_WRITE_ADDR**: Invalid part system setting
**EZNC_DATA_WRITE_SUBSECT**: Invalid subsection number
**EZ_ERR_NOT_SUPPORT**: Not supported

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Sets the manual numerical value command value of the M/S/T/B function of the axis No. or the set part system.

□ **Reference**

**GetCommand2( )**

C70 M700 M800

## 2.7.1 IEZNcProgram2::CurrentBlockRead Read current program block

□ **Custom call procedure**

```
HRESULT      CurrentBlockRead(
                    LONG lBlockNumber,              // (I) Number of blocks
                    LPOLESTR* lppwszProgramData,    // (O) Program storage
                    LONG* plCurrentBlockNo,         // (O) Block number being executed
                    LONG* plRet                     // (O) Error code
                    )
```

□ **Automation call procedure**

```
             Program_CurrentBlockRead(
                    lBlockNumber As LONG            // (I) Number of blocks
                    lppwszProgramData As STRING*    // (O) Program storage
                    plCurrentBlockNo As LONG*       // (O) Block number being executed
                    ) As LONG                       // (O) Error code
```

□ **Argument**

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

*lppwszProgramData*: Gets the program blocks as a **UNICODE** character string. To separate program blocks, **CR, LF** codes are inserted between them. In addition, **NULL** is inserted at the end.

*plCurrentBlockNo*: Returns the block number being executed in the got blocks.

| Value | Meaning |
|---|---|
| **0** | Not in operation |
| **1** | 1st block |
| **2** | 2nd block |

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK** : Normal termination
**EZNC_OPE_GETPRGBLK_ADDR**: Invalid part system setting
**EZNC_OPE_GETPRGBLK_DATAERR**: Error in communication data between NC and personal computer
**EZNC_OPE_GETPRGBLK_DATASIZE**: Application does not fit into prepared buffer
**EZNC_OPE_GETPRGBLK_NOS**: The number of blocks setting is invalid

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets the program for which the operation search has been completed or the program currently being executed. Reads the program for which the operation search has been completed for the set part system or the program block in operation.
If no operation search has been competed, the following applies.

*lppwszProgramData* = **"\0"**
*plCurrentBlockNo* = **0**

As the character string area memory is allocated in this product, clients using VC++ need to release the character string area memory explicitly with **CoTaskMemFree()**.
Even if no operation search has been completed, the character string area memory must be released.

□ **Reference**

**IEZNcOperation**::**Search( )**

□ **Specification**

System

C70 M700 M800

## 2.7.2 IEZNcProgram2::GetProgramNumber2 Get program number

□ **Custom call procedure**

```
HRESULT      GetProgramNumber2(
                    LONG lProgramType,                    // (I) Program type
                    LPOLESTR* lppwszProgramNo,            // (O) Program No.
                    LONG* plRet                           // (O) Error code
                    )
```

□ **Automation call procedure**

```
             Program_GetProgramNumber2(
                    lProgramType As LONG                  // (I) Program type
                    lppwszProgramNo As STRING*            // (O) Program No.
                    ) As LONG                             // (O) Error code
```

□ **Argument**

*lProgramType*: Sets the program type.

| Value | Meaning |
|---|---|
| **EZNC_MAINPRG** | Main program |
| **EZNC_SUBPRG** | Subprogram |

*lppwszProgramNo*: Returns the number of the program for which search has been completed or currently in automatic operation as a **UNICODE** character string. The program number is got as the program file name with the M700/M800 series.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZ_ERR_DATA_TYPE**: Invalid argument data type
**EZNC_DATA_READ_ADDR**: Invalid part system setting
**EZNC_DATA_READ_READ**: Data is not readable

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Returns the number of the program for which search has been completed or currently in automatic operation.
As the character string area memory is allocated in this product, clients using VC++ need to release the character string area memory explicitly with **CoTaskMemFree()**.

□ **Reference**

**GetSequenceNumber(), GetBlockNumber(), GetSubProLevel()**

□ **Specification**

System

## 2.7.3 IEZNcProgram2::GetSequenceNumber Read sequence number

□ **Custom call procedure**

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

□ **Automation call procedure**

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

□ **Argument**

*lProgramType*: Sets the program type.

| Value | Meaning |
|---|---|
| **EZNC_MAINPRG** | Main program |
| **EZNC_SUBPRG** | Subprogram |

*plSequenceNo*: Returns the sequence number of the program for which search has been completed or currently in automatic operation.

*plRet*: Returns an error code. (Upon automation, the return value is used.)

| | |
|---|---|
| **S_OK** | : Normal termination |
| **EZ_ERR_DATA_TYPE** | : Invalid argument data type |
| **EZNC_DATA_READ_ADDR** | : Invalid system specification |
| **EZNC_DATA_READ_READ** | : Data is not readable |

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Returns the sequence number of the program for which search has been completed or currently in automatic operation.

□ **Reference**

**GetProgramNumber2(), GetBlockNumber(), GetSubProLevel()**

□ **Specification**

System

## 2.7.4 IEZNcProgram2::GetBlockNumber Read block number

□ **Custom call procedure**

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

□ **Automation call procedure**

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

□ **Argument**

*lProgramType*: Sets the program type.

| Value | Meaning |
|---|---|
| **EZNC_MAINPRG** | Main program |
| **EZNC_SUBPRG** | Subprogram |

*plBlockNo*: Returns the block number of the program for which search has been completed or currently in automatic operation.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZ_ERR_DATA_TYPE**: Invalid argument data type
**EZNC_DATA_READ_ADDR**: Invalid part system setting
**EZNC_DATA_READ_READ**: Data is not readable

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Returns the block number of the program for which search has been completed or currently in automatic operation.

□ **Reference**

**GetProgramNumber2(), GetSequenceNumber(), GetSubProLevel()**

□ **Specification**

System

## 2.7.5 IEZNcProgram2::GetSubProLevel Get subprogram call level

□ **Custom call procedure**

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

□ **Automation call procedure**

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

□ **Argument**

*plLevel* : Returns the subprogram call level.
Value: **0** to **8**

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_ADDR**: Invalid part system setting
**EZNC_DATA_READ_READ**: Data is not readable

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets the sub-program call level of the set system.

□ **Reference**

**GetProgramNumber2(), GetSequenceNumber()**

□ **Specification**

System

## 2.7.6 IEZNcProgram2::GetInformation Get program information

□ **Custom call procedure**

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

□ **Automation call procedure**

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

□ **Argument**

*lInfoType*: Sets the type of information to get.

| Value | Meaning |
|---|---|
| **EZNC_PRG_MAXNUM** | Maximum number of registrable programs |
| **EZNC_PRG_CURNUM** | Number of programs currently registered |
| **EZNC_PRG_RESTNUM** | Remaining number of registrable programs |
| **EZNC_PRG_CHARNUM** | Number of registered characters |
| **EZNC_PRG_RESTCHARNUM** | Remaining number of registrable characters (250 character units) |

*plInfoData*: Returns the program information specified by *lInfoType*.
If **EZNC_PRG_MAXNUM** set, *lpInfoData*
means 1: 200 [programs]. The data range depends on the specifications of the NC control unit.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZ_ERR_DATA_TYPE**: Invalid argument data type
**EZNC_DATA_READ_READ**: Data is not readable

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets program information.

□ **Reference**

□ **Specifica-tion**

M700 M800

## 2.7.7 IEZNcProgram2:: GetCurrentBlockByByte Get program information

□ **Custom call procedure**

```
HRESULT         GetCurrentBlockByByte(
                        LONG* plSize,                       // (O) Number of bytes
                        LONG* plRet                         // (O) Error code
                        )
```

□ **Automation call procedure**

```
                Program_ GetCurrentBlockByByte(
                        plSize As LONG*                     // (O) Number of bytes
                        ) As LONG                           // (O) Error code
```

□ **Argument**

*plSize* : Returns the number of bytes from start of program of the set part system.
Value: From **0**

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_ADDR**: Invalid part system setting
**EZNC_DATA_READ_READ**: Data is not readable
**EZ_ERR_NOT_SUPPORT**: Not supported

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets the number of bytes from the start of the program in the searched block or the block currently stopped with single block stop.
This is valid only for the M700/M800 series.
This function is not supported with C70.

□ **Reference**

□ **Specification**

**C70** **M700** **M800**

## 2.8.1 IEZNcTime::GetClockData Get date and time

□ **Custom call procedure**

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

□ **Automation call procedure**

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

□ **Argument**

*plDate*: Returns the date (year, month, day).
Output example: 1998/12/05 = 19981205

*plTime*: Returns the time (hour, minute, second) of the clock in the NC.
Value: 0 to 235959
Output example: 23:59:59 = 235959

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_READ**: Data is not readable

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets the date and time from the clock in the NC.

□ **Reference**

**SetClockData()**

□ **Specification**

## 2.8.2 IEZNcTime::SetClockData Set date and time

□ **Custom call procedure**

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

□ **Automation call procedure**

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

| | | |
|---|---|---|
| □ **Argument** | *lDate*: Sets the date (year, month, day).<br>Setting example: 1998/12/05 = 19981205<br><br>*lTime:* Sets the time (hour, minute, second) for the clock in the NC.<br>Value: 0 to 235959<br>Setting example: 23:59:59 = 235959<br><br>*plRet*: Returns an error code. (Upon automation, the return value is used.)<br>**S_OK**: Normal termination<br>**EZNC_DATA_WRITE_WRITE**: Data is not writable | |
| □ **Return value** | Value | Meaning |
| | **S_OK** | Normal termination |
| | **S_FALSE** | Communication failure |
| □ **Function** | Sets the date and time for the clock in the NC. | |
| □ **Reference** | **GetClockData()** | |
| □ **Specifica-tion** | | |

## 2.8.3 IEZNcTime::GetAliveTime Get power-on time

□ **Custom call procedure**

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

□ **Automation call procedure**

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

□ **Argument**

*plTime*: Gets total power-on time (hour, minute, second) from the controller power ON to OFF.
Value: 0 to 99995959
Output example: 9999:59:59 = 99995959

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_READ**: Data is not readable

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets total power-on time (hour, minute, second) from the controller power ON to OFF.
Stops integration when the value reaches the maximum value, and retains the maximum value.

□ **Reference**

**SetAliveTime()**

□ **Specification**

## 2.8.4 IEZNcTime::SetAliveTime Set power-on time

□ **Custom call procedure**

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

□ **Automation call procedure**

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

| | |
|---|---|
| □ **Argument** | *lTime*: Sets total power-on time (hour, minute, second) from the controller power ON to OFF.<br>Value: 0 to 99995959<br>Setting example: 9999:59:59 = 99995959<br><br>*plRet*: Returns an error code. (Upon automation, the return value is used.)<br>**S_OK**: Normal termination<br>**EZNC_DATA_WRITE_WRITE**: Data is not writable |

| □ **Return value** | Value | Meaning |
|---|---|---|
| | **S_OK** | Normal termination |
| | **S_FALSE** | Communication failure |

| | |
|---|---|
| □ **Function** | Forcibly sets total power-on time (hour, minute, second) from the controller power ON to OFF. |
| □ **Reference** | **GetAliveTime()** |
| □ **Specification** | |

## 2.8.5 IEZNcTime::GetRunTime Get automatic operation time

**□ Custom call procedure**

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

**□ Automation call procedure**

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

| | |
|---|---|
| □ **Argument** | *plTime*: Returns total processing time (hour, minute, second) from the automatic operation start using memory (tape) or in MDI mode to the termination by M02/M30 or the reset operation.<br>Value: 0 to 99995959<br>Output example: 9999:59:59 = 99995959<br><br>*plRet*: Returns an error code. (Upon automation, the return value is used.)<br>**S_OK**: Normal termination<br>**EZNC_DATA_READ_READ**: Data is not readable |

| □ **Return value** | Value | Meaning |
|---|---|---|
| | **S_OK** | Normal termination |
| | **S_FALSE** | Communication failure |

| | |
|---|---|
| □ **Function** | Gets total processing time (hour, minute, second) from the automatic operation start using memory (tape) or in MDI mode to the termination by M02/M30 or the reset operation.<br>Stops integration when the value reaches the maximum value, and retains the maximum value. |
| □ **Reference** | **SetRunTime()** |
| □ **Specifica-tion** | |

## 2.8.6 IEZNcTime::SetRunTime Set automatic operation time

**□ Custom call procedure**

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

**□ Automation call procedure**

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

**□ Argument**

*lTime*: Sets total processing time (hour, minute, second) from the automatic operation start using memory (tape) or in MDI mode to the termination by M02/M30 or the reset operation.
Value: 0 to 99995959
Setting example: 9999:59:59 = 99995959

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_WRITE_WRITE**: Data is not writable

**□ Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

**□ Function**

Forcibly sets total processing time (hour, minute, second) from the automatic operation start using memory (tape) or in MDI mode to the termination by M02/M30 or the reset operation.

**□ Reference**

**GetRunTime()**

**□ Specification**

## 2.8.7 IEZNcTime::GetStartTime Get automatic start time

□ **Custom call procedure**

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

□ **Automation call procedure**

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

□ **Argument**

*plTime*: Returns total automatic operation time (hour, minute, second) from the automatic operation start using memory (tape) or in MDI mode to the termination by feed hold, block stop, or reset.
Value: 0 to 99995959
Output example: 9999:59:59 = 99995959

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_READ**: Data is not readable

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets total automatic operation time (hour, minute, second) from the automatic operation start using memory (tape) or in MDI mode to the termination by feed hold, block stop, or reset.

□ **Reference**

**SetStartTime()**

□ **Specification**

## 2.8.8 IEZNcTime::SetStartTime Set automatic start time

**□ Custom call procedure**

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

**□ Automation call procedure**

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

**□ Argument**

*lTime*: Setes total automatic operation time (hour, minute, second) from the automatic operation start using memory (tape) or in MDI mode to the termination by feed hold, block stop, or reset.
Value: 0 to 99995959
Setting example: 9999:59:59 = 99995959

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_WRITE_WRITE**: Data is not writable

**□ Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

**□ Function**

Sets total automatic operation time (hour, minute, second) from the automatic operation start using memory (tape) or in MDI mode to the termination by feed hold, block stop, or reset.

**□ Reference**

**GetStartTime()**

**□ Specification**

## 2.8.9 IEZNcTime::GetEstimateTime Get external integration time

□ **Custom call procedure**

```
HRESULT    GetEstimateTime(
                LONG lKind,          // (I) External integration time type
                LONG* plTime,        // (O) External integration time
                LONG* plRet          // (O) Error code
                )
```

□ **Automation call procedure**

```
           Time_GetEstimateTime(
                lKind As LONG        // (I) External integration time type
                plTime As LONG*      // (O) External integration time
                ) As LONG            // (O) Error code
```

□ **Argument**

*lKind*: Sets the external integration time type.

| Value | Meaning |
|---|---|
| 0 | External integration time 1 (programmable controller device C70: Y314 M700/M800 series: Y704): When counting with the device turned ON |
| 1 | External integration time 2 (programmable controller device C70: Y315 M700/M800 series: Y705): When counting with the device turned ON |

*plTime*: Returns the time (hour, minute, second) controlled by the programmable controller. Stops integration when the integration time display reaches the maximum value, and retains the display with the maximum value.
Value: 0 to 99995959
Output example: 9999:59:59 = 99995959

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_READ**: Data is not readable

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Returns the time (hour, minute, second) controlled by the programmable controller. Starts counting when the user programmable controller device is turned ON. Refer to the programmable controller interface manual for each model because the device number differs depending on models.

□ **Reference**

**SetEstimateTime()**

□ **Specifica-tion**

## 2.8.10 IEZNcTime::SetEstimateTime Set external integration time

**□ Custom call procedure**

```
HRESULT      SetEstimateTime(
                      LONG lKind,               // (I) External integration time type
                      LONG lTime,               // (I) External integration time
                      LONG* plRet               // (O) Error code
                      )
```

**□ Automation call procedure**

```
             Time_SetEstimateTime(
                      lKind As LONG             // (I) External integration time type
                      lTime As LONG             // (I) External integration time
                      ) As LONG                 // (O) Error code
```

**□ Argument**

*lKind*: Sets the external integration time type.

| Value | Meaning |
|---|---|
| 0 | External integration time 1 (programmable controller device C70: Y314 M700/M800 series: Y704): When counting with the device turned ON |
| 1 | External integration time 2 (programmable controller device C70: Y315 M700/M800 series: Y705): When counting with the device turned ON |

*lTime*: Sets the time (hour, minute, second) controlled by the PLC.
Stops integration when the integration time display reaches the maximum value, and retains the display with the maximum value.
Value: 0 to 99995959
Setting example: 9999:59:59 = 99995959

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_WRITE_WRITE**: Data is not writable

**□ Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

**□ Function**

Sets the time (hour, minute, second) controlled by the programmable controller. Starts counting when the user programmable controller device is turned ON. Refer to the programmable controller interface manual for each model because the device number differs depending on models.

**□ Reference**

**GetEstimateTime()**

**□ Specification**

C70 | M700 | M800

## 2.9.1 IEZNcAxisMonitor::GetServoMonitor Get servo monitor

**□ Custom call procedure**

```
HRESULT      GetServoMonitor(
                      LONG lAxisNo,                      // (I) Axis No.
                      LONG lIndex,                       // (I) Monitor data
                      LONG* plData,                      // (O) Monitor data
                      LPOLESTR* lppwszBuffer,            // (O) Monitor data character string
                      LONG* plRet                        // (O) Error code
                      )
```

**□ Automation call procedure**

```
             Monitor_GetServoMonitor(
                      lAxisNo As LONG                    // (I) Axis No.
                      lIndex As LONG                     // (I) Monitor data
                      plData As LONG*                    // (O) Monitor data
                      lppwszBuffer As STRING*            // (O) Monitor data character string
                      ) As LONG                          // (O) Error code
```

**□ Argument**

*lAxisNo*: Sets the axis No. (From Axis 1 = from **1**)

*lIndex*: Sets the parameter number for the set axis No. in the set system.

*plData*: Returns the axis status.

*lppwszBuffer*: Outputs data (return value) as a **UNICODE** character string when any of 100 to 104 is set for *lIndex*.
For the **M700/M800 series**, outputs data (return value) as a **UNICODE** character string when any of 11 to 15, 18 to 20, or 100 to 104 is set for *lIndex*.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_READ**: Data is not readable
**EZNC_DATA_READ_DATASIZE**: Application does not fit into prepared buffer
**EZNC_DATA_READ_DATATYPE: Invalid data type (parameter number)**
**EZNC_DATA_READ_ADDR**: Invalid part system, axis No. setting
**EZNC_DATA_READ_AXIS**: Invalid axis specification
**EZ_ERR_DATA_TYPE**: Invalid argument data type

| *lIndex* | Description | Data range | Remarks |
|---|---|---|---|
| 0 | GAIN. Position loop gain status display. | Unit: 1/s | |
| 1 | DROOP. (tracking delay) | Unit: i | |
| 2 | SPEED. Actual motor speed. | From 0 [rpm] | |
| 3 | CURRENT. Load current. Motor current (displayed by converting to continuous current when stalled). | From 0 [%] | |
| 4 | MAXCUR1. Maximum current I. | Unit: % | |
| 5 | MAXCUR2. Maximum current II. | Unit: % | |
| 6 | OVER LOAD. Overload. | Unit: % | |
| 7 | REGEN LOAD. Regenerative load. | Unit: % | |
| 10 | CYC CNT. Cycle counter. | Unit: Pulse | |

| □ **Argument** | *IIndex* | Description | Data range | Remarks |
|---|---|---|---|---|
| | 11 | GRIDSP. Grid interval. | Unit: Command unit* | |
| | 12 | GRID. Grid amount. | Unit: Command unit* | |
| | 13 | MACPOS. Machine position. | Unit: Command unit* | |
| | 14 | MOT POS. Motor end FB. | Unit: Command unit* | |
| | 15 | SCA POS. Machine end FB. | Unit: Command unit* | |
| | 16 | FB ERROR. FB error. | Unit: i | |
| | 17 | DFB COMP. DFB compensation amount. | | |
| | 18 | Remain command | Unit: Command unit* | |
| | 19 | Currnt posn. | Unit: Command unit* | |
| | 20 | Manual interrupt amount. | Unit: Command unit* | |
| | 100 | AMP DISP. Amplifier display. 7-segment LED display on a drive unite. | Outputs a 3-digit character string from "00\0" to "FF\0". | |
| | 101 | Alarm 1. | Outputs a 3-digit character string. | |
| | 102 | Alarm 2. | Same as above | |
| | 103 | Alarm 3. | Same as above | |
| | 104 | Alarm 4. | Same as above | |

* The M700/M800 series allows acquisition of value (actual value) converted according to the command unit as a character string.

| □ **Return value** | Value | Meaning |
|---|---|---|
| | **S_OK** | Normal termination |
| | **S_FALSE** | Communication failure |

□ **Function**

Gets servo monitor information of the set axis No. in the set part system.

When the data range in the *Index* table is [Unit: Command unit], the value got needs to be converted according to the command unit sets for the Mitsubishi CNC. For the setting unit, refer to the Mitsubishi CNC specifications.

< For Linear axis>

| | Metric system | Conversion for 1 of LONG type |
|---|---|---|
| For 10-µm specifications | -999,999.99 to 999,999.99 | 1 = 1/200 (mm) |
| For 1-µm specifications | -99,999.999 to 99,999.999 | 1 = 1/2000 (mm) |
| For 0.1-µm specifications | -9,999.9999 to 9,999.9999 | 1 = 1/20000 (mm) |
| For 10-nm specifications | -999.99999 to 999.99999 | 1 = 1/200000 (mm) |
| For 1-nm specifications | -99.999999 to 99.999999 | 1 = 1/2000000 (mm) |

| | Inch system | Conversion for 1 of LONG type |
|---|---|---|
| For 10-µm specifications | -99,999.999 to 99,999.999 | 1 = 1/200 (inch) |
| For 1-µm specifications | -9,999.9999 to 9,999.9999 | 1 = 1/2000 (inch) |
| For 0.1-µm specifications | -999.99999 to 999.99999 | 1 = 1/20000 (inch) |
| For 10-nm specifications | -999.99999 to 999.99999 | 1 = 1/200000 (inch) |
| For 1-nm specifications | -99.999999 to 99.999999 | 1 = 1/2000000 (inch) |

< For Rotary axis >

| | Metric system | Conversion for 1 of LONG type |
|---|---|---|
| For 10-µm specifications | -999999.99 to 999999.99 | 1 = 1/200 (mm) |
| For 1-µm specifications | -99,999.999 to 99,999.999 | 1 = 1/2000 (mm) |
| For 0.1-µm specifications | -9999.9999 to 9999.9999 | 1 = 1/20000 (mm) |
| For 10-nm specifications | -999.99999 to 999.99999 | 1 = 1/200000 (mm) |
| For 1-nm specifications | -99.999999 to 99.999999 | 1 = 1/2000000 (mm) |

| | Inch system | Conversion for 1 of LONG type |
|---|---|---|
| For 10-µm specifications | -999,999.99 to 999,999.99 | 1 = 1/200 (inch) |
| For 1-µm specifications | -99,999.999 to 99,999.999 | 1 = 1/2000 (inch) |
| For 0.1-µm specifications | -9999.9999 to 9999.9999 | 1 = 1/20000 (inch) |
| For 10-nm specifications | -999.99999 to 999.99999 | 1 = 1/200000 (inch) |
| For 1-nm specifications | -99.999999 to 99.999999 | 1 = 1/2000000 (inch) |

Conversion example) For the linear axis with the 1-µm specifications in the Metric system, when the LONG value got is 710001,

710001 ÷ 2000 = 355.0005. However, 355.0005 is rounded to minus infinity. Therefore the result is 355.000.

Data in units of 0.5 µm (1/2000 mm) is rounded to minus infinity when displayed.
This means that +0.5 µm is displayed as 0 and -0.5 µm is displayed as -1 µm.

As the string area memory is allocated in this product, the client using VC++ needs to release the string area memory explicitly with **CoTaskMemFree()**.

□ **Reference**

**GetServoVersion(), GetServoDiagnosis()**

□ **Specification**

System , PLC axis , Axis number

## 2.9.2 IEZNcAxisMonitor::GetServoVersion Get servo axis unit information

□ **Custom call procedure**

```
HRESULT        GetServoVersion(
                        LONG lAxisNo,                      // (I) Axis No.
                        LONG lIndex,                       // (I) Servo information
                        LPOLESTR* lppwszBuffer,            // (O) Servo information
                        LONG* plRet                        // (O) Error code
                        )
```

□ **Automation call procedure**

```
               Monitor_GetServoVersion(
                        lAxisNo As LONG                    // (I) Axis No.
                        lIndex As LONG                     // (I) Servo information
                        lppwszBuffer As STRING*            // (O) Servo information
                        ) As LONG                          // (O) Error code
```

□ **Argument**

*lAxisNo*: Sets the axis No. (From Axis 1 = from **1**)

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

*lppwszBuffer*: Sets servo information as a **UNICODE** character string.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_READ**: Data is not readable
**EZNC_DATA_READ_ADDR**: Invalid part system, axis No. setting
**EZNC_DATA_READ_AXIS**: Invalid axis No. setting

| *lIndex* | Description | Data range |
|---|---|---|
| 0 | Unit type | Up to 17 alphanumeric characters. |
| 1 | Unit serial No. | Up to 9 alphanumeric characters. |
| 2 | Software version | Up to 17 alphanumeric characters. |
| 3 | Control method. | Up to 7 alphanumeric characters. |
| 4 | Motor end detector | Up to 9 alphanumeric characters. |
| 5 | Machine end detector | Up to 9 alphanumeric characters. |
| 6 | Motor | Up to 9 alphanumeric characters. |

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets servo version information.
As the string area memory is allocated in this product, the client using VC++ needs to release the string area memory explicitly with **CoTaskMemFree()**.

□ **Reference**

**GetServoMonitor(), GetServoDiagnosis()**

□ **Specification**

System , PLC axis , Axis number

C70 M700 M800

## 2.9.3 IEZNcAxisMonitor::GetServoDiagnosis Get servo diagnostics information

□ **Custom call procedure**

```
HRESULT      GetServoDiagnosis(
                      LONG lAxisNo,                 // (I) Axis No.
                      LONG lIndex,                  // (I) Diagnostics information
                      LONG* plData,                 // (O) Diagnostics information value
                      LPOLESTR* lppwszBuffer,       // (O) Diagnostics information character string
                      LONG* plRet                   // (O) Error code
                      )
```

□ **Automation call procedure**

```
             Monitor_GetServoDiagnosis(
                      lAxisNo As LONG               // (I) Axis No.
                      lIndex As LONG                // (I) Diagnostics information
                      plData As LONG*               // (O) Diagnostics information value
                      lppwszBuffer As STRING*       // (O) Diagnostics information character string
                      ) As LONG                     // (O) Error code
```

□ **Argument**

*lAxisNo*: Sets the axis No. (From Axis 1 = from **1**)

*lIndex*: Sets servo diagnostics information. Refer to the table below.

*plData*: Returns the servo diagnostics information value.

*lppwszBuffer*: Gets diagnostics information as a **UNICODE** character string. Outputs a character string when *lIndex* is 21 to 30.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_READ**: Data is not readable
**EZNC_DATA_READ_ADDR**: Invalid system, spindle specification
**EZNC_DATA_READ_AXIS**: Invalid axis specification

| *lIndex* | Description | Data range |
|---|---|---|
| 0 | Work time | |
| 1 | Alarm history 1 (Alarm No). | Previous servo alarm number |
| 2 | ″ 2 (Alarm No). | Same as above |
| 3 | ″ 3 (Alarm No). | Same as above |
| 4 | ″ 4 (Alarm No). | Same as above |
| 5 | ″ 5 (Alarm No). | Same as above |
| 6 | ″ 6 (Alarm No). | Same as above |
| 7 | ″ 7 (Alarm No). | Same as above |
| 8 | ″ 8 (Alarm No). | Same as above |
| 11 | Alarm history 1 (time). | Previous servo alarm occurrence time |
| 12 | ″ 2 (time). | Same as above |
| 13 | ″ 3 (time). | Same as above |
| 14 | ″ 4 (time). | Same as above |
| 15 | ″ 5 (time). | Same as above |
| 16 | ″ 6 (time). | Same as above |
| 17 | ″ 7 (time). | Same as above |
| 18 | ″ 8 (time). | Same as above |
| 21 | MNT (1). | Up to 3 alphanumeric characters |
| 22 | MNT (2). | Same as above |
| 23 | MNT (3). | Same as above |
| 24 | MNT (4). | Same as above |
| 30 | SYS. | Up to 2 alphanumeric characters |

| | | |
|---|---|---|
| □ **Return value** | Value | Meaning |
| | **S_OK**<br>**S_FALSE** | Normal termination<br>Communication failure |
| □ **Function** | Gets servo diagnostics information.<br>As the string area memory is allocated in this product, the client using VC++ needs to release the string area memory explicitly with **CoTaskMemFree()**. | |
| □ **Reference** | **GetServoMonitor(), GetServoVersion()** | |
| □ **Specifica-tion** | System , PLC axis , Axis number | |

C70 | M700 | M800

## 2.9.4 IEZNcAxisMonitor::GetPowerVersion — Get power supply version information

**□ Custom call procedure**

```
HRESULT        GetPowerVersion(
                        LONG lAxisNo,                    // (I) Axis No.
                        LONG lIndex,                     // (I) Version information
                        LPOLESTR* lppwszBuffer,          //(O)Version information character string
                        LONG* plRet                      // (O) Error code
                        )
```

**□ Automation call procedure**

```
               Monitor_GetPowerVersion(
                        lAxisNo As LONG                  // (I) Axis No.
                        lIndex As LONG                   // (I) Version information
                        lppwszBuffer As STRING*          // (O) Version information
                        ) As LONG                        // (O) Error code
```

**□ Argument**

*lAxisNo*: Sets the axis No. (From Axis 1 = from **1**)

*lIndex*: Sets version information. Refer to the table below.

*lppwszBuffer*: Gets version information as a **UNICODE** character string.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_READ**: Data is not readable
**EZNC_DATA_READ_ADDR**: Invalid system, spindle specification

| *lIndex* | Description | Data range |
|---|---|---|
| 0 | Unit type | Up to 17 alphanumeric characters. |
| 1 | Unit serial No. | Up to 9 alphanumeric characters. |
| 2 | Software version | Up to 17 alphanumeric characters. |
| 3 | Connected drive. | 1 alphanumeric character. |

**□ Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

**□ Function**

Gets power supply version information.
As the string area memory is allocated in this product, the client using VC++ needs to release the string area memory explicitly with **CoTaskMemFree()**.

**□ Reference**

**GetPowerDiagnosis()**

**□ Specification**

Axis number

## 2.9.5 IEZNcAxisMonitor::GetPowerDiagnosis Get power supply diagnostics information

□ **Custom call procedure**

```
HRESULT     GetPowerDiagnosis(
                     LONG lAxisNo,                 // (I) Axis No.
                     LONG lIndex,                  // (I) Diagnostics information
                     LONG* plData,                 // (O) Diagnostics information value
                     LPOLESTR* lppwszBuffer,       // (O) Diagnostics information character string
                     LONG* plRet                   // (O) Error code
                     )
```

□ **Automation call procedure**

```
            Monitor_GetPowerDiagnosis(
                     lAxisNo As LONG               // (I) Axis No.
                     lIndex As LONG                // (I) Diagnostics information
                     plData As LONG*               // (O) Diagnostics information value
                     lppwszBuffer As STRING*       // (O) Diagnostics information character string
                     ) As LONG                     // (O) Error code
```

□ **Argument**

*lAxisNo*: Sets the axis No. (From Axis 1 = from **1**)

*lIndex*: Sets diagnostics information. Refer to the table below.

*plData*: Returns the diagnostics information value.

*lppwszBuffer*: Gets diagnostics information as a **UNICODE** character string.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_READ**: Data is not readable
**EZNC_DATA_READ_ADDR**: Invalid part system, axis No. setting

| *lIndex* | Description | Data range |
|---|---|---|
| 0 | Work time | |
| 1 | Alarm history 1 (Alarm No). | Previous servo alarm number |
| 2 | 〃 2 (Alarm No). | Same as above |
| 3 | 〃 3 (Alarm No). | Same as above |
| 4 | 〃 4 (Alarm No). | Same as above |
| 5 | 〃 5 (Alarm No). | Same as above |
| 6 | 〃 6 (Alarm No). | Same as above |
| 7 | 〃 7 (Alarm No). | Same as above |
| 8 | 〃 8 (Alarm No). | Same as above |
| 11 | Alarm history 1 (time). | Previous servo alarm occurrence time |
| 12 | 〃 2 (time). | Same as above |
| 13 | 〃 3 (time). | Same as above |
| 14 | 〃 4 (time). | Same as above |
| 15 | 〃 5 (time). | Same as above |
| 16 | 〃 6 (time). | Same as above |
| 17 | 〃 7 (time). | Same as above |
| 18 | 〃 8 (time). | Same as above |
| 21 | MNT (1). | Up to 3 alphanumeric characters |
| 22 | MNT (2). | Same as above |
| 23 | MNT (3). | Same as above |
| 24 | MNT (4). | Same as above |
| 30 | SYS | Up to 2 alphanumeric characters |

| | | |
|---|---|---|
| □ **Return value** | Value | Meaning |
| | **S_OK** | Normal termination |
| | **S_FALSE** | Communication failure |
| □ **Function** | Gets power supply diagnostics information.<br>As the string area memory is allocated in this product, the client using VC++ needs to release the string area memory explicitly with **CoTaskMemFree()**. | |
| □ **Reference** | **GetPowerVersion()** | |
| □ **Specification** | Axis number | |

## 2.9.6 IEZNcAxisMonitor::GetSpindleMonitor Monitor spindle

□ **Custom call procedure**

```
HRESULT     GetSpindleMonitor(
                    LONG lIndex,                  // (I) Monitor data
                    LONG lSpindle,                // (I) Spindle number
                    LONG* plData,                 // (O) Monitor data value
                    LPOLESTR* lppwszBuffer,       // (O) Monitor data character string
                    LONG* plRet                   // (O) Error code
                    )
```

□ **Automation call procedure**

```
            Monitor_GetSpindleMonitor(
                    lIndex As LONG                // (I) Monitor data
                    lSpindle As LONG              // (I) Spindle number
                    plData As LONG*               // (O) Monitor data value
                    lppwszBuffer As STRING*       // (O) Monitor data character string
                    ) As LONG                     // (O) Error code
```

□ **Argument**

*lIndex:* Sets the parameter number for the set spindle.

*lSpindle*: Set the spindle number.

*plData*: Returns the spindle status.

*lppwszBuffer*: Gets spindle information as a **UNICODE** character string.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_READ**: Data is not readable
**EZNC_DATA_READ_ADDR**: Invalid system, spindle specification
**EZ_ERR_DATA_TYPE**: Invalid argument data type

| *lIndex* | Description | Data range | Remarks |
|---|---|---|---|
| 0 | Gain. Spindle position loop gain. | Unit: 1/s | |
| 1 | Droop. Position deviation amount. | Unit: I | |
| 2 | Spindle (SR, SF) rotation speed. Actual spindle motor speed. Including override. | From 0 [rpm] | |
| 3 | Load. Spindle motor load. | From 0 [%] | |
| 4 | LED display.<br>7-segment LED display on a driver. | Outputs a 3-digit character string from "00\0" to "FF\0". | |
| 5 | Alarm 1. | Up to 3 alphanumeric characters. | |
| 6 | Alarm 2. | Same as above | |
| 7 | Alarm 3. | Same as above | |
| 8 | Alarm 4. | Same as above | M700/M800 series only |
| 10 | Cycle counter. | | |
| 11 | Control input 1. | | |
| 12 | 〃 2. | | |
| 13 | 〃 3. | | |
| 14 | 〃 4. | | |
| 15 | Control output 1. | | |
| 16 | 〃 2. | | |
| 17 | 〃 3. | | |
| 18 | 〃 4. | | |

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

| | |
|---|---|
| □<br>**Function** | Gets the set spindle status. |
| □<br>**Reference** | **GetSpindleVersion(), GetSpindleDiagnosis()** |
| □<br>**Specifica-tion** | |

## 2.9.7 IEZNcAxisMonitor::GetSpindleVersion Get spindle unit version information

□ **Custom call procedure**

```
HRESULT     GetSpindleVersion(
                 LONG lAxisNo,                // (I) Axis No.
                 LONG lIndex,                 // (I) Version information
                 LPOLESTR* lppwszBuffer,      // (O) Version information character string
                 LONG* plRet                  // (O) Error code
                 )
```

□ **Automation call procedure**

```
            Monitor_GetSpindleVersion(
                 lAxisNo As LONG              // (I) Axis No.
                 lIndex As LONG               // (I) Version information
                 lppwszBuffer As STRING*      // (O) Version information
                 ) As LONG                    // (O) Error code
```

□ **Argument**

*lAxisNo*: Sets the axis No. (From Axis 1 = from **1**)

*lIndex*: Sets version information. Refer to the table below.

*lppwszBuffer*: Gets version information as a **UNICODE** character string.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_READ**: Data is not readable
**EZNC_DATA_READ_ADDR**: Invalid part system, spindle setting
**EZNC_DATA_READ_AXIS**: Invalid axis No. setting

| *lIndex* | Description | Data range |
|---|---|---|
| 0 | Unit type | Up to 17 alphanumeric characters. |
| 1 | Unit serial No. | Up to 9 alphanumeric characters. |
| 2 | Software version | Up to 17 alphanumeric characters. |

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets spindle version information.
As the string area memory is allocated in this product, the client using VC++ needs to release the string area memory explicitly with **CoTaskMemFree()**.

□ **Reference**

**GetSpindleMonitor(), GetSpindleDiagnosis()**

□ **Specification**

Axis number

C70 M700 M800

## 2.9.8 IEZNcAxisMonitor::GetSpindleDiagnosis — Get spindle diagnostics information

□ **Custom call procedure**

```
HRESULT        GetSpindleDiagnosis(
                        LONG lAxisNo,                  // (I) Axis No.
                        LONG lIndex,                   // (I) Diagnostics information
                        LONG* plData,                  // (O) Diagnostics information value
                        LPOLESTR* lppwszBuffer,        // (O) Diagnostics information character string
                        LONG* plRet                    // (O) Error code
                        )
```

□ **Automation call procedure**

```
               Monitor_GetSpindleDiagnosis(
                        lAxisNo As LONG                // (I) Axis No.
                        lIndex As LONG                 // (I) Diagnostics information
                        plData As LONG*                // (O) Diagnostics information value
                        lppwszBuffer As STRING*        // (O) Diagnostics information character string
                        ) As LONG                      // (O) Error code
```

□ **Argument**

*lAxisNo*: Sets the axis No. (From Axis 1 = from **1**)

*lIndex*: Sets diagnostics information.

*plData*: Returns the diagnostics information value.

*lppwszBuffer*: Gets diagnostics information as a **UNICODE** character string.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_READ**: Data is not readable
**EZNC_DATA_READ_ADDR**: Invalid part system, axins No. setting
**EZNC_DATA_READ_AXIS**: Invalid axis No. setting
**EZ_ERR_DATA_TYPE**: Invalid argument data type

| *lIndex* | Description | Data range |
|---|---|---|
| 0 | Work time | |
| 1 | Alarm history 1 (Alarm No). | Previous spindle alarm number |
| 2 | ″ 2 (Alarm No). | Same as above |
| 3 | ″ 3 (Alarm No). | Same as above |
| 4 | ″ 4 (Alarm No). | Same as above |
| 5 | ″ 5 (Alarm No). | Same as above |
| 6 | ″ 6 (Alarm No). | Same as above |
| 7 | ″ 7 (Alarm No). | Same as above |
| 8 | ″ 8 (Alarm No). | Same as above |
| 11 | Alarm history 1 (time). | Previous spindle alarm time |
| 12 | ″ 2 (time). | Same as above |
| 13 | ″ 3 (time). | Same as above |
| 14 | ″ 4 (time). | Same as above |
| 15 | ″ 5 (time). | Same as above |
| 16 | ″ 6 (time). | Same as above |
| 17 | ″ 7 (time). | Same as above |
| 18 | ″ 8 (time). | Same as above |
| 21 | MNT (1). | Up to 3 alphanumeric characters. |
| 22 | MNT (2). | Same as above |
| 23 | MNT (3). | Same as above |
| 24 | MNT (4). | Same as above |
| 30 | SYS. | Up to 2 alphanumeric characters. |

| | | |
|---|---|---|
| □ **Return value** | Value | Meaning |
| | **S_OK**<br>**S_FALSE** | Normal termination<br>Communication failure |
| □ **Function** | Gets power supply diagnostics information.<br>As the string area memory is allocated in this product, the client using VC++ needs to release the string area memory explicitly with **CoTaskMemFree()**. | |
| □ **Reference** | **GetSpindleMonitor(), GetSpindleVersion()** | |
| □ **Specification** | Axis number | |

## 2.9.9 IEZNcAxisMonitor::GetAbsPositionMonitor Get absolute position monitor information

**□ Custom call procedure**

```
HRESULT     GetAbsPositionMonitor(
                  LONG lAxisNo,                // (I) Axis No.
                  LONG lIndex,                 // (I)Axis monitor information
                  LONG* plData,                // (O) Monitor information value
                  LPOLESTR* lppwszBuffer,      // (O) Absolute position monitor information
                                                    character string
                  LONG* plRet                  // (O) Error code
                  )
```

**□ Automation call procedure**

```
            Monitor_GetAbsPositionMonitor(
                  lAxisNo As LONG              // (I) Axis No.
                  lIndex As LONG               // (I) Monitor information
                  plData As LONG*              // (O) Monitor information value
                  lppwszBuffer As STRING*      // (O) Absolute position monitor information
                                                     character string
                  ) As LONG                    // (O) Error code
```

**□ Argument**

*lAxisNo*: Sets the axis No. (From Axis 1 = from **1**)

*lIndex*: Sets monitor information.

*plData*: Returns monitor information value.

*lppwszBuffer*: Gets information about the absolute position monitor as a **UNICODE** character string. Outputs a character string when *lIndex* is 0.
For the M700/M800 series, outputs the result as a **UNICODE** character string when *lIndex* is 0 to 3.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_READ**: Data is not readable
**EZNC_DATA_READ_ADDR**: Invalid part system, axis No. setting
**EZNC_DATA_READ_AXIS**: Invalid axis No. setting
**EZ_ERR_DATA_TYPE**: Invalid argument data type

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

* The M700/M800 series allows acquisition of value (actual value) converted according to the command unit as a character string. For conversion according to the command unit, refer to □ Function in IEZNcAxisMonitor::GetServoMonitor().

**□ Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

**□ Function**

Gets information about the absolute position monitor.
ABS0 is valid only for the M700/M800 series.
As the string area memory is allocated in this product, the client using VC++ needs to release the string area memory explicitly with **CoTaskMemFree()**.

**□ Reference**

**□ Specification**

System , PLC axis , Axis number

## 2.9.10 IEZNcAxisMonitor::GetAuxAxisMonitor Get auxiliary axis monitor information

**□ Custom call procedure**

```
HRESULT      GetAuxAxisMonitor (
                     LONG lAxisNo,                  // (I) Axis No.
                     LONG lIndex,                   // (I) Auxiliary axis information type
                     LONG* plData,                  // (O) Auxiliary axis information value
                     LPOLESTR* lppwszBuffer,        // (O) Auxiliary axis monitor information
                                                          character string
                     LONG* plRet                    // (O) Error code
                     )
```

**□ Automation call procedure**

```
             Monitor_GetAuxAxisMonitor (
                     lAxisNo As LONG                // (I) Axis No.
                     lIndex As LONG                 // (I) Auxiliary axis information type
                     plData As LONG*                // (O) Auxiliary axis information value
                     lppwszBuffer As STRING*        // (O) Auxiliary axis monitor information string
                     ) As LONG                      // (O) Error code
```

□ **Argument**

*lAxisNo*: Sets the axis No. (From Axis 1 = from **1**)

*lIndex*: Sets the auxiliary axis information type. Refer to the table below.

*plData:* Returns the auxiliary axis information.

*lppwszBuffer:* Gets monitor information as a **UNICODE** string.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_READ**: Data is not readable
**EZNC_DATA_READ_ADDR**: Invalid part system, axis No. setting
**EZ_ERR_NOT_SUPPORT**: Not supported

| *lIndex* | Description | Data range |
|---|---|---|
| 0 | Droop. | -999 to 999 |
| 1 | Rotation speed. | 0 to 9999 [rpm] |
| 2 | Load current. | -999 to 999 [%] |
| 3 | Maximum current 1. | -999 to 999 [%] |
| 4 | Maximum current 2. | -999 to 999 [%] |
| 5 | Motor load | -999 to 999 [%] |
| 6 | Regen load. | -999 to 999 [%] |
| 7 | Current station No. | 1 to 360 |
| 8 | Current position. | Auxiliary axis information:<br>Parameter value:<br>Unit: Command unit<br>Auxiliary axis monitor information character string:<br>-99,999.999 to 99,999.999 |
| 9 | Inst station No. | 1 to 360 |

□ **Argument**

| *lIndex* | Description | Data range |
|---|---|---|
| 10 | Inst posn: | Auxiliary axis information:<br>Parameter value:<br>Unit: Command unit<br>Auxiliary axis monitor information character string:<br>-99,999.999 to 99,999.999 |
| 11 | Position loop gain 1. | 0 to 999 |
| 12 | Speed loop gain 1. | 0 to 999 |
| 13 | Position loop gain 2. | 0 to 999 |
| 14 | Speed loop gain 2. | 0 to 999 |
| 15 | Speed integral compensation. | 0 to 999 |
| 16 | Load inertia. | 0 to 999.9 |

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets information about the auxiliary axis monitor.

When the data range in the *Index* table is [Unit: Command unit], the value got needs to be converted according to the command unit set for the Mitsubishi CNC. For the set unit, refer to the Mitsubishi CNC specifications.

<For Linear axis>

| | Metric system | Conversion for 1 of LONG type |
|---|---|---|
| For 10-μm specifications | -999,999.99 to 999,999.99 | 1 = 1/200 (mm) |
| For 1-μm specifications | -99,999.999 to 99,999.999 | 1 = 1/2000 (mm) |
| For 0.1-μm specifications | -9,999.9999 to 9,999.9999 | 1 = 1/20000 (mm) |
| For 10-nm specifications | -999.99999 to 999.99999 | 1 = 1/200000 (mm) |
| For 1-nm specifications | -99.999999 to 99.999999 | 1 = 1/2000000 (mm) |

| | Inch system | Conversion for 1 of LONG type |
|---|---|---|
| For 10-μm specifications | -99,999.999 to 99,999.999 | 1 = 1/200 (inch) |
| For 1-μm specifications | -9,999.9999 to 9,999.9999 | 1 = 1/2000 (inch) |
| For 0.1-μm specifications | -999.99999 to 999.99999 | 1 = 1/20000 (inch) |
| For 10-nm specifications | -999.99999 to 999.99999 | 1 = 1/200000 (inch) |
| For 1-nm specifications | -99.999999 to 99.999999 | 1 = 1/2000000 (inch) |

<For Rotary axis>

| | Metric system | Conversion for 1 of LONG type |
|---|---|---|
| For 10-μm specifications | -999999.99 to 999999.99 | 1 = 1/200 (mm) |
| For 1-μm specifications | -99,999.999 to 99,999.999 | 1 = 1/2000 (mm) |
| For 0.1-μm specifications | -9999.9999 to 9999.9999 | 1 = 1/20000 (mm) |
| For 10-nm specifications | -999.99999 to 999.99999 | 1 = 1/200000 (mm) |
| For 1-nm specifications | -99.999999 to 99.999999 | 1 = 1/2000000 (mm) |

| | Inch system | Conversion for 1 of LONG type |
|---|---|---|
| For 10-μm specifications | -999,999.99 to 999,999.99 | 1 = 1/200 (inch) |
| For 1-μm specifications | -99,999.999 to 99,999.999 | 1 = 1/2000 (inch) |
| For 0.1-μm specifications | -9999.9999 to 9999.9999 | 1 = 1/20000 (inch) |
| For 10-nm specifications | -999.99999 to 999.99999 | 1 = 1/200000 (inch) |
| For 1-nm specifications | -99.999999 to 99.999999 | 1 = 1/2000000 (inch) |

Conversion example) For the linear axis with the 1-μm specifications in the Metric system, when the LONG value got is 710001,

710001 ÷ 2000 = 355.0005.

However, 355.0005 is rounded to minus infinity. Therefore the result is 355.000.

Data in units of 0.5 μm (1/2000 mm) is rounded to minus infinity when displayed.

This means that +0.5 μm is displayed as 0 and -0.5 μm is displayed as -1 μm.

As the string area memory is allocated in this product, the client using VC++ needs to release the string area memory explicitly with **CoTaskMemFree()**.

This function is not supported with C70.

□ **Reference**

□ **Specification**

Axis number

## 2.9.11 IEZNcAxisMonitor::GetAuxAxisDiagnosis Get auxiliary axis diagnostics information

**□ Custom call procedure**

```
HRESULT      GetAuxAxisDiagnosis (
                     LONG lAxisNo,                // (I) Axis No.
                     LONG lIndex,                 // (I) Auxiliary axis diagnostics information type
                     LPOLESTR* lppwszBuffer,      // (O) Auxiliary axis diagnostics information
                                                       character string
                     LONG* plRet                  // (O) Error code
                     )
```

**□ Automation call procedure**

```
             Monitor_GetAuxAxisDiagnosis (
                     lAxisNo As LONG              // (I) Axis No.
                     lIndex As LONG               // (I) Auxiliary axis diagnostics information type
                     lppwszBuffer As STRING*      // (O) Auxiliary axis diagnostics information
                                                       character string
                     ) As LONG                    // (O) Error code
```

**□ Argument**

*lAxisNo*: Sets the axis No. (From Axis 1 = from **1**)

*lIndex*: Sets the auxiliary axis diagnostics information type. Refer to the table below.

*lppwszBuffer*: Gets the auxiliary axis diagnostics information as a **UNICODE** character string.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_READ**: Data is not readable
**EZNC_DATA_READ_ADDR**: Invalid part system, axis No. setting
**EZ_ERR_NOT_SUPPORT**: Not supported

| *lIndex* | Description | Data range |
|---|---|---|
| 0 | Alarm history (1). | "Alarm information by type" with 9 alphanumeric characters |
| 1 | Alarm history (2). | "Alarm information by type" with 9 alphanumeric characters |
| 2 | Alarm history (3). | "Alarm information by type" with 9 alphanumeric characters |
| 3 | Alarm history (4). | "Alarm information by type" with 9 alphanumeric characters |
| 4 | Alarm history (5). | "Alarm information by type" with 9 alphanumeric characters |
| 5 | Alarm history (6). | "Alarm information by type" with 9 alphanumeric characters |

**□ Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

**□ Function**

Gets the auxiliary axis diagnostics information.
As the string area memory is allocated in this product, the client using VC++ needs to release the string area memory explicitly with **CoTaskMemFree()**.
This function is not supported with C70.

**□ Reference**

**□ Specification**

Axis number

## 2.9.12 IEZNcAxisMonitor::GetAuxAxisVersion Get auxiliary axis version information

□ **Custom call procedure**

```
HRESULT      GetAuxAxisVersion (
                       LONG lAxisNo,                // (I) Axis No.
                       LONG lIndex,                 // (I) Auxiliary axis version information type
                       LPOLESTR* lppwszBuffer,      // (O) Auxiliary axis version information
                                                       character string
                       LONG* plRet                  // (O) Error code
                       )
```

□ **Automation call procedure**

```
             Monitor_GetAuxAxisVersion (
                       lAxisNo As LONG              // (I) Axis No.
                       lIndex As LONG               // (I) Auxiliary axis version information type
                       lppwszBuffer As STRING*      // (O) Auxiliary axis version information
                                                       character string
                       ) As LONG                    //   (O) Error code
```

□ **Argument**

*lAxisNo*: Sets the axis No. (From Axis 1 = from **1**)

*lIndex*: Sets the auxiliary axis version information type.

*lppwszBuffer*: Sets the auxiliary axis version information as a **UNICODE** character string.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_READ**: Data is not readable
**EZNC_DATA_READ_ADDR**: Invalid system, spindle specification
**EZ_ERR_NOT_SUPPORT**: Not supported

| *lIndex* | Description | Data range |
|---|---|---|
| 0 | Unit type | 9 alphanumeric characters |
| 1 | Unit serial No. | 16 alphanumeric characters |
| 2 | Motor | 9 alphanumeric characters |

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets auxiliary axis version information.
As the string area memory is allocated in this product, the client using VC++ needs to release the string area memory explicitly with **CoTaskMemFree()**.
This function is not supported with C70.

□ **Reference**

□ **Specification**

Axis number

## 2.9.13 IEZNcAxisMonitor::GetDowelTime Get remaining dwell time

□ **Custom call procedure**

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

□ **Automation call procedure**

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

□ **Argument**

*pdTime*: Returns the remaining dwell (G04) time.
Unit: Second
Value: 0.000 to 99,999.999 (s)

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_READ**: Data is not readable
**EZNC_DATA_READ_ADDR**: Invalid part system, axis No. setting

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Returns the remaining dwell (G04) time. Unit: Second, output up to second, 1/1000 (second).

□ **Reference**

□ **Specification**

System

## 2.9.14 IEZNcAxisMonitor:: GetPowerConsumption Get current power consumption

□ **Custom call procedure**

```
HRESULT    GetPowerConsumption (
                   LONG lAxisNo,                    // (I) Axis No.
                   DOUBLE** ppdPower,               // (O) Power consumption
                   LONG* plRet                      // (O) Error code
                   )
```

□ **Automation call procedure**

```
           GetPowerConsumption (
                   lAxisNo As LONG                  // (I) Axis No.
                   pvPower As VARIANT*              // (I) Power consumption
                   ) As LONG                        // (O) Error code
```

□ **Argument**

*lAxisNo*: Sets the axis No. (From Axis 1 = from **1**)

*ppdPower* : Returns the current total power consumption as an array. The data array is secured on the EZSocket side, so explicitly release it on the client side using CoTaskMemFree().
Refer to the index table for the got power consumption.

Index table

| Array Index | Type of power consumption |
|---|---|
| 0 | Present consumption of entire drive system |
| 1 | Present power consumption of servo axis in drive system (fluctuating part) |
| 2 | Present power consumption of spindle in drive system (fluctuating part) |

Automation argument:
*lAxisNo*: See the explanation of *lAxisNo*.

*pvPower* : Returns the power consumption array as a VARIANT.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_ADDR**: Invalid part system, axis No. setting
**EZNC_DATA_READ_READ**: Data is not readable
**EZ_ERR_NOT_SUPPORT**: Not supported

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets the present consumption of entire drive system as an array.
This function is not supported with the C70 or M700 series.

□ **Reference**

□ **Specification**

## 2.9.15 IEZNcAxisMonitor:: GetIntegralPower Get integral power

□ **Custom call procedure**

```
HRESULT     GetPowerConsumption (
                  LONG lAxisNo,                // (I) Axis No.
                  LONG lIndex,                 // (I) Parameter number
                  DOUBLE** ppdPower,           // (O) Power consumption
                  LONG* plRet                  // (O) Error code
                  )
```

□ **Automation call procedure**

```
            GetPowerConsumption (
                  lAxisNo As LONG              // (I) Axis NO.
                  lIndex As LONG,              // (I) Parameter number
                  pvPower As VARIANT*          // (I) Power consumption
                  ) As LONG                    // (O) Error code
```

□ **Argument**

*lAxisNo*: Sets the axis No. (From Axis 1 = from **1**)

*lIndex*：Sets the number of the integral power to be got.

*ppdPower* : Returns the total power consumption as an array. The data array is secured on the EZSocket side, so explicitly release it on the client side using CoTaskMemFree().
Refer to the index table for the got power consumption.
Index table

| Array Index | Type of power consumption |
|---|---|
| 0 | Accumulated consumption of entire drive system |
| 1 | Drive system's fixed consumption correction |
| 2 | Accumulated consumption of servo axis in drive system (fluctuating part) |
| 3 | Accumulated regeneration of servo axis in drive system (fluctuating part) |
| 4 | Accumulated consumption of spindle in drive system (fluctuating part) |
| 5 | Accumulated regeneration of spindle in drive system (fluctuating part) |

Automation argument:
*lAxisNo*: See the explanation of *lAxisNo*.

*lIndex*：See the explanation of *lIndex*.

*pvPower*：Returns the power consumption array as a VARIANT.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_ADDR**: Invalid part system, axix No. setting
**EZ_ERR_DATA_RANGE**: Invalid argument data range
**EZNC_DATA_READ_READ**: Data is not readable
**EZ_ERR_NOT_SUPPORT**: Not supported

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets the current total power consumption as an array.
This function is not supported with the C70 or M700 series.

□ **Reference**

□ **Specification**

**C70** **M700** **M800**

## 2.10.1 IEZNcRunStatus::GetInvalidStatus Get disabled status

□ **Custom call procedure**

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

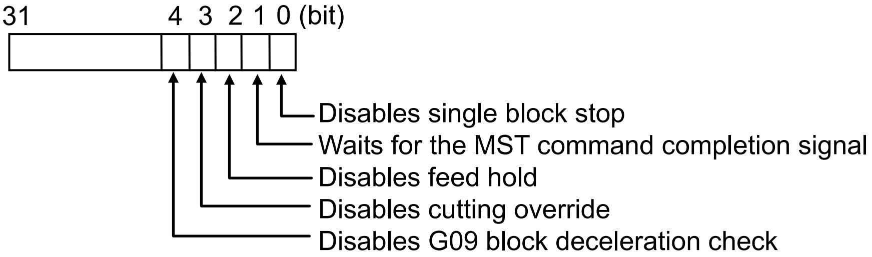
□ **Automation call procedure**

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

□ **Argument**

*plStatus :* Returns the disabled bit flag.

| Value | Meaning |
|---|---|
| **0** | OFF |
| **1** | ON |


31
4 3 2 1 0 (bit)
Disables single block stop
Waits for the MST command completion signal
Disables feed hold
Disables cutting override
Disables G09 block deceleration check

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_ADDR**: Invalid system specification
**EZNC_DATA_READ_READ**: Data is not readable

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Returns the disabled flag.

Disabled status: Disables single block stop
Waits for the MST command completion signal
Disables feed hold
Disables cutting override
Disables G09 block deceleration check

□ **Reference**

□ **Specification**

System

## 2.10.2 IEZNcRunStatus::GetCommandStatus Get operation command status

□ **Custom call procedure**

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

□ **Automation call procedure**

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

□ **Argument**

*plStatus* : Returns the operation command status with any of the following numbers.

| Value | Meaning | Value | Meaning |
|---|---|---|---|
| **0** | Positioning (independent axis) | **15** | 3rd reference position verification |
| **1** | Positioning (linear) | **16** | 4th reference position verification |
| **2** | Linear interpolation | **17** | Automatic reference position return |
| **3** | Circular interpolation (CW) | **18** | Return from the automatic reference position |
| **4** | Circular interpolation (CCW) | **19** | 2nd reference position return |
| **5** | Helical interpolation (CW) | **20** | 3rd reference position return |
| **6** | Helical interpolation (CCW) | **21** | 4th reference position return |
| **7** | Reserved | **22** | Skip function |
| **8** | Reserved | **23** | Multi-skip function 1 |
| **9** | Reserved | **24** | Multi-skip function 2 |
| **10** | Reserved | **25** | Multi-skip function 3 |
| **11** | Time command dwell | **26** | Threading |
| **12** | Reserved | **27** | Reserved |
| **13** | 1st reference position verification | **28** | Reserved |
| **14** | 2nd reference position verification | **29** | Coordinate system setting |

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_ADDR**: Invalid part system setting
**EZNC_DATA_READ_READ**: Data is not readable

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets the operation command status.

□ **Reference**

□ **Specification**

System

## 2.10.3 IEZNcRunStatus::GetCuttingMode Get cutting mode status

□ **Custom call procedure**

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

□ **Automation call procedure**

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

□ **Argument**

*plMode*: Returns the cutting mode.

| Value | Meaning |
|---|---|
| **1** | In G01, G02, G03, G31, G33, G34, or G35 mode |
| **0** | In any of other modes |

*plRet*: Returns an error code. (Upon automation, the return value is used.)

**S_OK**: Normal termination
**EZNC_DATA_READ_ADDR**: Invalid part system setting
**EZNC_DATA_READ_READ**: Data is not readable

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets the cutting mode.

□ **Reference**

□ **Specification**

System

## 2.10.4 IEZNcRunStatus::GetAxisStatus Get servo axis status

□ **Custom call procedure**

```
HRESULT      GetAxisStatus(
                    LONG lAxisNo,                    // (I) Axis No.
                    LONG lType,                      // (I) Status type
                    LONG*  plStatus,                 // (O) Servo axis status
                    LONG* plRet                      // (O) Error code
                    )
```

□ **Automation call procedure**

```
             Status_GetAxisStatus(
                    lAxisNo As LONG                  // (I) Axis No.
                    lType As LONG                    // (I) Status type
                    plStatus As LONG*                // (O) Servo axis status
                    ) As LONG                        // (O) Error code
```

□ **Argument**

*lAxisNo*: Sets the axis No. (From Axis 1 = from **1**)
Valid when *lType* = 4. All axes information in the set part system can get when *Type* = number other than 4.

*lType*: Sets the status type.

*plStatus* : Returns the servo axis status.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZ_ERR_DATA_TYPE**: Invalid argument data type
**EZNC_DATA_READ_ADDR**: Invalid system, spindle specification
**EZNC_DATA_READ_READ**: Data is not readable

| *lType* | Description | Data range |
|---|---|---|
| **0** | 1st reference position return completion. | A bit corresponding to the returned axis becomes 1.<br>Example) 00000101 = The 1st and 3rd axes returns have been completed. |
| **1** | 2nd reference position return completion. | A bit corresponding to the returned axis becomes 1.<br>Example) 00000010 = The 2nd axis return has been completed. |
| **2** | 3rd reference position return completion. | A bit corresponding to the returned axis becomes 1.<br>Example) 00001010 = The 2nd and 3th axes returns have been completed. |
| **3** | Fourth reference position return completion. | A bit corresponding to the returned axis becomes 1.<br>Example) 00001000 = The 4th axis return has been completed. |
| **4** | Axis status (axis being removed).<br>The axis specification is required. | 0: Axis not being removed<br>1: Axis being removed |
| **5** | Axis status (servo off). | Axis with servo turned off<br>A bit corresponding to the axis becomes 1. |
| **6** | Axis status (mirror image). | Mirror image axis. A bit corresponding to the axis set for mirror image becomes 1. |

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

| | |
|---|---|
| □<br>**Function** | Gets the servo axis status. |
| □<br>**Reference** | |
| □<br>**Specification** | System (Part system setting is required when the M700/M800 series is used or lType is 4 or 6.),<br>Axis number |

## 2.10.5 IEZNcRunStatus::GetRunStatus Get operation status

□ **Custom call procedure**

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

□ **Automation call procedure**

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

□ **Argument**

*lIndex*: Sets the status number.

*plStatus*: Returns the set operation status.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZ_ERR_DATA_TYPE**: Invalid argument data type
**EZNC_DATA_READ_ADDR**: Invalid part system setting
**EZNC_DATA_READ_READ**: Data is not readable

| *lIndex* | Description | Data range |
|---|---|---|
| **0** | Tool length measurement. | 0: Tool length not being measured<br>1: Tool length being measured |
| **1** | In automatic operation "run".<br>Gets the status indicating that the system is operating automatically. | 0: Not operating in automatically<br>1: Operating automatically |
| **2** | Automatic operation “start”.<br>Gets the status indicating that the system is operating automatically and that a movement command or M, S, T, B process is being executed. | 0: Not starting automatic operation<br>1: Starting automatic operation |
| **3** | Automatic operation "pause"<br>Gets the status indicating that automatic operation is paused while executing a movement command or miscellaneous command with automatic operation. | 0: Automatic operation not paused<br>1: Automatic operation paused |

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□
**Function**

Gets the operation status.
Automatic operation paused is enabled only for the M700/M800 series.
Signals that indicate the status of automatic operation with the PLC interface include ‘in automatic operation', ‘automatic operation start’, and ‘automatic operation pause’. The ON/OFF statuses of these three signals in each state are shown below.

| | In automatic operation "RUN" (OP) | In automatic operation "START" (STL) | In automatic operation "PAUSE" (SPL) |
|---|---|---|---|
| In "reset" | 0 | 0 | 0 |
| Automatic operation stop condition | 1 | 0 | 0 |
| Automatic operation pause condition | 1 | 0 | 1 |
| Automatic operation start condition | 1 | 1 | 0 |

Each status represents the following type of state.
* Reset state
Automatic operation is stopped because of the reset conditions.
(All states in which the system is not operating automatically correspond to this.)
* Automatic operation stopped state
Automatic operation is stopped after executing one block.
(Single block stop corresponds to this.)
* Automatic operation pause state
Automatic operation is stopped during the execution of one block.
(The automatic operation pause (*SP) signal OFF state corresponds to this.)
* Automatic operation started state
Automatic operation is actually being executed.

Refer to the PLC Interface Manual for details.

□
**Reference**

□
**Specification**

System

C70 M700 M800

## 2.11.1 IEZNcFile6::FindDir2 Search directory

□ **Custom call procedure**

```
HRESULT     FindDir2(
                 LPCOLESTR lpcwszDirectryName,   // (I) Directory name
                 LONG lFileType,                 // (I) Read type and format
                 LPOLESTR* lppwszFileInfo,       // (O) File information character string
                 LONG* plRet                     // (O) Error code
                 )
```

□ **Automation call procedure**

```
            File_FindDir2(
                 lpcwszDirectryName As STRING    // (I) Directory name
                 lFileType As LONG               // (I) Read type and format
                 lppwszFileInfo As STRING*       // (O) File information character string
                 ) As LONG                       // (O) Error code
```

□ **Argument**

*lpcwszDirectryName*: Sets the directory name as a **UNICODE** character string.
Specify directory with an absolute path as follows:

Drive name + ":" + \Directory name\File name …Gets the set file name information. (Note 1)
Drive name + ":" + "\Directory name" …Gets the set directory name information. (Note 1)
Drive name + ":" + \Directory name\ …Gets the set directory information.

(Note 1) This setting is for the M700/M800 series.

*lFileType*: Sets the type and format of data to be read.
The following can also be set with pipe (|). When **NULL** is set, file information is read.

| Value | Meaning |
|---|---|
| **EZNC_DISK_DIRTYPE** | Directory information read |
| **EZNC_DISK_COMMENT** | Comment information read (on the NC control unit side only) |
| **EZNC_DISK_DATE** | Date information read (on the personal computer side only) |
| **EZNC_DISK_SIZE** | Size information read |

*lppwszFileInfo:* Gets file information as a **UNICODE** character string.
The format of file information becomes as follows:

File name\tSize\tDate\tComment\0

A **TAB** code is inserted between file name, size, date, and comments.
The end of the data becomes a **NULL** code.

*plRet*: Returns whether or not there is file information read or returns an error code. (Upon automation, the return value is used.)

**0**: When there is no file information
**1** or more: When there is file information
**EZNC_FILE_DIR_DATASIZE:** Exceeded maximum data size
**EZNC_FILE_DIR_NOTOPEN**: Not open
**EZNC_FILE_DIR_READ:** File information read error
**EZNC_FILE_DIR_ALREADYOPENED**: A different directory is already opened
**EZNC_FILE_DIR_NODRIVE:** Drive does not exist
**EZNC_FILE_DIR_NODIR:** Directory does not exist

(Note 2) If an error occurs on the personal computer, the error code **EZNC_FILE_**… becomes **EZNC_PCFILE_**….

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Searches directory.

In **FindDir2()**, information of one file can be read by reading once. To continuously get directory information, a list of file names in the set directory can be got by calling **FindNextDir2()** repeatedly. The format of file information to be stored in the area indicated by *lpszFileInfo* is as follows:

File name\tSize\tDate\tComment\0

A **TAB** code is inserted between file name, size, date, and comments. The end of the data becomes a **NULL** code. Only information that follows file name set in the read type is stored. For example, if “**EZNC_DISK_COMMENT|EZNC_DISK_DATE**” is set, information is as follows:

Filename\tDate\tComment\0

If “**EZNC_DISK_SIZE|EZNC_DISK_COMMENT**” is set for a file to which comments cannot be added, the comment information will not be output and the comment will be as follows:

File name\tSize\t\0

For a file from which date cannot get, setting of **"EZNC_DISK_SIZE|EZNC_DISK_DATE|EZNC_DISK_COMMENT"** becomes as follows with no date information output.

File name\tSize\t\tComment\0

* The file from which a date cannot get refers to the file on the NC control unit side.

As the character string area memory is allocated in this product, the client using VC++ needs to release the character string area memory explicitly with **CoTaskMemFree()**.

(Note 1) Reading the directory size information on the NC-side compact flash (M700 series) or SD card (M800 series) is not supported. The directory size information read is invalid.

(Note 2 ) For the **C70**, when the file on the personal computer is specified and 0 is specified for *lFileType*,

\t is added to the end of file information got (filename\t\0).

To use file information got, remove \t before using.

□ **Restrictions**

When **FindDir2()** is used, **FindDir2(), OpenFile3(),OpenNcFile2()** cannot be executed until **ResetDir()** is executed.

When executed, an error "**EZNC_FILE_DIR_ALREADYOPENED** (0x80030101) A different directory is already opened" occurs. When using it, immediately execute **ResetDir()** after executing FindDir2().

□ **Reference**

**FindNextDir2(), ResetDir()**

□ **Specifica-tion**

## 2.11.2 IEZNcFile6::FindNextDir2 Search next directory

□ **Custom call procedure**

```
HRESULT       FindNextDir2(
                        LPOLESTR* lppwszFileInfo,      // (O) File information character string
                        LONG* plRet                    // (O) Error code
                        )
```

□ **Automation call procedure**

```
              File_FindNextDir2(
                        lppwszFileInfo As STRING*      // (O) File information character string
                        ) As LONG                      // (O) Error code
```

| | |
|---|---|
| □ **Argument** | *lppwszFileInfo:* Gets file information as a **UNICODE** character string.<br>The format of file information becomes as follows:<br>File name\tSize\tDate\tComment\0<br>A **TAB** code is inserted between file name, size, date, and comments.<br>The end of the data becomes a **NULL** code.<br><br>*plRet*: Returns whether or not there is file information read or returns an error code. (Upon automation, the return value is used.)<br>**0**: When there is no file information<br>**1** or more: When there is file information<br>**EZNC_FILE_DIR_DATASIZE:** Exceeded maximum data size<br>**EZNC_FILE_DIR_NOTOPEN**: Not open<br>**EZNC_FILE_DIR_READ:** File information read error<br>**EZNC_FILE_DIR_NODRIVE:** Drive does not exist<br>(Note) If an error occurs on the personal computer, the error code **EZNC_FILE_…** becomes **EZNC_PCFILE_….** |

| □ **Return value** | Value | Meaning |
|---|---|---|
| | **S_OK** | Normal termination |
| | **S_FALSE** | Communication failure |

| | |
|---|---|
| □ **Function** | Continuously searches for a directory.<br>To continuously get directory information after executing **FindDir2()**, a list of file names in the set directory can get by calling **FindNextDir2()** repeatedly. The format of file informatio n that is stored in the area indicated by *lpszFileInfo* is the same as that for **FindDir2()**.<br>As the character string area memory is allocated in this product, the client using VC++ ne eds to release the character string area memory explicitly with **CoTaskMemFree()**. |
| □ **Reference** | **FindDir2(), ResetDir()** |
| □ **Specifica-tion** | |

## 2.11.3 IEZNcFile6::ResetDir Terminate directory search

□ **Custom call procedure**

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

□ **Automation call procedure**

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

| | |
|---|---|
| □ **Argument** | *plRet*: Returns an error code. (Upon automation, the return value is used.)<br>**EZNC_FILE_DIR_DATASIZE:** Exceeded maximum data size<br>**EZNC_FILE_DIR_NOTOPEN**: Not open<br>**EZNC_FILE_DIR_READ:** File information read error<br>(Note) If an error occurs on the personal computer, the error code **EZNC_FILE_**… becomes **EZNC_PCFILE_**…. |

| □ **Return value** | Value | Meaning |
|---|---|---|
| | **S_OK** | Normal termination |
| | **S_FALSE** | Communication failure |

| | |
|---|---|
| □ **Function** | Terminates directory search.<br>To search for a directory again, execute **FindDir2()**. |
| □ **Reference** | **FindDir2()** |
| □ **Specifica-tion** | |

## 2.11.4 IEZNcFile6::Copy2 Copy file

□ **Custom call procedure**

```
HRESULT       Copy2(
                      LPCOLESTR lpcwszSrcFileName,    // (I)  Transfer source file name
                      LPCOLESTR lpcwszDstFileName,    // (I)  Transfer destination file name
                      LONG* plRet                     // (O)  Error code
                      )
```

□ **Automation call procedure**

```
              File_Copy2(
                      lpcwszSrcFileName As STRING     // (I) Transfer source file name
                      lpcwszDstFileName As STRING     // (I) Transfer destination file name
                      ) As LONG                       // (O) Error code
```

□ **Argument**

*lpcwszSrcFileName*: Sets transfer source file name using **UNICODE** character strings.

*lpcwszDstFileName*: Sets transfer destination file name as a **UNICODE** character strings.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_FILE_COPY_BUSY:** Copy is disabled (during operation)
**EZNC_FILE_COPY_ENTRYOVER:** Registration limit exceeded
**EZNC_FILE_COPY_FILEEXIST:** Copy destination file already exists
**EZNC_FILE_COPY_FILESYSTEM**: File system error
**EZNC_FILE_COPY_ILLEGALNAME:** Invalid file name format
**EZNC_FILE_COPY_MEMORYOVER:** Memory capacity exceeded
**EZNC_FILE_COPY_NODIR:** Directory does not exist
**EZNC_FILE_COPY_NODRIVE**: Drive does not exist
**EZNC_FILE_COPY_NOFILE**: File does not exist
**EZNC_FILE_COPY_PLCRUN**: Copy is disabled (programmable controller in operation)
**EZNC_FILE_COPY_READ**: Transfer source file is not readable
**EZNC_FILE_COPY_WRITE:** Transfer destination file is not writable
**EZNC_FILE_COPY_PROTECT:** Copying is disabled (protected)
**EZNC_PCFILE_COPY_CREATE:** File cannot be created (PC only)
**EZNC_PCFILE_COPY_OPEN**: File cannot be opened (personal computer only)
(Note) If an error occurs on the personal computer, the error code **EZNC_FILE_…** becomes **EZNC_PCFILE_….**

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Copies the file set by *lpcwszSrcFileName* to *lpcwszDstFileName*.
Set a file name with an absolute path as follows:

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

*lpcwszDstFileName* must not be a file name that already exists. The transfer destination directory must already exist.

In the C70, multiple program files under \PRG\USER can be collectively copied to a file on a personal computer. In this case, specify "*.*" as a file name for *lpcwszSrcFileName* (example: M01:\PRG\USER\*.*). Set any file name on a personal computer for *lpcwszDstFileName* (example: C:\PLURAL.PRG). To expand files combined into one in the C70, set any file name on a personal computer for *lpcwszSrcFileName* (example: C:\PLURAL.PRG) and set "ALL.PRG" as a file name under \PRG\USER of *lpcwszDstFileName* (example: M01:\PRG\USER\ALL.PRG).

This method does not check whether the set directory and file name are appropriate or not. It is recommended to check the appropriateness of file name and directory for irregular operations such as transfer between files with different types and applications (example: overwriting the user program (\PRG\USER\ to.PRG) with parameter file (PARAMET.BIN)) or copying a file to a directory with different purposes.

(Note) Do not perform this operation during automatic operation of the NC control unit. (C70 only)

□ **Reference**

**Delete2(), Rename2()**

□ **Specification**

## 2.11.5 IEZNcFile6::Delete2 Delete file

□ **Custom call procedure**

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

□ **Automation call procedure**

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

□ **Argument**

*lpcwszFileName*: Sets the file name as a **UNICODE** character string.

*plRet*: Returns an error code. (Upon automation, the return value is used.)

**S_OK:** Normal termination
**EZNC_FILE_DEL_BUSY**: Deletion is disabled (during operation)
**EZNC_FILE_DEL_FILESYSTEM**: File system error
**EZNC_FILE_DEL_ILLEGALNAME**: Invalid file name format
**EZNC_FILE_DEL_PROTECT:** Deletion is disabled (protected)
**EZNC_FILE_DEL_NODIR**: Directory does not exist
**EZNC_FILE_DEL_NODRIVE:** Drive does not exist
**EZNC_FILE_DEL_NOFILE**: File does not exist
**EZNC_PCFILE_DEL_NOTDELETE**: File cannot be deleted
(Note) If an error occurs on the personal computer, the error code **EZNC_FILE_…** becomes **EZNC_PCFILE_….**

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Deletes the file set in *lpcwszFileName*.
Set a file name with an absolute path as follows:
Drive name + ":" + \Directory name\File name

(Note) Do not perform this operation during automatic operation of the NC control unit. (C70 only) For the M700/M800 series, the operation can be performed unless automatic operation of the file intended for operation is being carried out.

□ **Reference**

**Copy2(), Rename2()**

□ **Specification**

C70 M700 M800

## 2.11.6 IEZNcFile6::Rename2 Change file name

□ **Custom call procedure**

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

□ **Automation call procedure**

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

□ **Argument**

*lpcwszSrcFileName*: Sets an old file name.

*lpcwszDstFileName*: Sets a new file name.

*plRet*: Returns an error code. (Upon automation, the return value is used.)

**S_OK**: Normal termination
**EZNC_FILE_REN_BUSY**: Rename is disabled (during operation)
**EZNC_FILE_REN_FILEEXIST**: New file name already exists
**EZNC_FILE_REN_FILESYSTEM**: File system error
**EZNC_FILE_REN_ILLEGALNAME**: Invalid file name format
**EZNC_FILE_REN_PROTECT**: Rename is disabled (protected)
**EZNC_FILE_REN_NODIR**: Directory does not exist
**EZNC_FILE_REN_NODRIVE:** Drive does not exist
**EZNC_FILE_REN_NOFILE**: File does not exist
**EZNC_PCFILE_REN_NOTRENAME**: Rename is disabled
**EZNC_PCFILE_REN_SAMENAME**: New and old file names are identical
(Note) If an error occurs on the personal computer, the error code **EZNC_FILE_…** becomes **EZNC_PCFILE_….**

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Changes the file name set for *lpcwszSrcFileName* to that set for *lpcwszDstFileName*.
For *lpszSrcFileName*, set with an absolute path.
Drive name + ":" + \Directory name\File name

For *lpcwszDstFileName*, set only file name without drive name and directory. For *lpcwszDstFileName*, a file name that already exists must not be set.

(Note) Do not perform this operation during automatic operation of the NC control unit. (C70 only) For the M700/M800 series, the operation can be performed during automatic operation of the NC control unit, unless automatic operation of the file intended for operation is being carried out.

□ **Reference**

**Copy2(), Delete2()**

□ **Specification**

## 2.11.7 IEZNcFile6::GetDriveInformation Get drive information

□ **Custom call procedure**

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

□ **Automation call procedure**

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

□ **Argument**

*lppwszDriveInfo*: Gets drive information as a **UNICODE** character string.
The format of drive information is as follows:
Drive name:**CRLF**Drive name:**CRLF**…Drive name:**CRLF**\0
To separate drive names, CR, LF codes are inserted between them. The end of the data becomes **CR, LF** codes and a **NULL** code. The end of the data becomes a **NULL** code.

*plRet*: Returns the size of drive information got or an error code. (Upon automation, the return value is used.)
0: When a drive does not exist
1 or more: Number of bytes
EZNC_FILE_DRVLIST_READ: Drive information read error
**EZNC_FILE_DIR_NODRIVE:** Drive does not exist
(Note) If an error occurs on the personal computer, the error code **EZNC_FILE_…** becomes **EZNC_PCFILE_….**

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Reads drive information of the NC control unit that is currently connected.
The format of drive information is as follows:
Drive name:**CRLF**Drive name:**CRLF**…Drive name:**CRLF**\0
To separate drive names, CR, LF codes are inserted between them. The end of the data becomes **CR, LF** codes and a **NULL** code. The end of the data becomes a **NULL** code.
Drive information on a personal computer cannot be read.
As the character string area memory is allocated in this product, the client using VC++ needs to release the character string area memory explicitly with **CoTaskMemFree()**.

□ **Reference**

**GetDriveSize()**

□ **Specification**

## 2.11.8 IEZNcFile6::GetDriveSize Get free drive space

□ **Custom call procedure**

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

□ **Automation call procedure**

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

□ **Argument**

*lpcwszDirectryName*: Sets the directory name as a **UNICODE** character string.
Directory is set with an absolute path as follows:
Drive name + ":" + \Directory name\

*plDriveSize*: Gets free space of the set directory. (unit: byte)

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**EZNC_FILE_REN_FILESYSTEM**: File system error
**EZNC_FILE_DIR_NODRIVE:** Drive does not exist
**EZNC_FILE_DISKFREE_NODIR**: Directory does not exist
**EZNC_FILE_DISKFREE_NODRIVE:** Drive does not exist
(Note) If an error occurs on the personal computer, the error code **EZNC_FILE_…** becomes **EZNC_PCFILE_….**

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Returns free space of the directory specified for *lpcwszDirectryName*.
The unit of free space is a byte.
Directory is specified with an absolute path as follows:
Drive name + ":" + \Directory name\

When a drive on a personal computer is set for drive name, setting of directory is ignored, and free space of the drive is returned.

When the NC control unit is set for a drive name, free space of the set directory is returned. When a subdirectory exists in the set directory, usage in the subdirectory is excluded from calculation of free space.

(Note) When a directory name (M01:\IC1\) that corresponds to the compact flash (M700 series) or SD card (M800 series) on the NC side is set, up to 2GB can get.

□ **Reference**

**GetDriveInformation()**

□ **Specification**

## 2.11.9 IEZNcFile6::GetDriveSize2 Get free drive space

□ **Custom call procedure**

```
HRESULT     GetDriveSize2(
                 LPCOLESTR lpcwszDirectryName,   // (I) Directory name
                 LONG* plDriveSize,              // (O) Free space
                 LONG* plRet                     // (O) Error code
                 )
```

□ **Automation call procedure**

```
            File_GetDriveSize2(
                 lpcwszDirectryName As STRING    // (I) Directory name
                 plDriveSize As LONG*            // (O) Free space
                 ) As LONG                       // (O) Error code
```

□ **Argument**

*lpcwszDirectryName*: Sets the directory name as a **UNICODE** character string.
Directory is set with an absolute path as follows:
Drive name + ":" + \Directory name\

*plDriveSize*: Gets free space of the set directory. (unit: byte)

Automation argument:
*lpcwszDirectryName:* See the explanation of *lpcwszDirectryName.*

*pvDriveSize*: Gets the free space of the set directory as a character string.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**EZNC_FILE_REN_FILESYSTEM**: File system error
**EZNC_FILE_DIR_NODRIVE:** Drive does not exist
**EZNC_FILE_DISKFREE_NODIR**: Directory does not exist
**EZNC_FILE_DISKFREE_NODRIVE:** Drive does not exist
(Note) If an error occurs on the personal computer, the error code **EZNC_FILE_…** becomes **EZNC_PCFILE_….**

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Returns free space of the directory set for *lpcwszDirectryName*.
The unit of free space is a byte.
Directory is set with an absolute path as follows:
Drive name + ":" + \Directory name\

When a drive on a personal computer is set for drive name, setting of directory is ignored, and free space of the drive is returned.

When the NC control unit is set for a drive name, free space of the set directory is returned. When a subdirectory exists in the set directory, usage in the subdirectory is excluded from calculation of free space.

This is valid only for the M800 series.
This function is not supported with C70.

□ **Reference**

**GetDriveInformation()**

□ **Specifica-tion**

## 2.11.10 IEZNcFile6::OpenFile3 Open file

□ **Custom call procedure**

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

□ **Automation call procedure**

```
           File_OpenFile3(
                bstrFileName As STRING       // (I) File name containing a path
                lModeAs LONG                 // (I) Open mode
                ) As LONG                    // (O) Error code
```

□ **Argument**

*lpcwszFileName*: Sets the file name containing a path as a **UNICODE** character string.
A file is set with an absolute path as follows:
Drive name + ":" + \Directory name\File name

For a list of files in the NC control unit that is accessible, refer to Table 2-1, Table 2-2 and Table 2-3. All files except for machining programs in the NC control unit can be backed up, but cannot be edited.

*bstrFileName*: Refer to the explanation of *lpcwszFileName*.

*lMode*: Sets open mode.

| Value | Meaning |
|---|---|
| **EZNC_FILE_READ** | Read mode |
| **EZNC_FILE_WRITE** | Write mode |
| **EZNC_FILE_OVERWRITE** | Overwrite mode (writes even if the specified file exists.) |

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_FILE_OPEN_OPEN**: File cannot be opened
**EZNC_FILE_OPEN_ALREADYOPENED**: File is already open
**EZNC_FILE_OPEN_FILEEXIST**: File already exists (in write mode)
**EZNC_FILE_OPEN_FILENOEXIST:** File does not exist (in read mode)
**EZNC_FILE_OPEN_MODE**: Invalid open mode
**EZNC_FILE_OPEN_NOTOPEN**: File cannot be opened
**EZNC_FILE_OPEN_CREATE**: Temporary file cannot be created (in write mode)
**EZNC_FILE_READFILE_CREATE**: Temporary file cannot be created (in read mode)
**EZNC_FILE_DIR_NODRIVE:** Drive does not exist
**EZNC_FILE_DIR_ALREADYOPENED**: A different directory is already opened
**EZNC_FILE_OPEN_ILLEGALPATH**: Invalid file path

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

|  |  |
|---|---|
| □<br>**Function** | Opens a file in the specified mode. The directory that creates a temporary file is created in the following order of priority:<br>• Directory set with environment variable TMP<br>• Directory to which the product is installed<br><br>The temporary file name is MELDASn. A number is placed in n.<br><br>(Note 1) Make sure to close the open file with **CloseFile2()** (or **AbortFile2()**). The temporary file will remain if **CloseFile2()** is not used.<br>(Note 2) Do not perform write or overwrite operation during automatic operation of the NC control unit. (C70 only)<br>For the M700/M800 series, write or overwrite operation can be performed during automatic operation of the NC control unit, unless automatic operation of the file intended for operation is being carried out. Read operation can be performed during automatic operation of the NC control unit.<br>(Note 3) It takes **C70** approximately 20 s to read SRAM.BIN (SRAM data (binary format)). Other methods cannot be used during that time. |
| □<br>**Reference** | **CloseFile2(), AbortFile2(), ReadFile2(), WriteFile()** |
| □<br>**Specifica-tion** |  |

Table 2-1 M700 series (version A0 or later) List of accessible files

| File description | Directory | File name | Remarks |
|---|---|---|---|
| Machining program | M01:\PRG\USER\ | Program No. | |
| MTB macro | M01:\PRG\MMACRO\ | Program No. | Program No. is O100001000 to O199999999. |
| Fixed cycle program | M01:\PRG\FIX\ | Program No. | Program No. is 0100000010 to 100009999. |
| MDI program | M01:\PRG\MDI\ | MDI.PRG | |
| Parameters [User, machine] | M01:\PRM\ | ALL.PRM | |
| Auxiliary axis parameter | M01:\PRM\ | AUXAXIS.PRM | M700/M700VW only |
| DeviceNet parameter file | M01:\PRM\ | DEVICENT.PRM | |
| Rotary axis geometric deviation parameter file | M01:\PRM\ | GEOMETRY.PRM | |
| PLC program | M01:\LAD\ | USERPLC.LAD | |
| Workpiece offset data | M01:\DAT\ | WORK.OFS | |
| Tool compensation amount 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 | |
| Tool life management data file | M01:\DAT\ | TLIFE.TLF | |
| Tool management data file | M01:\DAT\ | TOOLMNG.DAT | |
| SRAM open data file | M01:\DAT\ | SRAMOPEN.DAT | |
| Device open data file | M01:\DAT\ | DEVOPEN.DAT | |
| Machining surface data | M01:\DAT\ | RNAVI.DAT | |
| Extended SRAM data (binary format) | M01:\DAT\ | EXTSRAM.BIN | |
| All history | M01:\LOG\ | ALLLOGLOG | Key history, alarm history, programmable controller I/O signal history, AC input power supply faults history |
| Key history | M01:\LOG\ | KEYLOGLOG | |
| Sampling data file | M01:\LOG\ | NCSAMP.CSV | |
| NC-side compact flash | M01:\IC1\ | Any | The NC-side compact flash (hereinafter referred to as the "NC-side CF card") is recognized as DS (data server) from the NC control unit. Used for data backup, storing large-capacity programs, etc. |

Table 2-2 M800 series List of accessible files

| File description | Directory | File name | Remarks |
|---|---|---|---|
| Machining program | M01:\PRG\USER\ | Program No. | |
| MTB macro | M01:\PRG\MMACRO\ | Program No. | Program No. is 100010000 to 199999999. |
| Fixed cycle program | M01:\PRG\FIX\ | Program No. | Program No. is 100000010 to 100009999. |
| MDI program | M01:\PRG\MDI\ | MDI.PRG | |
| Parameters [User, machine] | M01:\PRM\ | ALL.PRM | The header (1st line) is different from the M700 series.<br>An M700 file can be used with M800, but an M800 file cannot be used with the M800 series. |
| Auxiliary axis parameter | M01:\PRM\ | AUXAXIS.PRM | |
| DeviceNet parameter file | M01:\PRM\ | DEVICENT.PRM | |
| Rotary axis geometric deviation parameter file | M01:\PRM\ | GEOMETRY.PRM | |
| Safety parameter file | M01:\PRM\ | SAFEPARA.BIN | |
| System files for maintenance and service | M01:\PRM\ | SYSTEM.PRM | This cannot be accessed by the user. |
| PLC program file | M01:\LAD\ | USERPLC.LAD | Not compatible with M700 series. |
| PLC program file for each project | M01:\LAD\ | PROJECTxx.LAD | xx01 to usable project numbers |
| Own station safety PLC program file | M01:\LAD\ | SAFEPLC1.LAD | |
| Other station safety PLC program file | M01:\LAD\ | SAFEPLC2.LAD | |
| Workpiece offset data file | M01:\DAT\ | WORK.OFS | |
| Tool compensation amount data | M01:\DAT\ | TOOL.OFS | |
| Tool life management data file | M01:\DAT\ | TLIFE.TLF | |
| Common variable data file | M01:\DAT\ | COMMON.VAR | The format is different from M700 series |
| SRAM data | M01:\DAT\ | SRAM.BIN | Not compatible with M700 series. |
| Tool management data file | M01:\DAT\ | TOOLMNG.DAT | |
| SRAM open data file | M01:\DAT\ | SRAMOPEN.DAT | |
| Device open data file | M01:\DAT\ | DEVOPEN.DAT | |
| Machining surface data | M01:\DAT\ | RNAVI.DAT | |
| Tool safety data file | M01:\DAT\ | TOOLALL.DAT | |
| Machine manufacturer macro variable data file | M01:\DAT\ | MMACRO.VAR | |
| All history | M01:\LOG\ | ALLLOGLOG | The format is different from M700 series. |
| Key history data file | M01:\LOG\ | KEYLOGLOG | The format is different from M700 series. |
| Touchscreen history | M01:\LOG\ | TOUCHLOG.LOG | |
| Sampling data file | M01:\LOG\ | NCSAMP.CSV | |
| Sampling data file | M01:\LOG\ | NCSAMP.BIN | |
| PLC message data file (English) | M01:\PLCMSG\ | PLCMSG_ENG.TXT | |
| PLC message data file (Japanese) | M01:\PLCMSG\ | PLCMSG_JPN.TXT, etc. | |
| NC-side SD card | M01:\IC1\ | Any | The NC-side SD card is recognized as DS (data server) from the NC control unit. Used for data backup, storing large-capacity programs, etc. |

Table 2-3 C70 List of accessible files

| File description | Directory | File name | Remarks |
|---|---|---|---|
| Machining program | M01:\PRG\USER\ | Program No. PRG | |
| Fixed cycle program | M01:\PRG\FIX\ | Program No. PRG | |
| MDI program | M01:\PRG\MDI\ | MDI.PRG | |
| Parameters [User, machine] | M01:\PRM\ | ALL.PRM | |
| PLC program file | 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 |
| Sampling data | M01:\LOG\ | NCSAMP.CSV | For maintenance |
| Operation history data | M01:\LOG\ | TRACE.TRC | For maintenance |

⚠ DANGER Cautions for writing to a file

Carefully check a file before writing to the file for the NC control unit. Writing to an incorrect file may cause unexpected operation, resulting in a serious accident.

## 2.11.11 IEZNcFile6::CloseFile2 Close file

□ **Custom call procedure**

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

□ **Automation call procedure**

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

| | |
|---|---|
| □ **Argument** | *plRet*: Returns an error code. (Upon automation, the return value is used.)<br>**S_OK**: Normal termination<br>**EZNC_FILE_WRITEFILE_WRITE:** File is not writable |

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Closes a file. When the file was opened with **OpenFile3()**, make sure to close it with **CloseFile2()** (or **AbortFile2()**).

(Note) Do not perform this operation during automatic operation of the NC control unit. (C70 only) For the M700/M800 series, the operation can be performed during automatic operation of the NC control unit, unless automatic operation of the file intended for operation is being carried out.

□ **Reference**

**OpenFile3(), AbortFile2(), ReadFile2(), WriteFile()**

□ **Specification**

## 2.11.12 IEZNcFile6::AbortFile2 Force close file

□ **Custom call procedure**

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

□ **Automation call procedure**

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

|  |  |  |
|---|---|---|
| □ **Argument** | *plRet*: Returns an error code. (Upon automation, the return value is used.)<br>**S_OK**: Normal termination | |
| □ **Return value** | Value | Meaning |
| | **S_OK** | Normal termination |
| | **S_FALSE** | Communication failure |
| □ **Function** | Forcibly closes an open file. Use this to stop writing. After writing is stopped, the file which was being written to will be deleted.<br>The difference from **CloseFile2()** is that an error is not output. | |
| □ **Reference** | **OpenFile3(), CloseFile2(), ReadFile2(), WriteFile()** | |
| □ **Specification** | | |

## 2.11.13 IEZNcFile6::ReadFile2 Read file

□ **Custom call procedure**

```
HRESULT      ReadFile2(
                     DWORD dwLength,                  // (I) Size of data to be read
                     BYTE** ppbData,                  // (O) Read data
                     DWORD* pdwNumRead,               // (O) Read data size
                     LONG* plRet                      // (O) Error code
                     )
```

□ **Automation call procedure**

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

□ **Argument**

*dwLength*: Sets the size of data to be read at a time in the number of bytes.

*ppbData*: Returns the pointer for the read byte data array. As the read data area is allocated in this product, the client needs to release it explicitly with **CoTaskMemFree()**.

*pdwNumRead:* Returns the number of bytes that were actually read. In automation call, the **VARIANT** data includes the number of bytes.

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

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_FILE_READFILE_NOTOPEN**: No file is open in the read mode
**EZNC_FILE_READFILE_READ**: File is not readable
**EZNC_FILE_READFILE_CREATE**: Temporary file cannot be created

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Reads data from the file opened with **OpenFile3()**. Data to be read returns a byte data array and its number of bytes. Determines as the end of file when *pdwNumRead* is smaller than *dwLength*. Setting the size of data to be read at a time. When reading a large file, it can be read in multiple parts. The file can be read in sequence until **CloseFile2()** is executed.

□ **Reference**

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

□ **Specification**

C70 M700 M800

## 2.11.14 IEZNcFile6::WriteFile Write file

□ **Custom call procedure**

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

□ **Automation call procedure**

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

□ **Argument**

*dwLength*: Sets the size of data that is written at a time in the number of bytes.

*pbData*: Sets data to be written as byte array.

Automation argument:
*vData*: Creates data to be written as a byte array and sets it by substituting it in *vData* (VARIANT type) as shown in the example below.

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

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_FILE_WRITEFILE_NOTOPEN**: No file is open in write mode
**EZNC_FILE_WRITEFILE_WRITE**: Cannot be written to a file
**EZNC_FILE_LENGTH:** Invalid write data size

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Writes data to a file opened with **OpenFile3()**. Data to be written is data in a byte array.
Sets the size of data to be written at a time. When writing a large amount of data, it can be written in multiple parts. Data can be written in sequence until **CloseFile2()** is executed.
(Note 1) When a file in the NC control unit except for a machining program is changed, extra care should be taken because it may cause the NC control unit malfunction. Make sure to back up in advance to restore to its original state.

□ **Reference**

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

□ **Specification**

DANGER

Cautions for writing to a file

Carefully check a file before writing to the file for the NC control unit. Writing to an incorrect file may cause unexpected operation, resulting in a serious accident.

## 2.11.15 IEZNcFile6::OpenNCFile2 Open machining program dedicated file

□ **Custom call procedure**

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

□ **Automation call procedure**

```
           File_OpenNCFile2 (
                 bstrFileName As STRING           // (I) File name containing a path
                 lMode As LONG                    // (I) Open mode
                 ) As LONG                        // (O) Error code
```

□ **Argument**

*lpcwszFileName*: Sets the file name containing a path as a **UNICODE** character string.
A file is set with an absolute path as follows:
Drive name + ":" + \Directory name\File name

Paths other than those shown below cannot be used.

| Model | Machining program |
|---|---|
| **M700 series** | M01:\PRG\USER\Machining Program No.<br>M01:\PRG\UMACRO\Machining Program No.<br>M01:\PRG\MMACRO\Machining Program No.<br>M01:\PRG\FIX\Machining Program No.<br>M01:\PRG\MDI\Machining Program No. |
| **M800 series** | M01:\PRG\USER\Machining Program Name (32 or less alphanumeric characters including extension)<br>M01:\PRG\MMACRO\Machining Program No. (100010000 –199999999)<br>M01:\PRG\FIX\Machining Program No. (100000010 –100009999)<br>M01:\PRG\MDI\MDI.PRG |

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

| Value | Meaning |
|---|---|
| **EZNC_FILE_READ** | Read mode |
| **EZNC_FILE_WRITE** | Write mode |
| **EZNC_FILE_OVERWRITE** | Overwrite mode (writes even if the specified file exists.) |

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_FILE_OPEN_ALREADYOPENED**: File is already open
**EZNC_FILE_OPEN_FILEEXIST**: File already exists (in write mode)
**EZNC_FILE_OPEN_MODE**: Invalid open mode
**EZNC_FILE_OPEN_NOTOPEN**: File cannot be opened
**EZNC_FILE_OPEN_CREATE**: File cannot be created (in write mode)
**EZNC_FILE_OPEN_ILLEGALPATH:** Invalid path
**EZNC_FILE_OPEN_FILENOTEXIST**: File does not exist
**EZNC_FILE_OPEN_OPEN**: File cannot be opened
**EZNC_FILE_DIR_ALREADYOPENED**: A different directory is already opened
**EZ_ERR_NOT_SUPPORT**: Not supported

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Opens the machining program file in the set mode. The directory that creates a temporary file is created in the following order of priority:

- Directory specified with environment variable TMP
- Directory to which the product is installed

The temporary file name is "MELDASn". A number is placed in n.
**OpenFile3()** cannot be used concurrently.
**This function is not supported with C70. (EZ_ERR_NOT_SUPPORT is returned to plRet.)**

(Note 1) Make sure to close the open file with **CloseNCFile2()** (or **AbortNCFile2()**). If **CloseNCFile2()** is not used, a temporary file will remain.

(Note 2)For the M700/M800 series, write or overwrite operation can be performed during automatic operation of the NC control unit, unless automatic operation of the file intended for operation is being carried out. Read operation can be performed during automatic operation of the NC control unit.

□ **Reference**

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

□ **Specifica-tion**

## 2.11.16 IEZNcFile6::CloseNCFile2 Close machining program dedicated file

□ **Custom call procedure**

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

□ **Automation call procedure**

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

□ **Argument**

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_FILE_WRITEFILE_WRITE:** File is not writable
**EZ_ERR_NOT_SUPPORT**: Not supported

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Closes the machining program file.
This function is not supported with C70. (EZ_ERR_NOT_SUPPORT is returned to plRet.)

(Note 1) For the M700/M800 series, the operation can be performed during automatic operation of the NC control unit, unless automatic operation of the file intended for operation is being carried out.
(Note 2) When the file was opened with **OpenNCFile2()**, make sure to close it with **CloseNCFile2()** (or **AborNCtFile2()**).

□ **Reference**

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

□ **Specifica-tion**

M700 M800

## 2.11.17 IEZNcFile6::AbortNCFile2 Force close machining program dedicated file

**□ Custom call procedure**

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

**□ Automation call procedure**

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

**□ Argument**

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZ_ERR_NOT_SUPPORT**: Not supported

**□ Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

**□ Function**

Force closes the open machining program file. Use this to stop writing. After writing is stopped, the file which was being written to will be deleted.
The difference from **CloseNCFile2()** is that an error is not output.
This function is not supported with C70. (EZ_ERR_NOT_SUPPORT is returned to plRet.)

## 2.11.18 IEZNcFile6::ReadNCFile2 Read machining program dedicated file

□ **Custom call procedure**

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

□ **Automation call procedure**

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

□ **Argument**

*dwLength*: Setss the size of data to be read at a time in the number of bytes.

*ppbData*: Returns the pointer for the read byte data array. As the read data area is allocated in this product, the client needs to release it explicitly with **CoTaskMemFree()**.

*pdwNumRead:* Returns the number of bytes that were actually read. In automation call, the **VARIANT** data includes the number of bytes.

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

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_FILE_READFILE_NOTOPEN**: No file is open in the read mode
**EZNC_FILE_READFILE_READ**: File is not readable
**EZNC_FILE_READFILE_CREATE**: Temporary file cannot be created
**EZ_ERR_NOT_SUPPORT**: Not supported

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Data is read from the machining program file opened with **OpenNCFile2()**. Data to be read returns a byte data array and its number of bytes. Determines as the end of file when *pdwNumRead* is smaller than *dwLength*.
Sets the size of data to be read at a time. When reading a large file, it can be read in multiple parts. The file can be read in sequence until CloseNCFile2() is executed.
This function is not supported with C70. (EZ_ERR_NOT_SUPPORT is returned to plRet.)

□ **Reference**

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

□ **Specification**

## 2.11.19 IEZNcFile6::WriteNCFile Write machining program dedicated file

□ **Custom call procedure**

```
HRESULT    WriteNCFile(
                DWORD dwLength,              // (I) Size of data to be written
                BYTE* pbData,                // (I) Data to be written
                LONG* plRet                  // (O) Error code
                )
```

□ **Automation call procedure**

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

□ **Argument**

*dwLength*: Sets the size of data that is written at a time in the number of bytes.

*pbData*: Sets data to be written as byte array. In automation call, vData includes the number of bytes.

Automation argument:
*vData*: Creates data to be written as a byte array and sets it by substituting it in *vData* (VARIANT type) as shown in the example below.

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

*plRet*: Returns an error code. (Upon automation, the return value is used.)

**S_OK**: Normal termination
**EZNC_FILE_WRITEFILE_NOTOPEN**: No file is open in write mode
**EZNC_FILE_WRITEFILE_WRITE**: Cannot be written to a file
**EZNC_FILE_LENGTH:** Invalid write data size
**EZ_ERR_NOT_SUPPORT**: Not supported

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Data is written to the machining program file opened with **OpenNCFile2()**. Data to be written is data in a byte array.
Sets the size of data to be written at a time. When writing a large amount of data, it can be written in multiple parts. Data can be written in sequence until **CloseNCFile2()** is executed.
This function is not supported with C70. (EZ_ERR_NOT_SUPPORT is returned to plRet.)
(Note) For the M700/M800 series, when edit lock B (#8105) parameter is 1, programs 8000 to 9999 cannot be written. When edit lock C (#1121) parameter is 1, programs 9000 to 9999 cannot be written.

□ **Reference**

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

□ **Specification**

⚠DANGER Cautions for writing to a file

Carefully check a file before writing to the file for the NC control unit. Writing to an incorrect file may cause unexpected operation, resulting in a serious accident.

**C70** **M700** **M800**

## 2.12.1 IEZNcCommonVariable2::CommonVRead — Read common variables

□ **Custom call procedure**

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

□ **Automation call procedure**

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

□ **Argument**

*lIndex*: Sets the common variable number to be read.

Value:

(For C70)

100 to 199, 500 to 999

(For M700/M800 series)

100 to 199、400 to 999, 100100 to 100199, 200100 to 200199, 300100 to 300199
400100 to 400199, 500100 to 500199, 600100 to 600199
700100 to 700199, 800100 to 800199, 900000 to 907399

*pdData*: Returns the common variable value of the set common variable number.

*plType:* Returns the variable value type. (For the **M700/M800 series**, enabled.)

| Value | Meaning |
|---|---|
| **1** | Numerical value |
| **0** | Not set |

*plRet*: Returns an error code. (Upon automation, the return value is used.)

**S_OK**: Normal termination
**EZNC_DATA_READ_ADDR**: Invalid system specification
**EZNC_DATA_READ_READ**: Data is not readable

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Reads common variables. There is limit to the common variable number that can be handled, depending on specifications for the number of sets for common variables. For common variables #100 to #199, system specification is necessary.

□ **Reference**

**CommonVWrite(), GetSize()**

□ **Specification**

(System common variables #100 to #199 only)

## 2.12.2 IEZNcCommonVariable2::CommonVWrite Write common variables

□ **Custom call procedure**

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

□ **Automation call procedure**

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

□ **Argument**

*lIndex*: Set the common variable number to be written.
Value:
(For C70)
100 to 199, 500 to 999
(For M700/M800 series)
100 to 199, 400 to 999, 100100 to 100199, 200100 to 200199、300100 to 300199
400100 to 400199, 500100 to 500199, 600100 to 600199
700100 to 700199, 800100 to 800199, 900000 to 907399

*dData:* Sets the common variable value to be written to the set common variable number.

*lType*: Specifies the type. (For the **M700/M800**, enabled.)

| Value | Meaning |
|---|---|
| **1** | Numerical value |
| **0** | Not set |

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_WRITE_ADDR**: Invalid part system setting
**EZNC_DATA_WRITE_WRITE**: Data is not writable

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Writes common variables. There is limit to the common variable number that can be handled, depending on specifications for the number of sets for common variables. For common variables #100 to #199, part system setting is necessary.

□ **Reference**

**CommonVRead(), GetSize()**

□ **Specification**

(System common variables #100 to #199 only)

## 2.12.3 IEZNcCommonVariable2::GetSize Get number of sets for common variables

□ **Custom call procedure**

```
HRESULT         GetSize(
                        LONG lType,                     // (I) Common variable type
                        LONG* plData,                   // (O) Number of sets
                        LONG* plRet                     // (O) Error code
                        )
```

□ **Automation call procedure**

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

□ **Argument**

*lType*: Specifies the common variable type to be read.

| Value | Meaning |
|---|---|
| **0** | When the number of sets of common variables #100 and greater is got. |
| **1** | When the number of sets of common variables #500 and greater is got. |

*plData*: Returns the number of sets of common variable type.
Value meaning: 40 = 40 [sets]

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_ADDR**: Invalid part system setting
**EZNC_DATA_READ_READ**: Data is not readable

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Reads the number of sets of common variables. For common variables #100 to #199, part system setting is necessary.

□ **Reference**

**CommonVRead(), CommonVWrite()**

□ **Specification**

(System common variables #100 to #199 only)

## 2.12.4 IEZNcCommonVariable2::GetName Get names of common variables

□ **Custom call procedure**

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

□ **Automation call procedure**

```
            CommonVarialbe_GetName(
                lIndex As LONG              // (I) Common variable number
                lppwszName As STRING*       // (O) Common variable name character string
                ) As LONG                   // (O) Error code
```

□ **Argument**

*lIndex*: Sets the common variable number to be read.
Value:
(For M700 series and C70)
500 to 519
(For M800 series)
500 to 599

*lppwszName*: Returns the common variable name as a **UNICODE** character string.
Names are seven alphanumeric characters starting with an alphabet letter. The character string ends with **a NULL** code.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_READ**: Data is not readable
**EZNC_DATA_READ_DATASIZE**: Exceeded maximum data size

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Reads common variable names. Names are seven alphanumeric characters starting with an alphabet letter. The character string ends with **a NULL** code.
As the character string area memory is allocated in this product, the client using VC++ needs to release the character string area memory explicitly with **CoTaskMemFree()**.

□ **Reference**

**CommonVRead(), CommonVWrite()**

□ **Specification**

## 2.12.5 IEZNcCommonVariable2::SetName — Set name settings for common variables

□ **Custom call procedure**

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

□ **Automation call procedure**

```
            CommonVariable_SetName(
                    lIndex As LONG                // (I) Common variable number
                    lpcwszName As STRING          // (I) Common variable name character string
                    ) As LONG                     // (O) Error code
```

□ **Argument**

*lIndex*: Sets the common variable number to be written.
Value:
(For M700 series and C70)
500 to 519
(For M800 series)
500 to 599

lpcwszName: Set the common variable name as a **UNICODE** character string.
Names are seven alphanumeric characters starting with an alphabet letter. The character string ends with **a NULL** code.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_WRITE_WRITE**: Data is not writable

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Writes common variable names. Names are seven alphanumeric characters starting with an alphabet letter. The character string ends with **a NULL** code.

□ **Reference**

**GetName()**

□ **Specification**

## 2.12.6 IEZNcCommonVariable2::GetCVNullData Get value when no numerical value is set

□ **Custom call procedure**

```
HRESULT     GetCVNullData(
                 DOUBLE* pdData,      // (O) Value when no numerical value is set
                 LONG* plRet          // (O) Error code
                 )
```

□ **Automation call procedure**

```
            CommonVariable_GetNullData(
                 pdData As DOUBLE*    // (O) Value when no numerical value is set
                 ) As LONG            // (O) Error code
```

□ **Argument**

*pdData*: Returns the value when no numerical value is set.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_READ**: Data is not readable

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets the value when no numerical value of variable (#100 to 199, #500 to 519) is set.

□ **Reference**

□ **Specifica-tion**

**C70** **M700** **M800**

## 2.13.1 IEZNcLocalVariable2::LocalVRead — Read local variable

□ **Custom call procedure**

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

□ **Automation call procedure**

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

□ **Argument**

*lIndex*: Sets the common variable number to be read.
Value: 1 to 33

*lLevel*: Sets the macro subprogram execution level.
Value: 0 to 4

*pdData*: Returns the local variable value of the set local variable number of the set system.

*plType:* Returns the type. (Unused)

| Value | Meaning |
|---|---|
| **1** | Numerical value |
| **0** | Not set |

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_ADDR**: Invalid part system setting
**EZNC_DATA_READ_READ**: Data is not readable

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Reads the local variable value of the specified system.

□ **Reference**

**GetMacroLevel()**

□ **Specification**

System

## 2.13.2 IEZNcLocalVariable2::GetMacroLevel Get macro subprogram call level

**□ Custom call procedure**

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

**□ Automation call procedure**

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

**□ Argument**

*plData*: Returns the macro subprogram call level.
 Value: 0 to 8

*plRet*: Returns an error code. (Upon automation, the return value is used.)
 **S_OK**: Normal termination
 **EZNC_DATA_READ_ADDR**: Invalid part system setting
 **EZNC_DATA_READ_READ**: Data is not readable

**□ Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

**□ Function**

Gets the macro subprogram call level.

**□ Reference**

**□ Specification**

System

## 2.13.3 IEZNcLocalVariable2::GetLVNullData Get value when no numerical value is set

□ **Custom call procedure**

```
HRESULT      GetLVNullData(
                      DOUBLE* pdData,      // (O) Value when no numerical value is set
                      LONG* plRet          // (O) Error code
                              )
```

□ **Automation call procedure**

```
             LocalVariable_GetNullData(
                      pdData As DOUBLE*    // (O) Value when no numerical value is set
                      ) As LONG            // (O) Error code
```

□ **Argument**

*pdData*: Returns the value when no numerical value is set.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_READ**: Data is not readable

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets the value when no numerical value of variable (#1 to 33) is set.

□ **Reference**

□ **Specification**

**C70** **M700** **M800**

## 2.14.1 IEZNcTool3::GetToolSetSize Get number of sets for tool offset

□ **Custom call procedure**

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

□ **Automation call procedure**

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

□ **Argument**

*plSize*: Returns the number of sets for tool offset of the set part system.
The number of sets is determined by NC specifications.
Value meaning: 200 = 200 [sets]

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_READ**: Data is not readable

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets the number of sets for tool offset of the set part system. The number of sets is determined by NC specifications.

□ **Reference**

**GetType()**

□ **Specification**

System

## 2.14.2 IEZNcTool3::GetType Get tool offset type

□ **Custom call procedure**

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

□ **Automation call procedure**

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

□ **Argument**

*plType*: Returns the tool offset type of the set 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 axises compensation amount |

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_READ**: Data is not readable

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets the tool offset type of the set part system.

□ **Reference**

**GetToolSetSize()**

□ **Specification**

System

## 2.14.3 IEZNcTool3::GetOffset Get tool offset amount

□ **Custom call procedure**

```
HRESULT        GetOffset(
                        LONG lType,            // (I) Tool offset type
                        LONG lKind,            // (I) Offset amount type
                        LONG lToolSetNo,       // (I) Tool set number
                        DOUBLE* pdOffset,      // (O) Offset amount
                        LONG* plNo,            // (O) Hypothetical tool nose pointnumber
                        LONG* plRet            // (O) Error code
                        )
```

□ **Automation call procedure**

```
               Tool_GetOffset(
                        lType As LONG          // (I) Tool offset type
                        lKind As LONG          // (I) Offset amount type
                        lToolSetNo As LONG     // (I) Tool set number
                        pdOffset As DOUBLE*    // (O) Offset amount
                        plNo As LONG*          // (O) Hypothetical tool nose point number
                        ) As LONG              // (O) Error code
```

□ **Argument**

*lType*: Sets the tool offset type. Refer to the parameter table.

*lKind*: Sets the type of tool offset amount. Refer to the parameter table.

*lToolSetNo*: Set the tool offset set number.
The number of sets can be got with **GetToolSetSize()**.

*pdOffset*: Returns the tool offset amount. Refer to the parameter table.

*plNo*: Returns the hypothetical tool nose point number. Refer to the parameter table.
L system type only. Returns none except for the L system type.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_READ**: Data is not readable
**EZNC_DATA_READ_DATATYPE**: Invalid data type
**EZNC_DATA_READ_SUBSECT**: Invalid subsection number
**EZ_ERR_NOT_SUPPORT**: Not supported

Parameter table

| Value: Type | Value: Type of tool offset amount | Data range |
|---|---|---|
| 1: M system type I | 0: Tool offset amount | -99,999.999 to 99,999.999 [mm] |
| 4: M system type II | 0: Tool length compensation amount (dimensions)<br>1: (Wear)<br>2: Tool radius compensation amount (dimensions)<br>3: (Wear) | -99,999.999 to 99,999.999 [mm] |
| 6: L system type | 0: Tool length compensation amount X<br>1: Z<br>2: C (Y*) | C70 : -99.999 to 99.999 [mm]<br>M700/M800 series : -99,999.999 to 99,999.999 [mm] |
| | 3: Tool length offset amount X<br>4: Z<br>5: C (Y*) | C70: -999.999 to 999.999 [mm]<br>M700/M800 series : -99,999.999 to 99,999.999 [mm] |
| | 6: Tool radius (nose R) R | C70 : 0 to 99.999 [mm]<br>M700/M800 series : 0 to 99,999.999 [mm] |
| | 7: Tool radius (nose R) wear amount r | C70 : 0 to 99.999 [mm]<br>M700/M800 series : 0 to 99,999.999 [mm] |
| | 8: Hypothetical tool nose point number P | 0 to 8 (Refer to Figure 1) |

* For the M700/M800 series

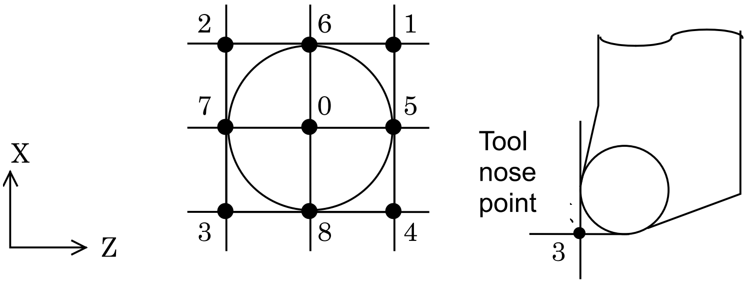


Figure 1 Hypothetical tool nose point number

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets the tool offset amount of the set part system/axis No. The range shown in the parameter table varies depending on the command unit such as inch system and metric system of the Mitsubishi CNC. For details, refer to the installation manual of each Mitsubishi CNC.

□ **Reference**

**GetType(), SetOffset(), GetToolSetSize()**

□ **Specification**

System

C70 M700 M800

## 2.14.4 IEZNcTool3::GetOffset2 Get tool offset amount

□ **Custom call procedure**

```
HRESULT      GetOffset2(
                      LONG lType,           // (I) Tool offset type
                      LONG lKind,           // (I) Offset amount type
                      LONG lToolSetNo,      // (I) Tool set number
                      DOUBLE* pdOffset,     // (O) Offset amount
                      LONG* plNo,           // (O) Hypothetical tool nose point number
                      LONG* plRet           // (O) Error code
                      )
```

□ **Automation call procedure**

```
             Tool_GetOffset2(
                      lType As LONG         // (I) Tool offset type
                      lKind As LONG         // (I) Offset amount type
                      lToolSetNo As LONG    // (I) Tool set number
                      pdOffset As DOUBLE*   // (O) Offset amount
                      plNo As LONG*         // (O) Hypothetical tool nose point number
                      ) As LONG             // (O) Error code
```

□ **Argument**

*lType*: Sets the tool offset type. Refer to the parameter table.

*lKind*: Sets the type of tool offset amount. Refer to the parameter table.

*lToolSetNo*: Sets the tool offset set number.
The number of sets can get with **GetToolSetSize()**.

*pdOffset*: Returns the tool offset amount. Refer to the parameter table.

*plNo*: Returns the Hypothetical tool nose point number. Refer to the parameter table.
L system type only. Returns none except for the L system type.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_READ**: Data is not readable
**EZNC_DATA_READ_DATATYPE**: Invalid data type
**EZNC_DATA_READ_SUBSECT**: Invalid subsection number

Parameter table

| Value: Type | Value: Type of tool offset amount | Data range |
|---|---|---|
| 1: M system type I | 0: Tool offset amount | -99,999.999 to 99,999.999 [mm] |
| 4: M system type II | 0: Tool length compensation amount (dimensions)<br>1: (Wear)<br>2: Tool radius compensation amount (dimensions)<br>3: (Wear) | -99,999.999 to 99,999.999 [mm] |
| 6: L system type | 0: Tool length compensation amount<br>1: Z<br>2: C (Y*) | C70 : -99.999 to 99.999 [mm]<br>M700/M800 series : -99,999.999 to 99,999.999 [mm] |
| | 3: Tool length offset amount X<br>4: Z<br>5: C (Y*) | C70: -999.999 to 999.999 [mm]<br>M700/M800 series : -99,999.999 to 99,999.999 [mm] |
| | 6: Tool radius (nose R) R | C70 : 0 to 99.999 [mm]<br>M700/M800 series : 0 to 99,999.999 [mm] |
| | 7: Tool radius (nose R) wear amount r | C70 : 0 to 99.999 [mm]<br>M700/M800 series : 0 to 99,999.999 [mm] |
| | 8: Hypothetical tool nose point number P | 0 to 8 (Refer to Figure 1) |

* For the M700/M800 series

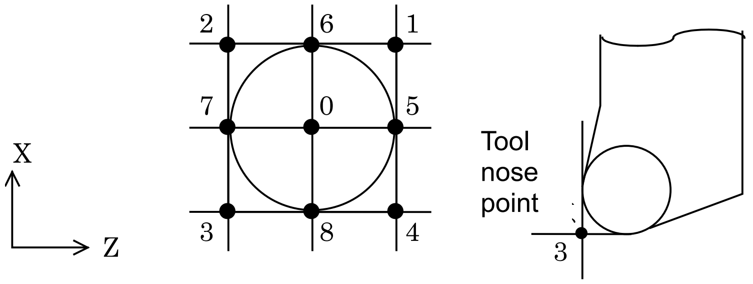


Figure 1 Hypothetical tool nose point

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets the tool offset amount of the set part system/axis No. The range shown in the parameter table varies depending on the command unit such as inch system and metrik system of the Mitsubishi CNC. For details, refer to the installation manual of each Mitsubishi CNC.

□ **Reference**

**GetType(), SetOffset(), GetToolSetSize()**

□ **Specification**

System

## 2.14.5 IEZNcTool3::SetOffset Set tool offset amount settings

□ **Custom call procedure**

```
HRESULT      SetOffset(
                   LONG lType,          // (I) Tool offset type
                   LONG lKind,          // (I) Offset amount type
                   LONG lToolSetNo,     // (I) Tool set number
                   DOUBLE dOffset,      // (I) Offset amount
                   LONG lNo,            // (I) Hypothetical tool nose point number
                   LONG* plRet          // (O) Error code
                   )
```

□ **Automation call procedure**

```
             Tool_SetOffset(
                   lType As LONG        // (I) Tool offset type
                   lKind As LONG        // (I) Offset amount type
                   lToolSetNo As LONG   // (I) Tool set number
                   dOffset As DOUBLE    // (I) Offset amount
                   lNo As LONG          // (I) Hypothetical tool nose point point number
                   ) As LONG            // (O) Error code
```

□ **Argument**

*lType*: Sets the tool offset type. Refer to the parameter table.

*lKind*: Sets the type of tool offset amount. Refer to the parameter table.

*lToolSetNo*: Sets the tool offset set number.
The number of sets can get with **GetToolSetSize()**.

*dOffset*: Sets the tool offset amount. Refer to the parameter table.

*lNo*: Sets the hypothetical tool nose point number. Refer to the parameter table.
L system type only. Disabled for the M system type.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_WRITE_WRITE**: Data is not writable
**EZNC_DATA_WRITE_DATATYPE**: Invalid data type
**EZNC_DATA_WRITE_SUBSECT**: Invalid subsection number

Parameter table

| Value: Type | Value: Type of tool offset amount | Data range |
|---|---|---|
| 1: M system type I | 0: Tool offset amount | -99,999.999 to 99,999.999 [mm] |
| 4: M system type II | 0: Tool length compensation amount (dimensions)<br>1: (Wear)<br>2: Tool radius compensation amount (dimensions)<br>3: (Wear) | -99,999.999 to 99,999.999 [mm] |
| 6: L system type | 0: Tool length compensation amount X<br>1: Z<br>2: C (Y*) | C70 : -99.999 to 99.999 [mm]<br>M700/M800 series : -99,999.999 to 99,999.999 [mm] |
| | 3: Tool length offset amount X<br>4: Z<br>5: C (Y*) | C70 : -999.999 to 999.999 [mm]<br>M700/M800 series : -99,999.999 to 99,999.999 [mm] |
| | 6: Tool radius (nose R) R<br>7: Tool radius (nose R) wear amount r<br>8: Hypothetical tool nose point number P | C70 : 0 to 99.999 [mm]<br>M700/M800 series : 0 to 99,999.999 [mm]<br>C70 : 0 to 99.999 [mm]<br>M700/M800 series : 0 to 99,999.999 [mm]<br>0 to 8 (Refer to Figure 1) |

* For the M700/M800 series

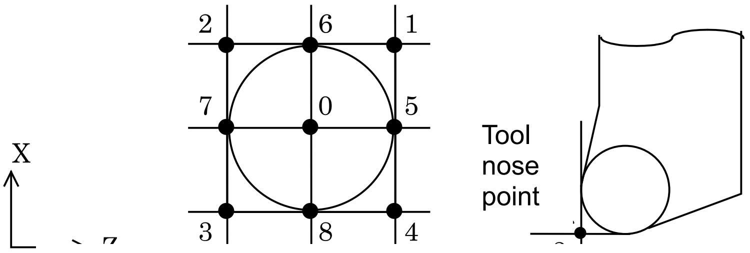


Figure 1 Hypothetical tool nose point number

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Configures the tool offset amount of the set part system/axis No. The range shown in the parameter table varies depending on the command unit such as inch system and metrik system of the Mitsubishi CNC. For details, refer to the installation manual of each Mitsubishi CNC.

□ **Reference**

**GetType(), GetOffset(), GetToolSetSize()**

□ **Specification**

System

## 2.14.6 IEZNcTool3::GetToolWorkOffset Get workpiece coordinate offset

□ **Custom call procedure**

```
HRESULT       GetToolWorkOffset(
                   LONG lAxisNo,              // (I) Axis No.
                   LONG lIndex,               // (I) Workpiece coordinate system number
                   DOUBLE* pdOffset,          // (O) Offset value
                   LONG* plRet                // (O) Error code
                   )
```

□ **Automation call procedure**

```
              Tool_GetToolWorkOffset(
                   lAxisNo As LONG            // (I) Axis No.
                   lIndex As LONG             // (I) Workpiece coordinate system number
                   pdOffset As DOUBLE*        // (O) Offset value
                   ) As LONG                  // (O) Error code
```

□ **Argument**

*lAxisNo*: Sets the axis. (From Axis 1 = from **1**)

*lIndex*: Sets the workpiece coordinate system number to be 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 offset value of the workpiece coordinate of the set part system/axis No.
Value: -99,999.999～99,999.999 [mm]

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_ADDR**: Invalid part system setting
**EZNC_DATA_READ_READ**: Data is not readable
**EZNC_DATA_READ_AXIS**: Invalid axis No. setting
**EZ_ERR_NOT_SUPPORT**: Not supported

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets the offset value of the workpiece coordinate of the set part system/axis No. For details, refer to the installation guide.
Refer to the instruction manual for each numerical controller for details.

□ **Reference**

□ **Specification**

System , Axis number

## 2.14.7 IEZNcTool3::GetToolWorkOffset2 Get workpiece coordinate offset

□ **Custom call procedure**

```
HRESULT     GetToolWorkOffset2(
                 LONG lAxisNo,            // (I) Axis No.
                 LONG lIndex,             // (I) Workpiece coordinate system number
                 DOUBLE* pdOffset,        // (O) Offset value
                 LONG* plRet              // (O) Error code
                 )
```

□ **Automation call procedure**

```
            Tool_GetToolWorkOffset2(
                 lAxisNo As LONG          // (I) Axis No.
                 lIndex As LONG           // (I) Workpiece coordinate system number
                 pdOffset As DOUBLE*      // (O) Offset value
                 ) As LONG                // (O) Error code
```

□ **Argument**

*lAxisNo*: Sets the axis No. (From Axis 1 = from **1**)

*lIndex*: Sets the workpiece coordinate system number to be 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 offset value of the workpiece coordinate of the set part system/axis No.
Value: -99,999.999～99,999.999 [mm]

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_ADDR**: Invalid parft system setting
**EZNC_DATA_READ_READ**: Data is not readable
**EZNC_DATA_READ_AXIS**: Invalid axis No. setting

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets the offset value of the workpiece coordinate of the set part system/axis No. For details, refer to the installation guide.
Refer to the instruction manual for each numerical controller for details.

Setting mode
(1) For setting absolute value,
Sets the set offset value as the current offset value.
(2) For setting additional value,
Sets the offset value got by adding the set offset value to the current offset value.

□ **Reference**

□ **Specification**

System , Axis number

## 2.14.8 IEZNcTool3:: SetToolWorkOffset — Set workpiece coordinate offset settings

□ **Custom call procedure**

```
HRESULT      SetToolWorkOffset(
                     LONG lAxisNo,              // (I) Axis No.
                     LONG lIndex,               // (I) Workpiece coordinate system number
                     DOUBLE dOffset,            // (I) Offset value
                     LONG lMode,                // (I) Mode
                     LONG* plRet                // (O) Error code
                     )
```

□ **Automation call procedure**

```
             Tool_SetToolWorkOffset(
                     lAxisNo As LONG            // (I) Axis No.
                     lIndex As LONG             // (I) Workpiece coordinate system number
                     dOffset As DOUBLE          // (I) Offset value
                     lMode As LONG              // (I) Mode
                     ) As LONG                  // (O) Error code
```

□ **Argument**

*lAxisNo*: Sets the axis No. (From Axis 1 = from **1**)

*lIndex*: Sets the workpiece coordinate system number to be 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*: Sets the offset value of the workpiece coordinate of the set part system/axis No.
Value: -99,999.999～99,999.999 [mm]

*lMode*: Sets the setting mode (absolute value setting/additional value setting).

| Value | Meaning |
|---|---|
| **0** | Sets absolute value |
| **1** | Sets additional value |

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_WRITE_ADDR**: Invalid part system setting
**EZNC_DATA_WRITE_WRITE**: Data is not writable
**EZNC_DATA_WRITE_AXIS**: Invalid axis No. setting

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

| | |
|---|---|
| □<br>**Function** | Sets the offset value of the workpiece coordinate of the set part system/axis No.<br>For details, refer to the instruction manual for each Mitsubishi CNC.<br><br>Setting mode<br>(1) For setting absolute value,<br>Sets the set offset value as the current offset value.<br>(2) For setting additional value,<br>Sets the offset value got by adding the set offset value to the current offset value. |
| □<br>**Reference** | |
| □<br>**Specification** | System , Axis number |

C70 M700 M800

## 2.14.9 IEZNcTool3:: SetToolWorkOffset9 Set workpiece coordinate offset settings

**□ Custom call procedure**

```
HRESULT       SetToolWorkOffset2(
                        LONG lAxisNo,               // (I) Axis No.
                        LONG lIndex,                // (I) Workpiece coordinate system number
                        DOUBLE dOffset,             // (I) Offset value
                        LONG lMode,                 // (I) Mode
                        LONG* plRet                 // (O) Error code
                        )
```

**□ Automation call procedure**

```
              Tool_SetToolWorkOffset2(
                        lAxisNo As LONG             // (I) Axis No.
                        lIndex As LONG              // (I) Workpiece coordinate system number
                        dOffset As DOUBLE           // (I) Offset value
                        lMode As LONG               // (I) Mode
                        ) As LONG                   // (O) Error code
```

**□ Argument**

*lAxisNo*: Sets the axis No. (From Axis 1 = from **1**)

*lIndex*: Sets the workpiece coordinate system number to be 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*: Sets the offset value of the workpiece coordinate of the set part system/axis No.
Value: -99,999.999～99,999.999 [mm]

*lMode*: Sets the setting mode (absolute value setting/additional value setting).

| Value | Meaning |
|---|---|
| **0** | Sets absolute value |
| **1** | Sets additional value |

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_WRITE_ADDR**: Invalid part system setting
**EZNC_DATA_WRITE_WRITE**: Data is not writable
**EZNC_DATA_WRITE_AXIS**: Invalid axisNo. setting

**□ Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Configures the offset value of the workpiece coordinate of the set part system/axis No.
Refer to the instruction manual for each Mitsubishi CNC for details.

Setting mode
(1) For setting absolute value,
   Sets the set offset value as the current offset value.
(2) For setting additional value,
   Sets the offset value got by adding the set offset value to the current offset value.

□ **Reference**

□ **Specification**

System , Axis number

## 2.14.10 IEZNcTool3::GetSurface Get reference surface height

□ **Custom call procedure**

```
HRESULT     GetSurface(
                  LONG lAxisNo,          // (I) Axis No.
                  DOUBLE* pdHight,       // (O) Reference surface height
                  LONG* plRet            // (O) Error code
                  )
```

□ **Automation call procedure**

```
            Tool_GetSurface(
                  lAxisNo As LONG        // (I) Axis No.
                  pdHight As DOUBLE*     // (O) Reference surface height
                  ) As LONG              // (O) Error code
```

| | |
|---|---|
| □ **Argument** | *lAxisNo*: Sets the axis No. (From Axis 1 = from **1**)<br>*pdHight*: Returns the reference surface coordinate position of the tool length measurement II of the set part system/axis No.<br>Value: -99999.999 to 99999.999 [mm]<br>*plRet*: Returns an error code. (Upon automation, the return value is used.)<br>**S_OK**: Normal termination<br>**EZNC_DATA_READ_READ**: Data is not readable<br>**EZNC_DATA_READ_ADDR**: Invalid part system, axis No. setting<br>**EZNC_DATA_READ_AXIS**: Invalid axis No. setting |

| □ **Return value** | Value | Meaning |
|---|---|---|
| | **S_OK** | Normal termination |
| | **S_FALSE** | Communication failure |

| | |
|---|---|
| □ **Function** | Gets the reference surface coordinate position of the tool length measurement II of the set part system/axis No. |
| □ **Reference** | |
| □ **Specification** | System , Axis number |

C70 M700 M800

## 2.14.11 IEZNcTool3::GetSurface2 Get reference surface height

□ **Custom call procedure**

```
HRESULT        GetSurface2(
                       LONG lAxisNo,            // (I) Axis No.
                       DOUBLE* pdHight,         // (O) Reference surface height
                       LONG* plRet              // (O) Error code
                       )
```

□ **Automation call procedure**

```
               Tool_GetSurface2(
                       lAxisNo As LONG          // (I) Axis No.
                       pdHight As DOUBLE*       // (O) Reference surface height
                       ) As LONG                // (O) Error code
```

□ **Argument**

*lAxisNo*: Sets the axis No. (From Axis 1 = from **1**)

*pdHight*: Returns the reference surface coordinate position of the tool length measurement II of the set part system/axis No.
Value: -99,999.999 to 99,999.999 [mm]

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_READ**: Data is not readable
**EZNC_DATA_READ_ADDR**: Invalid part system, axis No. setting
**EZNC_DATA_READ_AXIS**: Invalid axis No. setting

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets the reference surface coordinate position of the tool length measurement II of the set axis No. of the set part system.

□ **Reference**

□ **Specification**

System , Axis number

C70 M700 M800

## 2.14.12 IEZNcTool3::SetSurface — Set reference surface height settings

□ **Custom call procedure**

```
HRESULT       SetSurface(
                      LONG lAxisNo,              // (I) Axis No.
                      DOUBLE dHight,             // (I) Reference surface height
                      LONG* plRet                // (O) Error code
                      )
```

□ **Automation call procedure**

```
              Tool_SetSurface(
                      lAxisNo As LONG            // (I) Axis No.
                      dHight As DOUBLE*          // (O) Reference surface height
                      ) As LONG                  // (O) Error code
```

□ **Argument**

*lAxisNo*: Sets the axis No. (From Axis 1 = from **1**)

*dHight*: Sets the reference surface coordinate position of the tool length measurement II of the set part system/axis No.
Value: -99999.999 to 99999.999 [mm]

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_WRITE_WRITE**: Data is not writable
**EZNC_DATA_WRITE_ADDR**: Invalid part system, axis No. setting
**EZNC_DATA_WRITE_AXIS**: Invalid axis No. setting

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Sets the reference surface coordinate position of the tool length measurement II of the set part system/axis No.

□ **Reference**

□ **Specification**

System , Axis number

## 2.14.13 IEZNcTool3::GetToolLifeType2 Get tool life management method

□ **Custom call procedure**

```
HRESULT     GetToolLifeType2(
                 LONG* plType,          // (O) Tool life management method
                 LONG* plRet            // (O) Error code
                 )
```

□ **Automation call procedure**

```
            Tool_GetToolLifeType2(
                 plType As LONG*        // (O) Tool life management method
                 ) As LONG              // (O) Error code
```

□ **Argument**

*plType*: Returns the tool life control type.

| Value | Meaning |
|---|---|
| **0** | Disabled |
| **1** | Type **I** |
| **2** | Type **II** |

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_READ**: Data is not readable
**EZ_ERR_DATA_RANGE**: Invalid argument data range

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets tool life management method.

□ **Reference**

**SetToolLifeType2()**

□ **Specification**

## 2.14.14 IEZNcTool3::SetToolLifeType2 Select Tool life management method

**□ Custom call procedure**

```
HRESULT    SetToolLifeType2(
                    LONG lType,          // (I) Tool life management method
                    LONG* plRet          // (O) Error code
                    )
```

**□ Automation call procedure**

```
           Tool_SetToolLifeType2(
                    lType As LONG        // (O) Tool life management method
                    ) As LONG            // (O) Error code
```

**□ Argument**

*lType*: Sets the tool life management method.

| Value | Meaning |
|---|---|
| **0** | Disabled (For the C70, cannot be specified.) |
| **1** | Tool life management **I** |
| **2** | Tool life management **II** |

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_WRITE_WRITE**: Data is not writable
**EZ_ERR_DATA_RANGE**: Invalid argument data range
**EZ_ERR_DATA_TYPE**: Invalid argument data type
**EZ_ERR_NOT_SUPPORT**: Not supported

**□ Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

**□ Function**

Selects tool life management **I** or **II**. Write is inhibited by password mode. (**EZNC_DATA_WRITE_WRITE** is returned.)

**□ Reference**

**GetToolLifeType2()**

**□ Specification**

## 2.14.15 IEZNcTool3::GetToolLifeGroupList Get Tool Life management group No list

**□ Custom call procedure**

```
HRESULT       GetToolLifeGroupList (
                      LPDWORD lpdwLength,        // (O) Number of groups
                      LPDWORD* lppdwGroup,       // (O) Array of group numbers
                      LONG* plRet                // (O) Error code
                      )
```

**□ Automation call procedure**

```
              Tool_GetToolLifeGroupList(
                      pvGroup As VARIANT*        // (O) Group number
                      ) As LONG                  // (O) Error code
```

**□ Argument**

*lpdwLength*: Returns the number of sets of groups.

*lppdwGroup*: Returns the list of group numbers as an array. As the group number array is allocated in this product, the client needs to release it explicitly with **CoTaskMemFree()**.

Automation argument:
*pvGroup*: Returns the group number array as **VARIANT**.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_READ**: Data is not readable
**EZ_ERR_MEMORY_ALLOC**: Memory cannot be allocated.
**EZ_ERR_DATA_RANGE**: Invalid argument data range
**EZNC_DATA_READ_ADDR**: Invalid part system setting
**EZNC_DATA_TLFGROUP_ADDR**: Invalid address (system specification)
**EZNC_DATA_TLFGROUP_OUTOFSPEC**: The set group number is out of specifications
**EZ_ERR_NOT_SUPPORT**: Not supported

**□ Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

**□ Function**

Gets tool life management group No.

**□ Reference**

**AddToolLifeGroup(), ChangeToolLifeGroup(), DeleteToolLifeGroup()**

**□ Specification**

System (M700/M800 series only)

## 2.14.16 IEZNcTool3::ChangeToolLifeGroup Change tool life management group No

**□ Custom call procedure**

```
HRESULT     ChangeToolLifeGroup (
                      DWORD dwSrcGroup,     // (I) Old group number
                      DWORD dwDstGroup,     // (I) New group number
                      LONG* plRet           // (O) Error code
                      )
```

**□ Automation call procedure**

```
            Tool_ChangeToolLifeGroup(
                      lSrcGroup As LONG     // (I) Old group number
                      lDstGroup As LONG     // (I) New group number
                      ) As LONG             // (O) Error code
```

**□ Argument**

*dwSrcGroup:* Sets the old group number.

*dwDstGroup:* Sets the new group number.

Automation argument:
*lSrcGroup*: Refer to the explanation of *dwSrcGroup*.
*lDstGroup*: Refer to the explanation of *dwDstGroup*.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_WRITE_WRITE**: Data is not writable.
**EZNC_DATA_NOT_EXIST**: Data does not exist
**EZ_ERR_DATA_RANGE**: Invalid argument data range
**EZNC_DATA_READ_ADDR**: Invalid part system setting
**EZNC_DATA_READ_READ**: Data is not readable
**EZNC_DATA_TLFGROUP_ADDR**: Invalid address (system specification)
**EZNC_DATA_TLFGROUP_EXIST:** Group number already exists
**EZNC_DATA_TLFGROUP_NONEXIST**: Group number does not exist
**EZNC_DATA_TLFGROUP_OUTOFSPEC**: The set group number is out of specifications
**EZ_ERR_NOT_SUPPORT**: Not supported

**□ Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

**□ Function**

Changes the set group number to the new group number.
The set old group tool is changed to the new group.

**□ Reference**

**GetToolLifeGroupList(), AddToolLifeGroup(), DeleteToolLifeGroup()**

**□ Specification**

System (M700/M800 series only)

C70 M700 M800

## 2.14.17 IEZNcTool3::DeleteToolLifeGroup — Delete tool life management group No

□ **Custom call procedure**

```
HRESULT     DeleteToolLifeGroup (
                    DWORD dwGroup,          // (I) Group number
                    LONG* plRet             // (O) Error code
                    )
```

□ **Automation call procedure**

```
            Tool_DeleteToolLifeGroup(
                    lGroup As LONG          // (I) Group number
                    ) As LONG               // (O) Error code
```

□ **Argument**

*dwGroup*: Sets the group number to be deleted.

Automation argument:
*lGroup*: Refer to the explanation of *dwGroup*.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_WRITE_WRITE**: Data is not writable.
**EZNC_DATA_NOT_EXIST**: Data does not exist
**EZ_ERR_DATA_RANGE**: Invalid argument data range
**EZNC_DATA_READ_ADDR**: Invalid part system setting
**EZNC_DATA_TLFGROUP_ADDR**: Invalid address (part system setting)
**EZNC_DATA_TLFGROUP_NONEXIST**: Group number does not exist
**EZNC_DATA_TLFGROUP_OUTOFSPEC**: The set group number is out of specifications
**EZ_ERR_NOT_SUPPORT**: Not supported

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Deletes the group number.

□ **Reference**

**GetToolLifeGroupList(), AddToolLifeGroup(), ChangeToolLifeGroup()**

□ **Specification**

System

## 2.14.18 IEZNcTool3::GetToolLifeToolNoList Get list of tool numbers within Life management group

□ **Custom call procedure**

```
HRESULT     GetToolLifeToolNoList (
                   DWORD dwGroup,              // (I) Group number
                   LPDWORD lpdwLength,         // (O) Number of registered tools
                   LPDWORD *lppdwToolNo,       // (O) Array of tool numbers
                   LONG* plRet                 // (O) Error code
                   )
```

□ **Automation call procedure**

```
            Tool_ GetToolLifeToolNoList (
                   lGroup As LONG              // (I) Group number
                   pvToolNo As VARIANT         // (O) Array of tool numbers
                   ) As LONG                   // (O) Error code
```

□ **Argument**

*dwGroup*: Sets the group number for which the list of tool numbers is got.

*lpdwLength:* Returns the number of registered tools included in the group (array length of the list of tool numbers).

*lppdwToolNo:* Returns the list of tool numbers included in the group as an array. As the tool number list array is allocated in this product, the client needs to release it explicitly with **CoTaskMemFree()**.

Automation argument:
*lGroup*: Refer to the explanation of *dwGroup*.
*pvToolNo:* Returns the list of tool numbers included in the group as VARIANT.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_READ**: Data is not readable
**EZ_ERR_MEMORY_ALLOC**: Memory cannot be allocated
**EZ_ERR_DATA_RANGE**: Invalid argument data range
**EZNC_DATA_TLFGROUP_ADDR**: Invalid address (system specification)
**EZNC_DATA_TLFGROUP_NONEXIST**: Group number does not exist
**EZNC_DATA_TLFGROUP_OUTOFSPEC**: The set group number is out of specifications
**EZ_ERR_NOT_SUPPORT**: Not supported

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets the list of tool numbers of the set group.

□ **Reference**

**AddToolLifeToolNo(), ChangeToolLifeToolNo(), DeleteToolLifeToolNo()**

□ **Specification**

System

## 2.14.19 IEZNcTool3::AddToolLifeToolNo Add tool number to tool life management group No

□ **Custom call procedure**

```
HRESULT        AddToolLifeToolNo (
                         DWORD dwGroup,              // (I) Group number
                         DWORD dwToolNo,             // (I) Tool number
                         LONG* plRet                 // (O) Error code
                         )
```

□ **Automation call procedure**

```
               Tool_ AddToolLifeToolNo (
                         lGroup As LONG              // (I) Group number
                         lToolNo As LONG             // (I) Tool number
                         ) As LONG                   // (O) Error code
```

□ **Argument**

*dwGroup*: Sets the group numbers to which tool numbers are added.

*dwToolNo:* Sets the tool numbers to be added.

Automation argument:
*lGroup*: Refer to the explanation of *dwGroup*.
*lToolNo:* Refer to the explanation of *dwToolNo*.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_WRITE_WRITE**: Data is not writable
**EZNC_DATA_DUPLICATE:** Duplicated numbers
**EZ_ERR_DATA_RANGE**: Invalid argument data range
**EZNC_DATA_TLFGROUP_ADDR**: Invalid address (system specification)
**EZNC_DATA_TLFGROUP_OUTOFSPEC**: The set group number is out of specifications
**EZNC_DATA_TLFTOOL_EXIST:** Tool number already exists
**EZNC_DATA_TLFTOOL_OUTOFSPEC**: Set tool number is out of specifications
**EZ_ERR_NOT_SUPPORT**: Not supported

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Adds the tool numbers to the specified group.

□ **Reference**

**GetToolLifeToolNoList(), ChangeToolLifeToolNo(), DeleteToolLifeToolNo()**

□ **Specification**

System

## 2.14.20 IEZNcTool3::ChangeToolLifeToolNo Change tool life management group No

□ **Custom call procedure**

```
HRESULT     ChangeToolLifeToolNo(
                     DWORD dwGroup,              // (I) Group number
                     DWORD dwSrcToolNo,          // (I) Old tool number
                     DWORD dwDstToolNo,          // (I) New tool number
                     LONG* plRet                 // (O) Error code
                     )
```

□ **Automation call procedure**

```
            Tool_ ChangeToolLifeToolNo (
                     lGroup As LONG              // (I) Group number
                     lSrcToolNo As LONG          // (I) Old tool number
                     lDstToolNo As LONG          // (I) New tool number
                     ) As LONG                   // (O) Error code
```

□ **Argument**

*dwGroup*: Sets the group number in which the tool number is changed.

*dwSrcToolNo*: Sets the old tool number.

*dwDstToolNo*: Sets the new tool number.

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. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_WRITE_WRITE**: Data is not writable
**EZNC_DATA_NOT_EXIST**: Data does not exist
**EZ_ERR_DATA_RANGE**: Invalid argument data range
**EZNC_DATA_READ_READ**: Data is not readable
**EZNC_DATA_TLFGROUP_ADDR**: Invalid address (system specification)
**EZNC_DATA_TLFGROUP_NONEXIST**: Group number does not exist
**EZNC_DATA_TLFGROUP_OUTOFSPEC**: The set group number is out of specifications
**EZNC_DATA_TLFTOOL_EXIST:** Tool number already exists
**EZNC_DATA_TLFTOOL_NONEXIST:** Tool number does not exist
**EZNC_DATA_TLFTOOL_OUTOFSPEC**: Set tool number is out of specifications
**EZ_ERR_NOT_SUPPORT**: Not supported

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Changes the set tool number to the new tool number.

□ **Reference**

**GetToolLifeToolNoList(), AddToolLifeToolNo(), DeleteToolLifeToolNo()**

□ **Specification**

System

## 2.14.21 IEZNcTool3::DeleteToolLifeToolNo Delete tool life management tool number

□ **Custom call procedure**

```
HRESULT     DeleteToolLifeToolNo (
                DWORD dwGroup,          // (I) Group number
                DWORD dwToolNo,         // (I) Tool number
                LONG* plRet             // (O) Error code
                )
```

□ **Automation call procedure**

```
            Tool_ DeleteToolLifeToolNo (
                lGroup As LONG          // (I) Group number
                lToolNo As LONG         // (I) Tool number
                ) As LONG               // (O) Error code
```

□ **Argument**

*dwGroup*: Sets the group number from which tool numbers are deleted.

*dwToolNo:* Sets the tool numbers to be deleted.

Automation argument:
*lGroup*: Refer to the explanation of *dwGroup*.
*lToolNo:* Refer to the explanation of *dwToolNo*.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_WRITE_WRITE**: Data is not writable
**EZNC_DATA_NOT_EXIST**: Data does not exist
**EZ_ERR_DATA_RANGE**: Invalid argument data range
**EZNC_DATA_READ_READ**: Data is not readable
**EZNC_DATA_TLFGROUP_ADDR**: Invalid address (system specification)
**EZNC_DATA_TLFGROUP_NONEXIST**: Group number does not exist
**EZNC_DATA_TLFGROUP_OUTOFSPEC**: The set group number is out of specifications
**EZNC_DATA_TLFTOOL_NONEXIST:** Tool number does not exist
**EZNC_DATA_TLFTOOL_OUTOFSPEC**: Set tool number is out of specifications
**EZ_ERR_NOT_SUPPORT**: Not supported

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Deletes the set tool numbers.

□ **Reference**

**GetToolLifeToolNoList(), AddToolLifeToolNo(), ChangeToolLifeToolNo()**

□ **Specification**

System

C70 M700 M800

## 2.14.22 IEZNcTool3::GetToolLifeValue Get tool life management data

□ **Custom call procedure**

```
HRESULT     GetToolLifeValue (
                    DWORD dwGroup,             // (I) Group number
                    DWORD dwToolNo,            // (I) Tool number
                    LPOLESTR** lpppwszData,    // (O) Tool life management data value
                                                        character string array
                    LONG* plRet                // (O) Error code
                    )
```

□ **Automation call procedure**

```
            Tool_GetToolLifeValue (
                    lGroup As LONG             // (I) Group number
                    lToolNo As LONG            // (I) Tool number
                    pvData As VARIANT*         // (O) Tool life management data value
                                                        character string
                    ) As LONG                  // (O) Error code
```

□ **Argument**

*dwGroup*: Sets the group number for which tool life is got.

*dwToolNo*: Sets the tool number for which tool life is got.

*lpppwszData*: Returns the life management data value as a **UNICODE** character string array. As the data value array is allocated in this product, the client needs to release it explicitly with **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 life control data value (UNICODE character string) array as VARIANT. For life control data values, refer to the index.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_READ**: Data is not readable
**EZNC_DATA_NOT_EXIST**: Data does not exist
**EZ_ERR_MEMORY_ALLOC**: Memory cannot be allocated
**EZ_ERR_DATA_RANGE**: Invalid argument data range
**EZNC_DATA_TLFGROUP_ADDR**: Invalid address (system specification)
**EZNC_DATA_TLFGROUP_NONEXIST**: Group number does not exist
**EZNC_DATA_TLFGROUP_OUTOFSPEC**: The set group number is out of specifications
**EZNC_DATA_TLFTOOL_NONEXIST:** Tool number does not exist
**EZNC_DATA_TLFTOOL_OUTOFSPEC**: Set tool number is out of specifications
**EZ_ERR_NOT_SUPPORT**: Not supported

Index table

| Array Index | Tool life type (data range) | |
|---|---|---|
| | C70 M system | M700/M800 series M system |
| 0 | Tool number (1 to 99999999) | Tool number (1 to 99999999) |
| 1 | Status (depends on MTB specifications) | Status (0x00 to 0xFF) |
| 2 | Method (000 to 222)* | Method (000 to 222)* |
| 3 | Length dimension (±1 to 99999.999) | Length dimension (±9999.999) |
| 4 | Radius dimension (±1 to 99999.999) | Radius dimension (±9999.999) |
| 5 | Life (Time: 0 to 4000, Count: 0 to 65000) | Life (Time: 0 to 4000, Count: 0 to 65000) |
| 6 | Used (Time: 0 to 4000,Count: 0 to 65000) | Used (Time: 0 to 4000, Count: 0 to 65000) |
| 7 | Auxiliary (0 to 65535, depends on MTBr specifications) | Auxiliary (0 to 65535, depends on MTB specifications) |
| 8 | Length wear (Reserved: 0) | Length wear (Reserved: 0) |
| 9 | Radius wear (Reserved: 0) | Radius wear (Reserved: 0) |
| 10 | Group (1 to 99999999) | Group (1 to 99999999) |

| Array Index | Tool life type (data range) | |
|---|---|---|
| | M700/M800 series L system (TYPE I),C70 L system (TYPE I) | M700/M800 series L system (TYPE II) |
| 0 | Application of time management (0 to 995959) | Tool number (1 to 999999) |
| 1 | Application of number of times management (0 to 9999) | Compensation number (0 to 80) |
| 2 | Status A (0 to 2) | Usage (Time: 0 to 99999999, Count: 0 to 999999) |
| 3 | Life of time management (0 to 995959) | ST (0 to 3) |
| 4 | Life of count management (0 to 9999) | Method (Time: 0, Count: 1) |
| 5 | Status B (depends on MTB specifications) | Life (0 to 999999) |
| 6 to 10 | - | - |

| Array Index | Tool life type (data range) | |
|---|---|---|
| | C70 L system (TYPE II) | |
| 0 | Tool umber (1 to 999999) | |
| 1 | Group (1 to 9999) | |
| 2 | Method (0: Time, 1: Count) | |
| 3 | Compensation number (1 to 80) | |
| 4 | Status (0 to 3) | |
| 5 | Life (Time: 0 to 999999, Count: 0 to 999999) | |
| 6 | Used (Time: 0 to 999999,Count: 0 to 999999) | |
| 7 to 10 | - | |

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets life control data for the set tool number. Note that the number of elements of the character string array that returns life control data varies depending on models.

* For the "method" for tool life management data, refer to the installation manual of each Mitsubishi CNC.

□ **Reference**

**SetToolLifeValue()**

□ **Specification**

System

## 2.14.23 IEZNcTool3::SetToolLifeValue Set individual tool life management data

**□ Custom call procedure**

```
HRESULT      SetToolLifeValue (
                     DWORD dwGroup,               // (I) Group number
                     DWORD dwToolNo,              // (I) Tool number
                     DWORD dwKind,                // (I) Type of tool life management data
                     LPCOLESTR lpcwszData,        // (I) Tool life management
                     LONG* plRet                  // (O) Error code
                     )
```

**□ Automation call procedure**

```
             Tool_ SetToolLifeValue (
                     lGroup As LONG               // (I) Group number
                     lToolNo As LONG              // (I) Tool number
                     lKind As LONG                // (I) Type of tool life management l data
                     lpcwszData As STRING         // (I) Tool life management data
                     ) As LONG                    // (O) Error code
```

**□ Argument**

*dwGroup*: Sets the group number for which tool life is set.

*dwToolNo*: Sets the tool number for which tool life is set.

*dwKind*: Sets the type of tool life. Refer to the parameter table.

*lpcwszData:* Sets the specified type of life data.

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. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_WRITE_WRITE**: Data is not writable
**EZNC_DATA_NOT_EXIST**: Data does not exist
**EZNC_DATA_READ_READ**: Data is not readable
**EZ_ERR_DATA_RANGE**: Invalid argument data range (*dwKind*)
**EZ_ERR_DATA_TYPE**: Invalid argument data type
**EZNC_DATA_TLFGROUP_ADDR**: Invalid address (system specification)
**EZNC_DATA_TLFGROUP_NONEXIST**: Group number does not exist
**EZNC_DATA_TLFGROUP_OUTOFSPEC**: The set group number is out of specifications
**EZNC_DATA_TLFTOOL_NONEXIST:** Tool number does not exist
**EZNC_DATA_TLFTOOL_PARAMERR**: Invalid type specified for life control data
**EZNC_DATA_TLFTOOL_MAXMINERR**: Setting data is out of range
**EZNC_DATA_TLFTOOL_OUTOFSPEC**: Set tool number is out of specifications
**EZ_ERR_NOT_SUPPORT**: Not supported

Parameter table

| Value | Tool life type (data range) | |
|---|---|---|
| | C70 M system | M700/M800 series M system |
| 1 | Tool number (1 to 99999999) | Tool number (1 to 99999999) |
| 2 | Status (depends on machine manufacturer specifications) | Status (0x00 to 0xFF) |
| 3 | Method (000 to 222)* | Method (000 to 222)* |
| 4 | Length dimension (±99999.999) | Length dimension (±9999.999)* |
| 5 | Radius dimension (±99999.999) | Radius dimension (±9999.999)* |
| 6 | Life (Time: 0 to 4000, Count: 0 to 9999) | Life (Time: 0 to 4000, Count: 0 to 9999/65000) |
| 7 | Used (Time: 0 to 4000,Count: 0 to 9999) | Used (Time: 0 to 4000,Count: 0 to 9999/65000) |
| 8 | Auxiliary (0 to 65535, depends on MTB specifications) | Auxiliary (0 to 65535, depends on MTB specifications) |
| 9 | Length wear (Reserved: 0) | Length wear (Reserved: 0) |
| 10 | Radius wear (Reserved: 0) | Radius wear (Reserved: 0) |
| 11 | Group (1 to 99999999) | Group (1 to 99999999) |

| Value | Tool life type (data range) | |
|---|---|---|
| | M700/M800 series L system (TYPE I), C70 L system (TYPE I) | M700/M800 series L system (TYPE II) |
| 1 | Application of time management (0 to 995959) | Tool number (1 to 999999) |
| 2 | Application of count management (0 to 9999) | Compensation number (0 to 80) |
| 3 | Status A (0 to 2) | Used (Time: 0 to 99999999, Count: 0 to 999999) |
| 4 | Life of time management (0 to 995959) | ST (0 to 3) |
| 5 | Life of count management (0 to 9999) | Method (Time: 0, Count: 1) |
| 6 | Status B (depends on MTB specifications) | Life (0 to 999999) |

| Value | Tool life type (data range) | |
|---|---|---|
| | C70 L system (TYPE II) | |
| 1 | Tool number (1 to 999999) | |
| 2 | Group (1 to 9999) | |
| 3 | Method (0: Time, 1: Count) | |
| 4 | Compensation number (1 to 80) | |
| 5 | Status (0 to 3) | |
| 6 | Life (Time: 0 to 999999, Count: 0 to 999999) | |
| 7 | Used (Time: 0 to 999999, Count: 0 to 999999) | |

**□ Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

**□ Function**

Individually sets life control data for the set tool number. This method is used for updating. To newly add a tool, use the following procedure.

1) AddToolLifeGroup( )
2) AddToolLifeToolNo( )
3) SetToolLifeValue( )

* For the "method" for tool life management data, refer to the installation guide of each Mitsubishi CNC.

**□ Reference**

**GetToolLifeValue(), AddToolLifeGroup(), AddToolLifeToolNo()**

**□ Specifica-tion**

System

## 2.14.24 IEZNcTool3::SetToolLifeValue2 Set tool life management data

□ **Custom call procedure**

```
HRESULT      SetToolLifeValue2 (
                  DWORD dwGroup,              // (I) Group number
                  DWORD dwToolNo,             // (I) Tool number
                  LPCOLESTR* lppcwszData,     // (I) Tool life management data character
                                                     string array
                  LONG* plRet                 // (O) Error code
                  )
```

□ **Automation call procedure**

```
             Tool_ SetToolLifeValue2 (
                  lGroup As LONG              // (I) Group number
                  lToolNo As LONG             // (I) Tool number
                  vData As VARIANT            // (I) Tool life management data character
                                                     string array
                  ) As LONG                   // (O) Error code
```

□ **Argument**

*DwGroup*: Sets the group number for which tool life is set.

*dwToolNo*: Sets the tool number for which tool life is set.

*lppcwszData:* Sets a **UNICODE** character string array for the set type of life data.

Automation argument:
*lGroup*: Refer to the explanation of *dwGroup*.
*lToolNo:* Refer to the explanation of *dwToolNo*.
*vData*: Creates a **UNICODE** character string array for the specified type of life data and sets by substituting it in *vData* (VARIANT type). For examples of substitution, refer to " 2.11.13 WriteFile".

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_WRITE_WRITE**: Data is not writable
**EZNC_DATA_NOT_EXIST**: Data does not exist
**EZ_ERR_DATA_RANGE**: Invalid argument data range (*dwGroup*, *dwToolNo*)
**EZ_ERR_NULLPTR:** Argument is NULL pointer
**EZNC_DATA_TLFGROUP_ADDR**: Invalid address (system specification)
**EZNC_DATA_TLFGROUP_NONEXIST**: Group number does not exist
**EZNC_DATA_TLFGROUP_OUTOFSPEC**: The set group number is out of specifications
**EZNC_DATA_TLFTOOL_NONEXIST:** Tool number does not exist
**EZNC_DATA_TLFTOOL_MAXMINERR**: Setting data is out of range
**EZNC_DATA_TLFTOOL_OUTOFSPEC**: Set tool number is out of specifications
**EZ_ERR_NOT_SUPPORT**: Not supported

Parameter table

| Array Index | Tool life type (data range) | |
|---|---|---|
| | C70 M system | M700/M800 series M system |
| 0 | Tool number (1 to 99999999) | Tool number (Reserved: 0) |
| 1 | Status (depends on MTB specifications) | Status (0x00 to 0xFF) |
| 2 | Method (000 to 222)* | Method (000 to 222)* |
| 3 | Length dimension (±99999.999) | Length dimension (±9999.999)* |
| 4 | Radius dimension (±99999.999) | Radius dimension (±9999.999)* |
| 5 | Life (Time: 0 to 4000,Count: 0 to 9999) | Life (Time: 0 to 4000, Count: 0 to 65000) |
| 6 | Used (Time: 0 to 4000,Count: 0 to 9999) | Usage (Time: 0 to 4000, Count: 0 to 65000) |
| 7 | Auxiliary (0 to 65535, depends on MTB specifications) | Auxiliary (0 to 65535, depends on MTB specifications) |
| 8 | Length wear (Reserved: 0) | Length wear (Reserved: 0) |
| 9 | Radius wear (Reserved: 0) | Radius wear (Reserved: 0) |
| 10 | Group (1 to 99999999) | Group (Reserved: 0) |

| Array Index | Tool life type (data range) | |
|---|---|---|
| | M700/M800 series L system (TYPE I),<br>C70 L system (TYPE I) | M700/M800 series L system (TYPE II) |
| 0 | Application of time management (0 to 995959) | Tool number (Reserved: 0) |
| 1 | Application of count management (0 to 9999) | Compensation number (0 to 80) |
| 2 | Status A (0 to 2) | Usage (Time: 0 to 99999999, Number of times: 0 to 999999) |
| 3 | Life of time management (0 to 995959) | ST (0 to 3) |
| 4 | Life of count management (0 to 9999) | Method (Time: 0,Count: 1) |
| 5 | Status B (depends on MTB specifications) | Life (0 to 999999) |
| 6 to 10 | - | - |

| Array Index | Tool life type (data range) | |
|---|---|---|
| | C70 L system (TYPE II) | |
| 0 | Tool number (1 to 999999) | |
| 1 | Group (1 to 9999) | |
| 2 | Method (0: Time, 1:Count) | |
| 3 | Compensation number (1 to 80) | |
| 4 | Status (0 to 3) | |
| 5 | Life (Time: 0 to 999999, Count: 0 to 999999) | |
| 6 | Usage (Time: 0 to 999999, Count: 0 to 999999) | |
| 7 to 10 | - | |

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Sets tool life management data for the set tool number. This method is used for updating. To newly add a tool, use the following procedure.

1) AddToolLifeGroup( )
2) AddToolLifeToolNo( )
3) SetToolLifeValue2( )

For the "method" for tool life management data, refer to the installation guide of each Mitsubishi CNC.

[Example]

```
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()**

□ **Specification**

System

C70 M700 M800

## 2.15.1 IEZNcATC3::GetMGNControl Get ATC tool registration control parameter

□ **Custom call procedure**

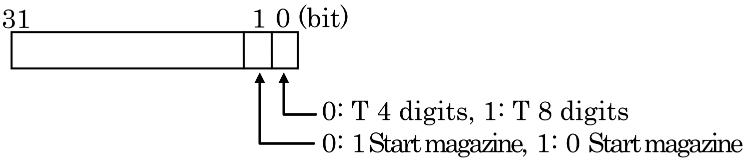
```
HRESULT     GetMGNControl(
                    LONG* plData,              // (O) Parameter value
                    LONG* plRet                // (O) Error code
                    )
```

□ **Automation call procedure**

```
            ATC_GetMGNControl(
                    plData As LONG*            // (O) Parameter value
                    ) As LONG                  // (O) Error code
```

□ **Argument**

*plData*: Returns a parameter that controls start magazine.


31
1 0 (bit)
0: T 4 digits, 1: T 8 digits
0: 1Start magazine, 1: 0 Start magazine

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_READ**: Data is not readable

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets the control parameter value for ATC tool registration.

□ **Reference**

□ **Specifica-tion**

**C70** **M700** **M800**

## 2.15.2 IEZNcATC3::GetMGNSize — Get total number of sets of magazine pots for ATC tool registration

**□ Custom call procedure**

```
HRESULT         GetMGNSize(
                        LONG* plSize,             // (O) Total number of sets of magazine pots
                        LONG* plRet               // (O) Error code
                        )
```

**□ Automation call procedure**

```
                ATC_GetMGNSize(
                        plSize As LONG*           // (O) Total number of sets of magazine pots
                        ) As LONG                 // (O) Error code
```

**□ Argument**

*plSize*: Returns the total number of sets of magazine pots.
Value: 0 to 360 (maximum)

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_READ**: Data is not readable

**□ Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

**□ Function**

Gets the total number of sets of magazine pots.

**□ Reference**

**□ Specification**

## 2.15.3 IEZNcATC3::GetMGNSize2 Get number of sets of pots for each magazine for ATC tool registration

**□ Custom call procedure**

```
HRESULT     GetMGNSize2(
                 LONG lMagazineNo,      // (I) Magazine number
                 LONG* plSize,          // (O) Number of sets of magazine pots
                 LONG* plRet            // (O) Error code
                 )
```

**□ Automation call procedure**

```
            ATC_GetMGNSize2(
                 lMagazineNo As LONG    // (I) Magazine number
                 plSize As LONG*        // (O) Number of sets of magazine pots
                 ) As LONG              // (O) Error code
```

**□ Argument**

*lMagazineNo:* Set the magazine number.
Value: 1 to 5 (maximum) M700/M800 series
Value: 1 to 3 (maximum) C70

*plSize*: Returns the number of sets of magazine pots.
Value: 0 to 360 (maximum) M700/M800 series
Value: 0 to 80 (maximum) C70

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_READ**: Data is not readable
**EZ_ERR_DATA_RANGE**: Invalid argument data range (*lMagazineNo*)

**□ Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

**□ Function**

Gets the number of sets of the set magazine pots.

**□ Reference**

**□ Specification**

## 2.15.4 IEZNcATC3::GetMGNReady2 Get tool number for ATC tool registration

**□ Custom call procedure**

```
HRESULT     GetMGNReady2(
                 LONG lMagazineNo,          // (I) Magazine number
                 LONG lReady,               // (I) On standby
                 LONG* plToolNo,            // (O) Tool number
                 LONG* plRet                // (O) Error code
                 )
```

**□ Automation call procedure**

```
            ATC_GetMGNReady2(
                 lMagazineNo As LONG        // (I) Magazine number
                 lReady As LONG             // (I) On standby
                 plToolNo As LONG*          // (O) Tool number
                 ) As LONG                  // (O) Error code
```

**□ Argument**

*lMagazineNo:* Sets the magazine number.
Value: 1 to 2 (Even if value is set for M700/M800 series, it is invalid.)

*lReady:* Sets standby state.

| Value | Meaning |
|---|---|
| **0** | Tool number to be installed |
| **1** | Tool number on standby 1 |
| **2** | Tool number on standby 2 |
| **3** | Tool number on standby 3 |
| **4** | Tool number on standby 4 |

*plToolNo:* Returns the tool number.
Value: 1 to 99999999 (maximum)

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_READ**: Data is not readable
**EZ_ERR_DATA_RANGE**: Invalid argument data range (*lMagazineNo, lReady*)

**□ Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

**□ Function**

Gets the tool number for ATC tool registration.

**□ Reference**

**□ Specification**

## 2.15.5 IEZNcATC3::GetMGNPot Get tool number for magazine pot for ATC tool registration

□ **Custom call procedure**

```
HRESULT     GetMGNPot(
                 LONG lIndex,              // (I) Magazine pot number
                 LONG* plToolNo,           // (O) Tool number
                 LONG* plRet               // (O) Error code
                 )
```

□ **Automation call procedure**

```
            ATC_GetMGNPot(
                 lIndex As LONG            // (I) Magazine pot number
                 plToolNo As LONG*         // (O) Tool number
                 ) As LONG                 // (O) Error code
```

□ **Argument**

*lIndex*: Sets the magazine pot number.
Value: 1 to 360 (maximum) M700/M800 series
Value: 1 to 80 (maximum) C70

*plToolNo:* Returns the tool number.
Value: 0 to 99999999 (maximum)

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_READ**: Data is not readable
**EZ_ERR_DATA_RANGE**: Invalid argument data range (*lIndex)*

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets the tool number which is stored in the set pot of the magazine.

□ **Reference**

**SetMGNPot(), GetMGNPotEx()**

□ **Specification**

## 2.15.6 IEZNcATC3::GetMGNPot3 Get tool number for each magazine pot for ATC tool registration

□ **Custom call procedure**

```
HRESULT      GetMGNPot3(
                  LONG lMagazineNo,           // (I) Magazine number
                  LONG lIndex,                // (I) Pot number
                  LONG* plToolNo,             // (O) Tool number
                  LONG* plRet                 // (O) Error code
                  )
```

□ **Automation call procedure**

```
             ATC_GetMGNPot3(
                  lMagazineNo As LONG         // (I) Magazine number
                  lIndex As LONG              // (I) Pot number
                  plToolNo As LONG*           // (O) Tool number
                  ) As LONG                   // (O) Error code
```

|  |  |
|---|---|
| □ **Argument** | *lMagazineNo:* Sets the magazine number.<br>Value: 1 to 5 (maximum) M700/M800 series<br>Value: 1 to 3 (maximum) C70<br><br>*lIndex*: Sets the pot number.<br>Value: 1 to 360 (maximum) M700/M800 series<br>Value: 1 to 80 (maximum) C70<br><br>*plToolNo:* Returns the tool number.<br>Value: 0 to 99999999 (maximum)<br><br>*plRet*: Returns an error code. (Upon automation, the return value is used.)<br>**S_OK**: Normal termination<br>**EZNC_DATA_READ_READ**: Data is not readable<br>**EZ_ERR_DATA_RANGE**: Invalid argument data range *(lMagazineNo, lIndex)* |

| □ **Return value** | Value | Meaning |
|---|---|---|
| | **S_OK** | Normal termination |
| | **S_FALSE** | Communication failure |

|  |  |
|---|---|
| □ **Function** | Gets the tool number which is stored in the pot of the set magazine number. |
| □ **Reference** | **SetMGNPot3()** |
| □ **Specifica-tion** |  |

## 2.15.7 IEZNcATC3::SetMGNPot Set tool number for magazine pots for ATC tool registration

□ **Custom call procedure**

```
HRESULT      SetMGNPot(
                  LONG lIndex,            // (I) Pot number
                  LONG lToolNo,           // (I) Tool number
                  LONG* plRet             // (O) Error code
                  )
```

□ **Automation call procedure**

```
             ATC_SetMGNPot(
                  lIndex As LONG          // (I) Pot number
                  lToolNo As LONG         // (I) Tool number
                  ) As LONG               // (O) Error code
```

□ **Argument**

*lIndex*: Sets the pot number.
Value: 1 to 360 (maximum) M700/M800 series
Value: 1 to 80 (maximum) C70

*lToolNo*: Sets the tool number.
Value: 1 to 99999999 (maximum)

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_WRITE_WRITE**: Data is not writable
**EZ_ERR_DATA_RANGE**: Invalid argument data range (*lIndex, lToolNo)*

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Sets the tool number to be stored in the magazine pot. Care should be taken to avoid overlaps in tool numbers because double registration of a tool number is not checked. An alarm is raised when double registration is made. Alarm information can be got with GetAlarm().
For the **M700/M800 series**, if the tool number of the Mitsubishi CNC is specified for 4 digits, the lower 4 digits are registered when the number with 5 digits or more is specified, and the 5th digit and higher will be discarded.

□ **Reference**

**GetMGNPot(), SetMGNPotEx(), GetAlarm()**

□ **Specification**

## 2.15.8 IEZNcATC3::SetMGNPot3 Set tool number for each magazine pot for ATC tool registration

□ **Custom call procedure**

```
HRESULT     SetMGNPot3(
                 LONG lMagazineNo,          // (I) Magazine number
                 LONG lIndex,               // (I) Pot number
                 LONG lToolNo,              // (I) Tool number
                 LONG* plRet                // (O) Error code
                 )
```

□ **Automation call procedure**

```
            ATC_SetMGNPot3(
                 lMagazineNo As LONG        // (I) Magazine number
                 lIndex As LONG             // (I) Pot number
                 lToolNo As LONG            // (I) Tool number
                 ) As LONG                  // (O) Error code
```

□ **Argument**

*lMagazineNo:* Sets the magazine number.
Value: 1 to 5 (maximum) M700/M800 series
Value: 1 to 3 (maximum) C70

*lIndex*: Sets the pot number.
Value: 1 to 360 (maximum) M700/M800 series
Value: 1 to 80 (maximum) C70

*lToolNo*: Sets the tool number.
Value: 1 to 99999999 (maximum)

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_WRITE_WRITE**: Data is not writable
**EZ_ERR_DATA_RANGE**: Invalid argument data range *(lMagazineNo, lIndex, lToolNo)*

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Sets the tool number to be stored in the pot of the set magazine number. Care should be taken to avoid overlaps in tool numbers because double registration of a tool number is not checked. An alarm is raised when double registration is made. Alarm information can be got with GetAlarm().
If the tool number of the Mitsubishi CNC is specified for 4 digits, the lower 4 digits are registered when the number with 5 digits or more is specified, and the 5th digit and higher will be discarded.

□ **Reference**

**GetMGNPot3(), GetAlarm()**

□ **Specification**

## 2.15.9 IEZNcATC3::GetMGNAux Get user PLC interface for ATC tool registration

□ **Custom call procedure**

```
HRESULT        GetMGNAux(
                    LONG* plData,                    // (O) Data
                    LONG* plRet                      // (O) Error code
                    )
```

□ **Automation call procedure**

```
               ATC_GetMGNAux(
                    plData As LONG*                  // (O) Data
                    ) As LONG                        // (O) Error code
```

□ **Argument**

*plData*: Returns sequence processing data for user programmable controller.
Value: 0 to 65535

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_READ**: Data is not readable

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Returns sequence processing data for user PLC.

□ **Reference**

□ **Specifica-tion**

## 2.15.10 IEZNcATC3::SetMGNAux Set user PLC interface for ATC tool registration

**□ Custom call procedure**

```
HRESULT     SetMGNAux(
                     LONG lData,                // (I) Data
                     LONG* plRet                // (O) Error code
                     )
```

**□ Automation call procedure**

```
            ATC_SetMGNAux(
                     lData As LONG              // (I) Data
                     ) As LONG                  // (O) Error code
```

**□ Argument**

*lData:* Sets sequence processing data for userPLC.
Value: 0 to 65535

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_READ**: Data is not readable

**□ Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

**□ Function**

Sets sequence processing data for user PLC.

**□ Reference**

**□ Specification**

**C70**

## 2.16.1 IEZNcParameter3::GetParameterData2 Get parameters

□ **Custom call procedure**

```
HRESULT    GetParameterData2(
                   LONG lGroup,                // (I) Group number
                   LONG lItem,                 // (I) First item number
                   LONG lSize,                 // (I) Number of items
                   LONG lAxis,                 // (I) Axis No.
                   LPOLESTR* lppwszValue,      // (O) Parameter value character string array
                   LONG* plRet                 // (O) Error code
                   )
```

□ **Automation call procedure**

```
           Parameter_GetData2(
                   lGroup As LONG              // (I) Group number
                   lItem As LONG               // (I) First item number
                   lSize As LONG               // (I) Number of items
                   lAxis As LONG               // (I) Axis No.
                   pvValue As VARIANT*         // (O) Parameter value character string array
                   ) As LONG                   // (O) Error code
```

□ **Argument**

*lGroup*: Sets the group number of parameter.

| Model | Setting |
|---|---|
| **M700 series** | Disabled |
| **C70** | Disabled |

*lItem*: Sets the first item number of parameter. This must be set. The parameter number described in setup manuals of each CNC will be this item number.

| Model | | IB# of parameter manual | Item number to be set |
|---|---|---|---|
| **M700 series** | Mitsubishi CNC M700 series | IB-1500123 | Parameter number |
| | Mitsubishi CNC M700VS series | IB-1500905 | Parameter number |
| | Mitsubishi CNC M700VW series | IB-1500932 | Parameter number |
| | Mitsubishi CNC M70 series | IB-1500878 | Parameter number |
| | Mitsubishi CNC M70V series | IB-1500957 | Parameter number |
| **C70** | | IB-1500264 | Parameter number |

*lSize*: Sets the number of items of parameter. The range starts from 1.

*lAxis*: Sets the axis No. whose parameter is to be got. This argument needs not be set unless the parameter to be got is axis-dependent.

*lppwszValue*: Gets the parameter value as a **UNICODE** character string array. Though the character string area will internally be allocated in this product, the pointer of the character string (for *lSize*) needs to be reserved by the client. As the character string area is internally allocated in this product, the client using VC++ needs to release the memory area explicitly with **CoTaskMemFree()**.
The parameter value will be got as signed value regardless of its item number. In the case of unsigned SHORT type data "65535", "-1" will be got.

*pvValue*: See the explanation of *lppwszValue*.

|  |  |  |
|---|---|---|
| □ **Argument** | *plRet*: Returns an error code. (Upon automation, the return value is used.)<br>**S_OK**: Normal termination<br>**EZNC_DATA_READ_ADDR**: Invalid part system, axis No. setting<br>**EZNC_PARAM_FILENOTEXIST**: No parameter information file<br>**EZNC_DATA_NOT_EXIST**: Data does not exist<br>**EZ_ERR_NOT_SUPPORT**: Not supported | |
| □ **Return value** | Value | Meaning |
| | **S_OK**<br>**S_FALSE** | Normal termination<br>Communication failure |
| □ **Function** | Gets the parameter.<br>This function is not supported with the M800 Series. (EZ_ERR_NOT_SUPPORT is returned to plRet.) | |
| □ **Reference** | **SetParameterData2()** | |
| □ **Specifica-tion** | | |

## 2.16.2 IEZNcParameter3::GetParameterData3 Get parameters

□ **Custom call procedure**

```
HRESULT      GetParameterData3(
                     LONG lGroup,               // (I) Group number
                     LONG lItem,                // (I) First item number
                     LONG lSize,                // (I) Number of items
                     LONG lAxis,                // (I) Aaxis No.
                     LPOLESTR* lppwszValue,     // (O) Parameter value character string array
                     LONG* plRet                // (O) Error code
                     )
```

□ **Automation call procedure**

```
             Parameter_GetData3(
                     lGroup As LONG             // (I) Group number
                     lItem As LONG              // (I) First item number
                     lSize As LONG              // (I) Number of items
                     lAxis As LONG              // (I) Axis No.
                     pvValue As VARIANT*        // (O) Parameter value character string array
                     ) As LONG                  // (O) Error code
```

□ **Argument**

*lGroup*: Sets the group number of parameter.

| Model | Setting |
|---|---|
| **C70** | Disabled |
| **M700/M800 series** | Disabled |

*lItem*: Sets the first item number of parameter. This must be set. The parameter number described in setup manuals of each CNC will be this item number.

| Model | | IB# of parameter manual | Item number to be set |
|---|---|---|---|
| **C70** | | IB-1500264 | Parameter number |
| **M700 series** | Mitsubishi CNC M700 series | IB-1500123 | Parameter number |
| | Mitsubishi CNC M700VS series | IB-1500905 | Parameter number |
| | Mitsubishi CNC M700VW series | IB-1500932 | Parameter number |
| | Mitsubishi CNC M70 series | IB-1500878 | Parameter number |
| | Mitsubishi CNC M70V series | IB-1500957 | Parameter number |
| **M800 series** | | IB-1501265 | Parameter number |

*lSize*: Sets the number of items of parameter. The range starts from 1.

*lAxis*: Sets the axis whose parameter is to be got. This argument needs not be set unless the parameter to be got is axis-dependent.

*lppwszValue*: Gets the parameter value as a **UNICODE** character string array. Though the character string area will internally be allocated in this product, the pointer of the character string (for *lSize*) needs to be reserved by the client. As the character string area is internally allocated in this product, the client using VC++ needs to release the memory area explicitly with **CoTaskMemFree()**.
The parameter value will be got as signed value regardless of its item number. In the case of unsigned SHORT type data "65535", "-1" will be got.

*pvValue*: See the explanation of *lppwszValue*.

□ **Argument**

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_ADDR**: Invalid part system, axis No. setting
**EZNC_PARAM_FILENOTEXIST**: No parameter information file
**EZNC_DATA_NOT_EXIST**: Data does not exist

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Gets the parameter.

□ **Reference**

**SetParameterData3()**

□ **Specification**

## 2.16.3 IEZNcParameter3::SetParameterData2 Set parameters

□ **Custom call procedure**

```
HRESULT     GetParameterData2(
                  LONG lGroup,                 // (I) Group number
                  LONG lItem,                  // (I) First item number
                  LONG lSize,                  // (I) Number of items
                  LONG lAxis,                  // (I) Axis No.
                  LPOLESTR* lppwszValue,       // (O) Parameter value character string array
                  LONG* plRet                  // (O) Error code
                  )
```

□ **Automation call procedure**

```
            Parameter_GetData2(
                  lGroup As LONG               // (I) Group number
                  lItem As LONG                // (I) First item number
                  lSize As LONG                // (I) Number of items
                  lAxis As LONG                // (I) Axis No.
                  pvValue As VARIANT*          // (O) Parameter value character string array
                  ) As LONG                    // (O) Error code
```

□ **Argument**

*lGroup*: Sets the group number of parameter.

| Model | Setting |
|---|---|
| **C70** | Disabled |
| **M700 series** | Disabled |

*lItem*: Sets the first item number of parameter. This must be set. The parameter number described in setup manuals of each CNC will be this item number.

| Model | | IB# of parameter manual | Item number to be set |
|---|---|---|---|
| **C70** | | IB-1500264 | Parameter number |
| **M700 series** | Mitsubishi CNC M700 series | IB-1500123 | Parameter number |
| | Mitsubishi CNC M700VS series | IB-1500905 | Parameter number |
| | Mitsubishi CNC M700VW series | IB-1500932 | Parameter number |
| | Mitsubishi CNC M70 series | IB-1500878 | Parameter number |
| | Mitsubishi CNC M70V series | IB-1500957 | Parameter number |

*lSize*: Sets the number of items of parameter. The range starts from 1.

*lAxis*: Sets the axis whose parameter is to be got. This argument needs not be set unless the parameter to be got is axis-dependent.

*lppwszValue*: Gets the parameter value as a **UNICODE** character string array.

Automation argument:
*vValue*: See the explanation of *lppwszValue*.

□ **Argument**

*plRet*: Returns an error code. (Upon automation, the return value is used.)

**S_OK**: Normal termination
**EZNC_DATA_READ_ADDR**: Invalid part system, axis No. setting
**EZNC_PARAM_FILENOTEXIST**: No parameter information file
**EZNC_DATA_NOT_EXIST**: Data does not exist
**EZ_ERR_NOT_SUPPORT**: Not supported

| □ **Return value** | Value | Meaning |
|---|---|---|
| | **S_OK**<br>**S_FALSE** | Normal termination<br>Communication failure |
| □ **Function** | Gets the parameter.<br>When setting the machine parameters, the NC must be set to the machine parameter setting mode state. Refer to the Setup Manual for each CNC for details on setting the machine parameter setting mode.<br>This function is not supported with the M800 Series. (EZ_ERR_NOT_SUPPORT is returned to plRet.) | |
| □ **Reference** | **SetParameterData2()** | |
| □ **Specifica-tion** | | |

## 2.16.4 IEZNcParameter3::SetParameterData3 Set Parameters

□ **Custom call procedure**

```
HRESULT     SetParameterData3(
                    LONG lGroup,                // (I) Group number
                    LONG lItem,                 // (I) First item number
                    LONG lSize,                 // (I) Number of items
                    LONG lAxis,                 // (I) Axis No.
                    LPCOLESTR* lppcwszValue,    // (I) Parameter value character string array
                    LONG* plRet                 // (O) Error code
                    )
```

□ **Automation call procedure**

```
            Parameter_SetData3(
                    lGroup As LONG              // (I) Group number
                    lItem As LONG               // (I) First item number
                    lAxis As LONG               // (I) Axis No.
                    vValue As VARIANT           // (I) Parameter value character string array
                    ) As LONG                   // (O) Error code
```

□ **Argument**

*lGroup*: Sets the group number of parameter.

| Model | Setting |
|---|---|
| **C70** | Disabled |
| **M700/M800 series** | Disabled |

*lItem*: Sets the first item number of parameter. This must be set. The parameter number described in setup manuals of each CNC will be this item number.

| Model | | IB# of parameter manual | Item number to be set |
|---|---|---|---|
| **C70** | | IB-1500264 | Parameter number |
| **M700 series** | Mitsubishi CNC M700 series | IB-1500123 | Parameter number |
| | Mitsubishi CNC M700VS series | IB-1500905 | Parameter number |
| | Mitsubishi CNC M700VW series | IB-1500932 | Parameter number |
| | Mitsubishi CNC M70 series | IB-1500878 | Parameter number |
| | Mitsubishi CNC M70V series | IB-1500957 | Parameter number |
| **M800 series** | | IB-1501265 | Parameter number |

*lSize*: Sets the number of items of parameter. The range starts from 1.

*lAxis*: Sets the axis whose parameter is to be got. This argument needs not be set unless the parameter to be got is axis-dependent.

*lppcwszValue*: Sets the parameter value as a **UNICODE** character string array.

*vValue*: See the explanation of *lppcwszValue*.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_WRITE_ADDR**: Invalid part system, axis No.setting
**EZNC_PARAM_FILENOTEXIST**: No parameter information file
**EZNC_DATA_NOT_EXIST**: Data does not exist

| | | |
|---|---|---|
| □ **Return value** | Value | Meaning |
| | **S_OK** | Normal termination |
| | **S_FALSE** | Communication failure |
| □ **Function** | Sets the parameter.<br>To set a machine parameter, the NC needs to be in the machine parameter setting mode. For how to set the NC to the machine parameter setting mode, see the setup manual of the Mitsubishi CNC. | |
| □ **Reference** | **GetParameterData3()** | |
| □ **Specification** | | |

| C70 | M700 | M800 |
|---|---|---|

## 2.17.1 IEZNcOperation::Search Operation search

□ **Custom call procedure**

```
HRESULT      Search(
                     LPCOLESTR lpcwszSelectProgram,      // (I) Program file name
                     LONG lSequenceNo,                   // (I) Sequence number
                     LONG lBlockNo,                      // (I) Block number
                     LONG* plRet                         // (O) Error code
                     )
```

□ **Automation call procedure**

```
             Operation_Search(
                     lpcwszSelectProgram As STRING       // (I) Program file name
                     lSequenceNo As LONG                 // (I) Sequence number
                     lBlockNo As LONG                    // (I) Block number
                     ) As LONG                           // (O) Error code
```

□ **Argument**

*lpcwszSelectProgram*: Sets the program file name for operation search as a **UNICODE** character string.

*lSequenceNo*: Sets the sequence number to be searched.

*lBlockNo*: Sets the block number to be searched.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_OPE_SELECTPRG_ADDR: Invalid system specification**
**EZNC_OPE_SELECTPRG_FILESYSTEM**: File system error
**EZNC_OPE_SELECTPRG_NOPRG**: No program file
**EZNC_OPE_SELECTPRG_PRGFORMAT**: Invalid program file name format
**EZNC_OPE_SELECTPRG_RUNNING**: Program operation in progress

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Executes operation search.
Use *lpwcszSelectProgram* to set the program file name whose operation is to be started.
The name of the program files in the \PRG\USER\ directory of NC control unit or in the M□:\IC1\ directory of NC's CF card or NC's SD card (M800S/M80: at the front panel SD card slot, M800W: at the rear panel SD card slot #2) can be set.
Note that the file name of the program file in the \PRG\USER\ directory of NC control unit does not need to contain the drive name and directory path.

Use a character string as below to specify the program file name.
C70: "<program number>.PRG"  Example) "1000.PRG"
M700/M800 series: "<program file name>"  Example) "1000"
M700/M800 series: "M□:\IC1\<program file name>"  Example) "M01:\IC1\1000"
*lSequenceNo* or *lBlockNo can be used to set the sequence number or block number whose operation is to be started.* Set 0 in *lSequenceNo* and *lBlockNo when operation is to be started from the top of the program.*

□ **Reference**

**IEZNcProgram**::**CurrentBlockRead()**

□ **Specification**

System

## 2.17.2 IEZNcOperation::Run Start PLC program

□ **Custom call procedure**

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

□ **Automation call procedure**

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

| | |
|---|---|
| □ **Argument** | *plRet*: Returns an error code. (Upon automation, the return value is used.)<br>**S_OK**: Normal termination<br>**EZNC_OPE_ACTPLC_ADDR**: Invalid NC control unit |

| □ **Return value** | Value | Meaning |
|---|---|---|
| | **S_OK** | Normal termination |
| | **S_FALSE** | Communication failure |

| | |
|---|---|
| □ **Function** | Starts the PLC program. |
| □ **Reference** | **Stop()** |
| □ **Specifica-tion** | |

## 2.17.3 IEZNcOperation::Stop Stop PLC program

□ **Custom call procedure**

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

□ **Automation call procedure**

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

| | | |
|---|---|---|
| □ **Argument** | *plRet*: Returns an error code. (Upon automation, the return value is used.)<br>**S_OK**: Normal termination<br>**EZNC_OPE_ACTPLC_ADDR**: Invalid NC control unit | |
| □ **Return value** | Value | Meaning |
| | **S_OK** | Normal termination |
| | **S_FALSE** | Communication failure |
| □ **Function** | Stops the PLC program. | |
| □ **Reference** | **Run()** | |
| □ **Specification** | | |

M700 M800

## 2.18.1 IEZNcDevice::SetDevice Set device settings

□ **Custom call procedure**

```
HRESULT      SetDevice(
                       DWORD dwLength,                 // (I) Number of device points
                       LPCOLESTR* lppcwszDevice,       // (I) Device character string
                       LPDWORD lpdwDataType,           // (I) Data type
                       LPDWORD lpdwValue,              // (I) Device value array
                       LONG* plRet                     // (O) Error code
                       )
```

□ **Automation call procedure**

```
             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*: Sets the number of device points to be set. The maximum value is 1K points.

*lppcwszDevice*: Sets the array of the device character string to be set. The device character string needs to be specified as UNICODE string. However if word format (or double word format) is set in the data type, the device character string needs to be set in multiples of 16 (or 32).

*lpdwDataType*: Sets each data type of the device to be set as an array.

| Value | Meaning | Unit in Table 2-4 |
|---|---|---|
| **EZNC_PLC_BIT** | Bit | 1 bit |
| **EZNC_PLC_WORD** | Word | 16 bits |
| **EZNC_PLC_DWORD** | Double word | 32 bits |

*lpdwValue*: Sets the array used to set the device value. When reading, set a dummy value that has the same number of array elements as that of the device character string.

Automation argument:

*vDevice*: Sets the array of the device character string to be set as **VARIANT**. It needs to be sets as UNICODE character string. However if word format (or double word format) is set in the data type, the device character string needs to be set in multiples of 16 (or 32).

*vDataType*: Sets the array of the data type of the device value to be set as **VARIANT**.

| Value | Meaning | Unit in Table 2-4 |
|---|---|---|
| **EZNC_PLC_BIT** | Bit | 1 bit |
| **EZNC_PLC_WORD** | Word | 16 bits |
| **EZNC_PLC_DWORD** | Double word | 32 bits |

*vValue*: Sets the array of the device value to be set as **VARIANT**. When reading, set a dummy value that has the same number of array elements as that of the device character string.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_DATATYPE**: Invalid data type
**EZ_ERR_NOT_SUPPORT**: Not supported

| | | |
|---|---|---|
| □ **Return value** | Value | Meaning |
| | **S_OK**<br>**S_FALSE** | Normal termination<br>Communication failure |
| □ **Function** | Defines the device to be used in the user PLC. The device settings are all based on one-shot operation. The settings for one-shot operation may return to the original state after one cycle of the PLC operation. The settings cannot be retrieved when making the settings next time.<br>This function is not supported with C70. (EZ_ERR_NOT_SUPPORT is returned to plRet.) | |
| □ **Reference** | **ReadDevice(), WriteDevice(), DeleteDeviceAll()** | |
| □ **Specification** | | |

Table 2-4 Applicable devices list

| Device name | Name | Unit | Model | |
|---|---|---|---|---|
| | | | M700 | M800 |
| B | Counter (fixed counter) | 1 bit/16 bits/32 bits | B0～B1FFF (8192 points) | B0～B1BFFF (114688 points) |
| CI | Counter coil | 1 bit/16 bits/32 bits | C0～C1255 (1256 points) | C0～C101023 (101024 points) |
| D | Data register | 16 bits/32 bits | D0～D2047 (2048 points) | D0～D8191 (8192 points) |
| E | Special relay | 1 bit/16 bits/32 bits | E0～E127 (128 points) | E0～E9999 (10000 points) |
| F | Alarm message interface. Temporary memory. | 1 bit/16 bits/32 bits | F0～F1024 (1025 points) | F0～F4095 (4096 points) |
| G | Temporary memory | 1 bit/16 bits/32 bits | G0～G3071 (3072 points) | － |
| I | I device | 1 bit/16 bits/32 bits | I0～I3FF (1024 points) | － |
| J | J device | 1 bit/16 bits/32 bits | J0～J63F (1600 points) | － |
| L | Latch relay (backup memory) | 1 bit/16 bits/32 bits | L0～L511 (512 points) | L0～L2047 (2048 points) |
| M | Temporary memory | 1 bit/16 bits/32 bits | M0～M10239 (10240 points) | M0～M122879 (122880 points) |
| Q | Q device | 1 bit/16 bits/32 bits | Q0～Q1151 (1152 points) | Q0～Q2047 (2048 points) |
| R | File register *1 | 16 bits/32 bits | R0～R32767 (32768 points) | R0～R32767 (32768 points) |
| SM | Special relay *1 | 1 bit/16 bits/32 bits | SM0～SM127 (128 points) | SM0～SM16383 (16384 points) |
| SB | Special relay (for link) | 1 bit/16 bits/32 bits | SB0～SB1FF (512 points) | SB0～SB7FF (2048 points) |
| SD | Special register | 16 bits/32 bits | SD0～SD127 (128 points) | SD0～SD16383 (16384 points) |
| ST | Cumulative timer | 1 bit/16 bits/32 bits | ST0～ST1063 (1064 points) | ST0～ST1255 (1256 points) |
| SW | Special register (for link) | 16 bits/32 bits | SW0～SWFDF (4064 points) | SW0～SW7FF (2048 points) |
| TI | 10 ms unit timer coil | 1 bit | T0～T1703 (1704 points) | T0～104095 (104096 points) |
| U | For two input signal lines to programmable controller *1 | 1 bit/16 bits/32 bits | U0～T17F (384 points) | － |
| V | V device | 1 bit/16 bits/32 bits | V0～V255 (256 points) | V0～V1023 (1024 points) |
| W | For two output signal lines to programmable controller *1 | 1 bit/32 bits | W0～W1FF (512 points) | W0～W5FFF (24576 points) |
| X | Input signal to programmable controller *1 | 1 bit/16 bits/32 bits | X0～X1FFF (8192 points) | X0～X1FFF (8192 points) |
| Y | Output signal to programmable controller *1 | 1 bit/16 bits/32 bits | Y0～Y1FFF (8192 points) | Y0～Y1FFF (8192 points) |
| ZR | File register | 16 bits/32 bits | | ZR0～ZR32767 (32768 points) |

*1: The intended purpose of this device cannot be changed. Do not use any other device (including undefined device) than that corresponding to the I/O signal of the machine.

*2: The devices are read-only.

## 2.18.2 IEZNcDevice::DeleteDeviceAll Delete all device settings

**□ Custom call procedure**

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

**□ Automation call procedure**

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

|  |  |  |
|---|---|---|
| □ **Argument** | *plRet*: Returns an error code. (Upon automation, the return value is used.)<br>**S_OK**: Normal termination<br>**EZ_ERR_NOT_SUPPORT**: Not supported | |
| □ **Return value** | Value | Meaning |
| | **S_OK** | Normal termination |
| | **S_FALSE** | Communication failure |
| □ **Function** | Deletes all the data set in **SetDevice()**.<br><br>This function is not supported with C70. (EZ_ERR_NOT_SUPPORT is returned to plRet.) | |
| □ **Reference** | **SetDevice()** | |
| □ **Specifica-tion** | | |

## 2.18.3 IEZNcDevice::ReadDevice Read device

□ **Custom call procedure**

```
HRESULT    ReadDevice(
                LPDWORD lpdwLength,        // (O) Number of read device points
                LPDWORD* lppdwValue,       // (O) Array of read device value
                LONG* plRet                // (O) Error code
                )
```

□ **Automation call procedure**

```
           Device_Read(
                pvValue As VARIANT*        // (O) Device value array
                ) As LONG                  // (O) Error code
```

□ **Argument**

*lpdwLength*: Returns the number of read devices.

*lppdwValue*: Returns the array that contains the device value. As the device value array is allocated in this product, the client needs to release it explicitly with **CoTaskMemFree()**.

Automation argument:
*pvValue*: Returns the array of the device value as **VARIANT**.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_DATATYPE**: Invalid data type
**EZNC_DATA_READ_READ**: Data is not readable.
**EZNC_DATA_READ_WRITEONLY**: Write-only data
**EZ_ERR_NOT_SUPPORT**: Not supported

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Reads all the devices **set in SetDevice()**.

This function is not supported with C70. (EZ_ERR_NOT_SUPPORT is returned to plRet.)

□ **Reference**

**SetDevice(), WriteDevice()**

□ **Specifica-tion**

## 2.18.4 IEZNcDevice::WriteDevice Write device

□ **Custom call procedure**

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

□ **Automation call procedure**

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

| | | |
|---|---|---|
| □ **Argument** | *plRet*: Returns an error code. (Upon automation, the return value is used.)<br>**S_OK**: Normal termination<br>**EZNC_DATA_WRITE_DATATYPE**: Invalid data type<br>**EZNC_DATA_WRITE_WRITE**: Data is not writable.<br>**EZNC_DATA_WRITE_READONLY**: Read-only data<br>**EZ_ERR_NOT_SUPPORT**: Not supported | |
| □ **Return value** | Value | Meaning |
| | **S_OK** | Normal termination |
| | **S_FALSE** | Communication failure |
| □ **Function** | Writes all the devices **set in SetDevice()**.<br>This function is not supported with C70. (EZ_ERR_NOT_SUPPORT is returned to plRet.) | |
| □ **Reference** | **SetDevice(), ReadDevice()** | |
| □ **Specifica-tion** | | |

## 2.18.5 IEZNcDevice::ReadBlockDevice Batch read devices

□ **Custom call procedure**

```
HRESULT     ReadBlockDevice(
                    DWORD dwLength,                  // (I) Number of device points
                    LPCOLESTR lpcwszDevice,          // (I) Head device character string
                    DWORD dwDataType,                // (I) Data type
                    LPDWORD* lppdwValues,            // (I) Read device value array
                    LONG* plRet                      // (O) Error code
                    )
```

□ **Automation call procedure**

```
            Device_ReadBlock(
                    lLength As LONG                  // (I) Number of device points
            bstrDevice As STRING                     // (I) Head device character string
                    lDataType As LONG                // (I) Data type
                    pvValues As VARIANT*             // (I) Read device value array
                    ) As LONG                        // (O) Error code
```

□ **Argument**

*dwLength*: Sets the number of device points to be set. (2 or more)
The maximum number of points that can be got is determined by the data type of the device being got.

| Device data type | Maximum number of getting points |
|---|---|
| **EZNC_PLC_BIT** | 1280 points |
| **EZNC_PLC_BYTE** | 1280 points |
| **EZNC_PLC_WORD** | 640 points |
| **EZNC_PLC_DWORD** | 320 points |

*lppcwszDevice*: Sets the array of the head device character string to be set. The device character string needs to be set as UNICODE string. However if word format (or double word format) is set in the data type, the device character string needs to be set in multiples of 16 (or 32).

*dwDataType*: Sets each data type of the device to be set.

| Value | Meaning | Unit in |
|---|---|---|
| **EZNC_PLC_BIT** | Bit | 1 bit |
| **EZNC_PLC_BYTE** | Byte | 8 bit |
| **EZNC_PLC_WORD** | Word | 16 bits |
| **EZNC_PLC_DWORD** | Double word | 32 bits |

*lpdwValue*: Sets the array used to set the device value. The data array is secured on the EZSocket side, so explicitly release it on the client side using CoTaskMemFree().
The values are got in the device corresponds to the size for the data type at the lower end of the array.
Example) Data type: **EZNC_PLC_BYTE**, Set device value: 0x82
Read device value: 0x00000082

Automation argument:
*lLength:*See the explanation of *dwLength*.

*bstrDevice:*See the explanation of *lpcwszDevice*.

*lDataType*: See the explanation of *lpdwDataType*.

*pvValue*：Returns the device value array in VARIANT.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_DATATYPE**: Invalid data type
**EZNC_DATA_READ_READ**: Data is not readable.
**EZNC_DATA_READ_WRITEONLY**: Write-only data
**EZ_ERR_NOT_SUPPORT**: Not supported

|  |  |  |
|---|---|---|
| □ **Return value** | Value | Meaning |
|  | **S_OK**<br>**S_FALSE** | Normal termination<br>Communication failure |
| □ **Function** | A device with continuous device points from the head device character string is read out.<br><br>This is valid only for the M700V/M70V/M800 series.<br>This function is not supported with C70 and M700/M70 series. (EZ_ERR_NOT_SUPPORT is returned to plRet.) |  |
| □ **Reference** | **WriteBlockDevice()** |  |
| □ **Specification** |  |  |

## 2.18.6 IEZNcDevice::WriteBlockDevice Batch write devices

□ **Custom call procedure**

```
HRESULT        WriteBlockDevice(
                        DWORD dwLength,                  // (I) Number of device points
                        LPCOLESTR lpcwszDevice,          // (I) Head device character string
                        DWORD dwDataType,                // (I) Data type
                        LPDWORD lppdwValues,             // (I) Write device value array
                        LONG* plRet                      // (O) Error code
                        )
```

□ **Automation call procedure**

```
               Device_WriteBlock(
                        lLength As LONG                  // (I) Number of device points
               bstrDevice As STRING                      // (I) Head device character string
                        lDataType As LONG                // (I) Data type
                        vValues As VARIANT               // (I) Write device value array
                        ) As LONG                        // (O) Error code
```

□ **Argument**

*dwLength*: Sets the number of device points to be set. (2 or more)
The maximum number of points that can be got is determined by the data type of the device being got.

| Device data type | Maximum number of gettable points |
|---|---|
| **EZNC_PLC_BIT** | 1280 points |
| **EZNC_PLC_BYTE** | 1280 points |
| **EZNC_PLC_WORD** | 640 points |
| **EZNC_PLC_DWORD** | 320 points |

*lppcwszDevice*: Sets the array of the head device character string to be set. The device character string needs to be set as UNICODE string. However if word format (or double word format) is set in the data type, the device character string needs to be set in multiples of 16 (or 32).

*dwDataType*: Sets each data type of the device to be set.

| Value | Meaning | Unit in Table 2-4 |
|---|---|---|
| **EZNC_PLC_BIT** | Bit | 1 bit |
| **EZNC_PLC_BYTE** | Byte | 8 bit |
| **EZNC_PLC_WORD** | Word | 16 bits |
| **EZNC_PLC_DWORD** | Double word | 32 bits |

*lpdwValue*: Sets the array used to set the device value.
The value set in the device corresponds to the size for the data type at the lower end of the array.
Example) Data type: **EZNC_PLC_BYTE**, Written device value: 0x12345678
Set device value: 0x78

Automation argument:
*lLength:*See the explanation of *dwLength*.

*bstrDevice:*See the explanation of *lpcwszDevice*.

*lDataType*: See the explanation of *lpdwDataType*.

*vValue*：Returns the device value array in VARIANT.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_DATA_READ_DATATYPE**: Invalid data type
**EZNC_DATA_WRITE_WRITE**: Data is not writable.
**EZNC_DATA_ WRITE_READONLY**: Read-only data
**EZ_ERR_NOT_SUPPORT**: Not supported

| | | |
|---|---|---|
| □ **Return value** | Value | Meaning |
| | **S_OK**<br>**S_FALSE** | Normal termination<br>Communication failure |
| □ **Function** | A device with continuous device points from the head device character string is read out.<br><br>This is valid only for the M700V/M70V/M800 series.<br>This function is not supported with C70 and M700/M70 series. (EZ_ERR_NOT_SUPPORT is returned to plRet.) | |
| □ **Reference** | **ReadBlockDevice()** | |
| □ **Specifica-tion** | | |

C70 M700 M800

## 2.19.1 IEZNcSubFunction3::ChangeInit2 Initialize subfunction

□ **Custom call procedure**

```
HRESULT     ChangeInit2 (
                LONG lSystemType,           // (I) Mitsubishi CNC type
                LONG lReserve1,             // (I) Reservation 1
                LONG lReserve2,             // (I) Reservation 2
                LONG* plRet                 // (O) Error code
                )
```

□ **Automation call procedure**

```
            ChangeInit2 (
                lSystemType As LONG         // (I) Mitsubishi CNC type
                lReserve1 As LONG           // (I) Reservation 1
                lReserve2 As LONG           // (I) Reservation 2
                ) As LONG                   // (O) Error code
```

□ **Argument**

*lSystemType*: Sets the Mitsubishi CNC type.

| Value | Meaning |
|---|---|
| **EZNC_SYS_MELDASC70** | Perform initialization on C70. |
| **EZNC_SYS_MELDAS700M** | Perform initialization on M700 M series. |
| **EZNC_SYS_MELDAS700L** | Perform initialization on M700 L series. |
| **EZNC_SYS_MELDAS800M** | Perform initialization on M800 M series. |
| **EZNC_SYS_MELDAS800L** | Perform initialization on M800 L series. |

*lReserve1*: Not used. (Always set 0.)
*lReserve2*: Not used. (Always set 0.)

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZ_ERR_DATA_RANGE**: Invalid data range

□ **Return value**

| Value | Meaning |
|---|---|
| **S_OK** | Normal termination |
| **S_FALSE** | Communication failure |

□ **Function**

Initializes **IEZNcSubFunction**.

□ **Reference**

□ **Specification**

## 2.19.2 IEZNcSubFunction3::GetToolWorkOffsetOfFile Get data from workpiece offset file

□ **Custom call procedure**

```
HRESULT     GetToolWorkOffsetOfFile(
                  LPCOLESTR lpcwszFileName,    // (I) File name containing a path
                  LONG lHead,                  // (I) Part system
                  LONG lIndex,                 // (I) Workpiece coordinate system number
                  LPCOLESTR* lppcwszAxis,      // (I) Axis name character string array
                  LPOLESTR**    lpppwszData,   // (O) Workpiece coordinate data value
                                                       character string array
                  LONG* plRet                  // (O) Error code
                  )
```

□ **Automation call procedure**

```
            GetToolWorkOffsetOfFile(
                  bstrFileName As STRING       // (I) File name containing a path
                  lHeadAs LONG                 // (I) Part system
                  lIndex As LONG               // (I) Workpiece coordinate system number
                  vAxis As VARIANT             // (I) Axis name character string array
                  pvData As VARIANT*           // (O) Workpiece coordinate data value
                                                       character string array
                  ) As LONG                    // (O) Error code
```

□ **Argument**

*lpcwszFileName*: Sets the file name including path of the workpiece offset file as a **UNICODE** character string.

Set the file with absolute path as below.

Drive name + ":" + \directory name\file name

*lHead*: Sets the system.

*lIndex*: Sets the workpiece coordinate system number to be read.

| Value | Meaning |
|---|---|
| **54** | G54 offset |
| **55** | G55 offset |
| **56** | G56 offset |
| **57** | G57 offset |
| **58** | G58 offset |
| **59** | G59 offset |
| **60** | EXT offset |
| **61** | P1 offset |
| **62** | P2 offset |
| : | : |
| **108** | P48 offset |

*lppcwszAxis*: Sets the axis name as a **UNICODE** character string array. (Example: "X") The number of array elements is 8 (0 to 7). Set a **NULL** character string to the axes not applicable. (A NULL pointer cannot be set.) This is used only in C70. Set a NULL character string to all elements in any other model.

*lpppwszData*: Returns the workpiece offset data value as a UNICODE character string array. As the data value array is allocated in this product, the client needs to release it explicitly with **CoTaskMemFree ()**.

| *lpppwszData* | Tool change data type | 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 is [inch] or [mm] depending on the parameter setting of NC.

Automation argument:
*bstrFileName*: See the explanation of *lpcwszFileName*.
*vAxis: See the explanation of lppcwszAxis*.
*pvData*: Returns the workpiece offset data value as VARIANT.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_FILE_OPEN_FILENOTEXIST**: File does not exist
**EZNC_FILE_OPEN_OPEN**: File cannot be opened
**EZNC_FILE_READFILE_READ**: Data is not readable
**EZNC_DATA_NOT_EXIST**: Data does not exist
**EZ_ERR_MEMORY_ALLOC**: Memory cannot be allocated.

| | | |
|---|---|---|
| □ **Return value** | Value | Meaning |
| | **S_OK**<br>**S_FALSE** | Normal termination<br>Communication failure |
| □ **Function** | Gets the offset value of the workpiece coordinate system of the set part system and axis No. | |
| □ **Reference** | **OpenFile3(), CloseFile2(), ReadFile2(), WriteFile(), SetToolWorkOffsetFile()** | |
| □ **Specification** | | |

## 2.19.3 IEZNcSubFunction3::SetToolWorkOffsetOfFile Set data to workpiece offset file

**□ Custom call procedure**

```
HRESULT    SetToolWorkOffsetOfFile(
                LPCOLESTR lpcwszFileName,     // (I) File name containing a path
                LONG lMode,                   // (I) Setting mode
                LONG lHead,                   // (I) Part System
                LONG lIndex,                  // (I) Workpiece coordinate system number
                LPCOLESTR* lppcwszAxis,       // (I) Axis name string character array
                LPCOLESTR* lppcwszData,       // (I) Workpiece coordinate data value
                                              //     character string array
                LONG* plRet                   // (O) Error code
                )
```

**□ Automation call procedure**

```
           SetToolWorkOffsetOfFile(
                bstrFileName As STRING        // (I) File name containing a path
                lModeAs LONG                  // (I) Setting mode
                lHeadAs LONG                  // (I) Part system
                lIndex As LONG                // (I) Workpiece coordinate system number
                vAxis As VARIANT              // (I) Axis name character string array
                vData As VARIANT              // (I) Workpiece coordinate data value
                                              //     character string array
                ) As LONG                     // (O) Error code
```

**□ Argument**

*lpcwszFileName*: Sets the file name including path of the workpiece offset file as a **UNICODE** character string.

Set the file with absolute path as below.

Drive name + ":" + \directory name\file name

*lMode*: Sets the setting mode of the tool life management file.

| Value | Meaning |
|---|---|
| **EZNC_FILE_CREATE** | Creates a new tool life management file. |
| **EZNC_FILE_OPEN** | Modifies an existing tool life management file |

*lHead*: Sets the system.

*lIndex*: Sets the workpiece coordinate system number to be written to.

| Value | Meaning |
|---|---|
| **54** | G54 offset |
| **55** | G55 offset |
| **56** | G56 offset |
| **57** | G57 offset |
| **58** | G58 offset |
| **59** | G59 offset |
| **60** | EXT offset |
| **61** | P1 offset |
| **62** | P2 offset |
| : | : |
| **108** | P48 offset |

*lppcwszAxis*: Sets the axis name as a **UNICODE** character string. Set a **NULL** character string to the axes that do not exist. This is used only in C70. Set a NULL character string to all elements in any other model.

*lppcwszData*: Returns the workpiece offset data as a UNICODE character string. Set a **NULL** character string to the axes that do not exist.
The unit is [inch] or [mm] depending on the parameter setting of CNC.

| *lppcwszData* | Tool change data type | 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*: See the explanation of *lpcwszFileName*.
*vAxis: See the explanation of lppcwszAxis*.
*vData*: Creates the workpiece offset data as a UNICODE character string and sets it by substituting it in vData(VARIANT). For details of the workpiece offset data, see the explanation of *lppcwszData* and the index above.

*plRet*: Returns an error code. (Upon automation, the return value is used.)
**S_OK**: Normal termination
**EZNC_FILE_OPEN_FILENOTEXIST**: File does not exist
**EZNC_FILE_OPEN_OPEN**: File cannot be opened
**EZNC_FILE_WRITEFILE_WRITE:** Data is not writable
**EZ_ERR_NULLPTR:** Argument is NULL pointer

```
[Example]
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->SetToolLifeValueOfFile(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** | Value | Meaning |
| | **S_OK**<br>**S_FALSE** | Normal termination<br>Communication failure |
| □ **Function** | Sets the offset value of the workpiece coordinate system of the set part system and axis No. | |
| □ **Reference** | **OpenFile3(), CloseFile2(), ReadFile2(), WriteFile(), GetToolWorkOffsetFile()** | |
| □ **Specifica-tion** | | |

# 3. ERROR CODE LIST

This section provides a list of error codes.

Table 3-1 Error code list

| No. | Error code | Number | Description |
|---|---|---|---|
| 1. | EZ_ERR_NOT_OPEN | 0x80A00101 | Communication lines are not open. |
| 2. | EZ_ERR_DOUBLE_OPEN | 0x80A00104 | Double open error. |
| 3. | EZ_ERR_DATA_TYPE | 0x80A00105 | Invalid argument data type. |
| 4. | EZ_ERR_DATA_RANGE | 0x80A00106 | Invalid argument data range. |
| 5. | EZ_ERR_NOT_SUPPORT | 0x80A00107 | Not supported. |
| 6. | EZ_ERR_CANNOT_OPEN | 0x80A00109 | Communication line cannot be opened. |
| 7. | EZ_ERR_NULLPTR | 0x80A0010A | Argument is NULL pointer. |
| 8. | EZ_ERR_DATA_LENGTH | 0x80A0010B | Invalid argument data. |
| 9. | EZ_ERR_OPEN_COMM | 0x80A0010C | COMM port handle error. |
| 10. | EZ_ERR_MEMORY_ALLOC | 0x80B00101 | Memory cannot be allocated. |
| 11. | EZNC_ERR_CANNOT_GETPCERR | 0x80B00102 | EZSocketPc error cannot be obtained. |
| 12. | EZNC_FILE_OPEN_MODE | 0x80B00201 | Invalid mode specification. |
| 13. | EZNC_FILE_OPEN_NOTOPEN | 0x80B00202 | File is not open. |
| 14. | EZNC_FILE_OPEN_FILEEXIST | 0x80B00203 | File already exists. |
| 15. | EZNC_FILE_OPEN_ALREADYOPENED | 0x80B00204 | File is already open. |
| 16. | EZNC_FILE_OPEN_CREATE | 0x80B00205 | Temporary file cannot be created. |
| 17. | EZNC_FILE_WRITEFILE_NOTOPEN | 0x80B00206 | File is open without write mode specification. |
| 18. | EZNC_FILE_WRITEFILE_LENGTH | 0x80B00207 | Invalid write data size. |
| 19. | EZNC_FILE_WRITEFILE_WRITE | 0x80B00208 | Not writable. |
| 20. | EZNC_FILE_READFILE_NOTOPEN | 0x80B00209 | File is open without read mode specification. |
| 21. | EZNC_FILE_READFILE_READ | 0x80B0020A | Not readable. |
| 22. | EZNC_FILE_READFILE_CREATE | 0x80B0020B | Temporary file cannot be created. |
| 23. | EZNC_FILE_OPEN_FILENOTEXIST | 0x80B0020C | File does not exist (READ mode). |
| 24. | EZNC_FILE_OPEN_OPEN | 0x80B0020D | File cannot be opened. |
| 25. | EZNC_FILE_OPEN_ILLEGALPATH | 0x80B0020E | Invalid file path. |
| 26. | EZNC_FILE_READFILE_ILLEGALFILE | 0x80B0020F | Invalid read file. |
| 27. | EZNC_FILE_WRITEFILE_ILLEGALFILE | 0x80B00210 | Invalid write file. |
| 28. | EZNC_COMM_CANNOT_OPEN | 0x80B00301 | Host name for local connection used for automation call is invalid. |
| 29. | EZNC_COMM_NOTSETUP_PROTOCOL | 0x80B00302 | TCP/IP communication is not configured. |
| 30. | EZNC_COMM_ALREADYOPENED | 0x80B00303 | Cannot be set because communication is already in progress. |
| 31. | EZNC_COMM_NOTMODULE | 0x80B00304 | No submodule. |
| 32. | EZNC_COMM_CREATEPC | 0x80B00305 | EZSocketPc objects cannot be created. |
| 33. | EZNC_DATA_NOT_EXIST | 0x80B00401 | Data does not exist. |
| 34. | EZNC_DATA_DUPLICATE | 0x80B00402 | Duplicate data. |
| 35. | EZNC_PARAM_FILENOTEXIST | 0x80B00501 | No parameter information file. |
| 36. | EZNC_SYSFUNC_IOCTL_ADDR | 0x80020190 | Invalid NC control unit number. |
| 37. | EZNC_SYSFUNC_IOCTL_NOTOPEN | 0x80020102 | Device is not open. |
| 38. | EZNC_SYSFUNC_IOCTL_FUNCTION | 0x80020132 | Invalid command. |
| 39. | EZNC_SYSFUNC_IOCTL_DATA | 0x80020133 | Invalid communication parameter data range. |
| 40. | EZNC_FILE_DIR_FILESYSTEM | 0x80030143 | File system error. |
| 41. | EZNC_FILE_DIR_NODIR | 0x80030191 | Directory does not exist. |
| 42. | EZNC_FILE_DIR_NODRIVE | 0x8003019B | Drive does not exist. |
| 43. | EZNC_PCFILE_DIR_NODIR | 0x800301A2 | Directory does not exist. |
| 44. | EZNC_PCFILE_DIR_NODRIVE | 0x800301A8 | Drive does not exist. |
| 45. | EZNC_OPE_CURRALM_ADDR | 0x80050D90 | Invalid system, spindle specification. |
| 46. | EZNC_OPE_CURRALM_ALMTYPE | 0x80050D02 | Invalid alarm type. |
| 47. | EZNC_OPE_CURRALM_DATAERR | 0x80050D03 | Error in communication data between NC and personal computer. |
| 48. | EZ_ERR_NOT_OPEN | 0x80A00101 | Communication lines are not open. |
| 49. | EZ_ERR_DOUBLE_OPEN | 0x80A00104 | Double open error. |
| 50. | EZ_ERR_DATA_TYPE | 0x80A00105 | Invalid argument data type. |

| No. | Error code | Number | Description |
|---|---|---|---|
| 51. | EZ_ERR_DATA_RANGE | 0x80A00106 | Invalid argument data range. |
| 52. | EZ_ERR_NOT_SUPPORT | 0x80A00107 | Not supported. |
| 53. | EZ_ERR_CANNOT_OPEN | 0x80A00109 | Communication line cannot be opened. |
| 54. | EZ_ERR_NULLPTR | 0x80A0010A | Argument is NULL pointer. |
| 55. | EZ_ERR_DATA_LENGTH | 0x80A0010B | Invalid argument data. |
| 56. | EZ_ERR_OPEN_COMM | 0x80A0010C | COMM port handle error. |
| 57. | EZ_ERR_MEMORY_ALLOC | 0x80B00101 | Memory cannot be allocated. |
| 58. | EZNC_ERR_CANNOT_GETPCERR | 0x80B00102 | EZSocketPc error cannot be obtained. |
| 59. | EZNC_FILE_OPEN_MODE | 0x80B00201 | Invalid mode specification. |
| 60. | EZNC_FILE_OPEN_NOTOPEN | 0x80B00202 | File is not open. |
| 61. | EZNC_FILE_OPEN_FILEEXIST | 0x80B00203 | File already exists. |
| 62. | EZNC_FILE_OPEN_ALREADYOPENED | 0x80B00204 | File is already open. |
| 63. | EZNC_FILE_OPEN_CREATE | 0x80B00205 | Temporary file cannot be created. |
| 64. | EZNC_FILE_WRITEFILE_NOTOPEN | 0x80B00206 | File is open without write mode specification. |
| 65. | EZNC_FILE_WRITEFILE_LENGTH | 0x80B00207 | Invalid write data size. |
| 66. | EZNC_FILE_WRITEFILE_WRITE | 0x80B00208 | Not writable. |
| 67. | EZNC_FILE_READFILE_NOTOPEN | 0x80B00209 | File is open without read mode specification. |
| 68. | EZNC_FILE_READFILE_READ | 0x80B0020A | Not readable. |
| 69. | EZNC_FILE_READFILE_CREATE | 0x80B0020B | Temporary file cannot be created. |
| 70. | EZNC_FILE_OPEN_FILENOTEXIST | 0x80B0020C | File does not exist (READ mode). |
| 71. | EZNC_FILE_OPEN_OPEN | 0x80B0020D | File cannot be opened. |
| 72. | EZNC_FILE_OPEN_ILLEGALPATH | 0x80B0020E | Invalid file path. |
| 73. | EZNC_FILE_READFILE_ILLEGALFILE | 0x80B0020F | Invalid read file. |
| 74. | EZNC_FILE_WRITEFILE_ILLEGALFILE | 0x80B00210 | Invalid write file. |
| 75. | EZNC_COMM_CANNOT_OPEN | 0x80B00301 | Host name for local connection used for automation call is invalid. |
| 76. | EZNC_COMM_NOTSETUP_PROTOCOL | 0x80B00302 | TCP/IP communication is not configured. |
| 77. | EZNC_COMM_ALREADYOPENED | 0x80B00303 | Cannot be set because communication is already in progress. |
| 78. | EZNC_COMM_NOTMODULE | 0x80B00304 | No submodule. |
| 79. | EZNC_COMM_CREATEPC | 0x80B00305 | EZSocketPc objects cannot be created. |
| 80. | EZNC_DATA_NOT_EXIST | 0x80B00401 | Data does not exist. |
| 81. | EZNC_DATA_DUPLICATE | 0x80B00402 | Duplicate data. |
| 82. | EZNC_PARAM_FILENOTEXIST | 0x80B00501 | No parameter information file. |
| 83. | EZNC_SYSFUNC_IOCTL_ADDR | 0x80020190 | Invalid NC control unit number. |
| 84. | EZNC_SYSFUNC_IOCTL_NOTOPEN | 0x80020102 | Device is not open. |
| 85. | EZNC_SYSFUNC_IOCTL_FUNCTION | 0x80020132 | Invalid command. |
| 86. | EZNC_SYSFUNC_IOCTL_DATA | 0x80020133 | Invalid communication parameter data range. |
| 87. | EZNC_FILE_DIR_FILESYSTEM | 0x80030143 | File system error. |
| 88. | EZNC_FILE_DIR_NODIR | 0x80030191 | Directory does not exist. |
| 89. | EZNC_FILE_DIR_NODRIVE | 0x8003019B | Drive does not exist. |
| 90. | EZNC_PCFILE_DIR_NODIR | 0x800301A2 | Directory does not exist. |
| 91. | EZNC_PCFILE_DIR_NODRIVE | 0x800301A8 | Drive does not exist. |
| 92. | EZNC_OPE_CURRALM_ADDR | 0x80050D90 | Invalid system, spindle specification. |
| 93. | EZNC_OPE_CURRALM_ALMTYPE | 0x80050D02 | Invalid alarm type. |
| 94. | EZNC_OPE_CURRALM_DATAERR | 0x80050D03 | Error in communication data between NC and personal computer. |
| 95. | EZNC_DATA_TLFTOOL_PARAMERR | 0x80041194 | Invalid type specified for life control data. |
| 96. | EZNC_DATA_TLFTOOL_MAXMINERR | 0x80041195 | Setting data is out of range. |
| 97. | EZNC_DATA_TLFTOOL_UNMACH | 0x80041196 | Specified tool number mismatch. |
| 98. | EZNC_DATA_TLFTOOL_OUTOFSPEC | 0x80041197 | Specified tool number is out of specifications. |
| 99. | EZNC_DATA_READ_ADDR | 0x80040190 | Invalid system, spindle specification. |
| 100. | EZNC_DATA_READ_SECT | 0x80040191 | Invalid section number. |

| No. | Error code | Number | Description |
|---|---|---|---|
| 101. | EZNC_DATA_READ_SUBSECT | 0x80040192 | Invalid subsection number. |
| 102. | EZNC_DATA_READ_DATASIZE | 0x80040196 | Application does not fit into prepared buffer. |
| 103. | EZNC_DATA_READ_DATATYPE | 0x80040197 | Invalid data type. |
| 104. | EZNC_DATA_READ_READ | 0x8004019D | Data is not readable. |
| 105. | EZNC_DATA_READ_WRITEONLY | 0x8004019F | Write-only data. |
| 106. | EZNC_DATA_READ_AXIS | 0x800401A0 | Invalid axis specification. |
| 107. | EZNC_DATA_READ_DATANUM | 0x800401A1 | Invalid data number. |
| 108. | EZNC_DATA_READ_NODATA | 0x800401A3 | Read data not found |
| 109. | EZNC_DATA_READ_VALUE | 0x8004019A | Invalid read data range. |
| 110. | EZNC_DATA_WRITE_ADDR | 0x80040290 | Invalid system, spindle specification. |
| 111. | EZNC_DATA_WRITE_SECT | 0x80040291 | Invalid section number. |
| 112. | EZNC_DATA_WRITE_SUBSECT | 0x80040292 | Invalid subsection number. |
| 113. | EZNC_DATA_WRITE_DATASIZE | 0x80040296 | Application does not fit into prepared buffer. |
| 114. | EZNC_DATA_WRITE_DATATYPE | 0x80040297 | Invalid data type. |
| 115. | EZNC_DATA_WRITE_READONLY | 0x8004029B | Read-only data. |
| 116. | EZNC_DATA_WRITE_WRITE | 0x8004029E | Data is not writable. |
| 117. | EZNC_DATA_WRITE_AXIS | 0x800402A0 | Invalid axis specification. |
| 118. | EZNCDATA_WRITE_SAFETYPWLOCK | 0x8004024D | Safety password locked. |
| 119. | EZNCDATA_WRITE_UOPEN_FORMAT | 0x800402A2 | Formatting canceled because of invalid SRAM open parameter. |
| 120. | EZNCDATA_WRITE_EDTFILE_REGIST | 0x800402A4 | Cannot register edit file (already being edited). |
| 121. | EZNCDATA_WRITE_EDTFILE_RELEASE | 0x800402A5 | Cannot release edit file. |
| 122. | EZNCDATA_WRITE_NODATA | 0x800402A3 | No data at write destination. |
| 123. | EZNCDATA_WRITE_VALUE | 0x8004029A | Invalid write data range. |
| 124. | EZNCDATA_WRITE_SAFE_NOPASSWD | 0x800402A6 | Safety password not set. |
| 125. | EZNCDATA_WRITE_SAFE_CHECKERR | 0x800402A7 | Safety data consistency check error |
| 126. | EZNCDATA_WRITE_SAFE_DATATYPE | 0x800402A9 | Safety data type invalid |
| 127. | EZNCDATA_WRITE_SORT | 0x800402A8 | Cannot write while sorting tool data. |
| 128. | EZNC_DATA_MDLCANCEL_NOTREGIST | 0x80040501 | Not registered for fast read. |
| 129. | EZNC_DATA_MDLREGIST_PRIORITY | 0x80040402 | Invalid priority specification. |
| 130. | EZNC_DATA_MDLREGIST_REGIST | 0x80040401 | Exceeded the limit of registrations. |
| 131. | EZNC_DATA_MDLREGIST_ADDR | 0x80040490 | Invalid address. |
| 132. | EZNC_DATA_MDLREGIST_SECT | 0x80040491 | Invalid section number. |
| 133. | EZNC_DATA_MDLREGIST_SUBSECT | 0x80040492 | Invalid subsection number. |
| 134. | EZNC_DATA_MDLREGIST_DATATYPE | 0x80040497 | Invalid data type. |
| 135. | EZNC_DATA_MDLREGIST_READONLY | 0x8004049B | Read-only data. |
| 136. | EZNC_DATA_MDLREGIST_READ | 0x8004049D | Data is not readable. |
| 137. | EZNC_DATA_MDLREGIST_WRITEONLY | 0x8004049F | Write-only data. |
| 138. | EZNC_DATA_MDLREGIST_AXIS | 0x800404A0 | Invalid axis specification. |
| 139. | EZNC_DATA_RETHREADWRITE_NODATA | 0x80040BA3 | Rethread cut position not set. |
| 140. | EZNC_FILE_DIR_ALREADYOPENED | 0x80030101 | A different directory is already opened. |
| 141. | EZNC_FILE_DIR_DATASIZE | 0x80030103 | Exceeded maximum data size. |
| 142. | EZNC_FILE_DIR_NAMELENGTH | 0x80030148 | File name is too long. |
| 143. | EZNC_FILE_DIR_ILLEGALNAME | 0x80030198 | Invalid file name format. |
| 144. | EZNC_FILE_DIR_NOTOPEN | 0x80030190 | Not open. |
| 145. | EZNC_FILE_DIR_READ | 0x80030194 | File information read error. |
| 146. | EZNC_PCFILE_DIR_ALREADYOPENED | 0x80030102 | A different directory is already opened (personal computer only). |
| 147. | EZNC_PCFILE_DIR_NOTOPEN | 0x800301A0 | Not open. |
| 148. | EZNC_PCFILE_DIR_NOFILE | 0x800301A1 | File does not exist. |
| 149. | EZNC_PCFILE_DIR_READ | 0x800301A5 | File information read error. |
| 150. | EZNC_FILE_COPY_BUSY | 0x80030447 | Copying is disabled (during operation). |

| No. | Error code | Number | Description |
|---|---|---|---|
| 151. | EZNC_FILE_COPY_ENTRYOVER | 0x80030403 | Exceeded the limit of registrations. |
| 152. | EZNC_FILE_COPY_FILEEXIST | 0x80030401 | Copy destination file already exists. |
| 153. | EZNC_FILE_COPY_FILESYSTEM | 0x80030443 | File system error. |
| 154. | EZNC_FILE_COPY_NAMELENGTH | 0x80030448 | File name is too long. |
| 155. | EZNC_FILE_COPY_ILLEGALNAME | 0x80030498 | Invalid file name format. |
| 156. | EZNC_FILE_COPY_MEMORYOVER | 0x80030404 | Memory capacity exceeded. |
| 157. | EZNC_FILE_COPY_NODIR | 0x80030491 | Directory does not exist. |
| 158. | EZNC_FILE_COPY_NODRIVE | 0x8003049B | Drive does not exist. |
| 159. | EZNC_FILE_COPY_NOFILE | 0x80030442 | File does not exist. |
| 160. | EZNC_FILE_COPY_PLCRUN | 0x80030446 | Copying is disabled (PLC in operation). |
| 161. | EZNC_FILE_COPY_READ | 0x80030494 | Transfer source file is not readable. |
| 162. | EZNC_FILE_COPY_WRITE | 0x80030495 | Transfer destination file is not writable. |
| 163. | EZNC_FILE_COPY_PROTECT | 0x8003044A | Copying is disabled (protected). |
| 164. | EZNC_FILE_COPY_DIFFER | 0x80030405 | Verification error. |
| 165. | EZNC_FILE_COPY_NOTSUPPORTED | 0x80030449 | Verification function is not supported. |
| 166. | EZNC_FILE_COPY_EXECUTING | 0x8003044C | Copying file. |
| 167. | EZNC_FILE_COPY_NOTOPEN | 0x80030490 | File is not open. |
| 168. | EZNC_FILE_COPY_WRITE_WARNING | 0x80030495 | Transfer destination file is not writable. |
| 169. | EZNC_FILE_COPY_SAFETYPWLOCK | 0x8003044D | Safety password locked. |
| 170. | EZNC_FILE_COPY_ILLEGALFORMAT | 0x8003049D | Invalid file format. |
| 171. | EZNC_FILE_COPY_WRONGPASSWORD | 0x8003049E | Password is wrong. |
| 172. | EZNC_PCFILE_COPY_CREATE | 0x800304A4 | File cannot be created (personal computer only). |
| 173. | EZNC_PCFILE_COPY_OPEN | 0x800304A3 | File cannot be opened (personal computer only). |
| 174. | EZNC_PCFILE_COPY_FILEEXIST | 0x80030402 | Copy destination file already exists. |
| 175. | EZNC_PCFILE_COPY_ILLEGALNAME | 0x800304A7 | Invalid file name format. |
| 176. | EZNC_PCFILE_COPY_NODIR | 0x800304A2 | Directory does not exist. |
| 177. | EZNC_PCFILE_COPY_NODRIVE | 0x800304A8 | Drive does not exist. |
| 178. | EZNC_PCFILE_COPY_NOFILE | 0x800304A1 | File does not exist. |
| 179. | EZNC_PCFILE_COPY_READ | 0x800304A5 | Transfer source file is not readable. |
| 180. | EZNC_PCFILE_COPY_WRITE | 0x800304A6 | Transfer destination file is not writable. |
| 181. | EZNC_PCFILE_COPY_MEMORYOVER | 0x80030406 | Disk space exceeded. |
| 182. | EZNC_PCFILE_COPY_NOTOPEN | 0x800304A0 | File is not open. |
| 183. | EZNC_FILE_DEL_NOTDELETE | 0x80030201 | File cannot be deleted. |
| 184. | EZNC_FILE_DEL_NOFILE | 0x80030242 | File does not exist. |
| 185. | EZNC_FILE_DEL_FILESYSTEM | 0x80030243 | File system error. |
| 186. | EZNC_FILE_DEL_BUSY | 0x80030247 | Deletion is disabled (during operation). |
| 187. | EZNC_FILE_DEL_NAMELENGTH | 0x80030248 | File name is too long. |
| 188. | EZNC_FILE_DEL_PROTECT | 0x8003024A | Deletion is disabled (protected). |
| 189. | EZNC_FILE_DEL_NODIR | 0x80030291 | Directory does not exist. |
| 190. | EZNC_FILE_DEL_ILLEGALNAME | 0x80030298 | Invalid file name format. |
| 191. | EZNC_FILE_DEL_NODRIVE | 0x8003029B | Drive does not exist. |
| 192. | EZNC_PCFILE_DEL_NOTDELETE | 0x80030202 | File cannot be deleted. |
| 193. | EZNC_PCFILE_DEL_ILLEGALNAME | 0x800302A7 | Invalid file name format. |
| 194. | EZNC_PCFILE_DEL_NODIR | 0x800302A2 | Directory does not exist. |
| 195. | EZNC_PCFILE_DEL_NODRIVE | 0x800302A8 | Drive does not exist. |
| 196. | EZNC_PCFILE_DEL_NOFILE | 0x800302A1 | File does not exist. |
| 197. | EZNC_FILE_REN_FILEEXIST | 0x80030301 | New file name already exists. |
| 198. | EZNC_FILE_REN_NOFILE | 0x80030342 | File does not exist. |
| 199. | EZNC_FILE_REN_FILESYSTEM | 0x80030343 | File system error. |
| 200. | EZNC_FILE_REN_BUSY | 0x80030347 | Renaming is disabled (during operation). |

| No. | Error code | Number | Description |
|---|---|---|---|
| 201. | EZNC_FILE_REN_NAMELENGTH | 0x80030348 | File name is too long. |
| 202. | EZNC_FILE_REN_PROTECT | 0x8003034A | Renaming is disabled (protected). |
| 203. | EZNC_FILE_REN_NODIR | 0x80030391 | Directory does not exist. |
| 204. | EZNC_FILE_REN_ILLEGALNAME | 0x80030398 | Invalid file name format. |
| 205. | EZNC_FILE_REN_NODRIVE | 0x8003039B | Drive does not exist. |
| 206. | EZNC_PCFILE_REN_NOTRENAME | 0x80030303 | Renaming is disabled. |
| 207. | EZNC_PCFILE_REN_SAMENAME | 0x80030305 | New and old file names are identical. |
| 208. | EZNC_PCFILE_REN_FILEEXIST | 0x80030302 | New file name already exists. |
| 209. | EZNC_PCFILE_REN_ILLEGALNAME | 0x800303A7 | Invalid file name format. |
| 210. | EZNC_PCFILE_REN_NODIR | 0x800303A2 | Directory does not exist. |
| 211. | EZNC_PCFILE_REN_NODRIVE | 0x800303A8 | Drive does not exist. |
| 212. | EZNC_PCFILE_REN_NOFILE | 0x800303A1 | File does not exist. |
| 213. | EZNC_FILE_DISKFREE_NODIR | 0x80030691 | Directory does not exist. |
| 214. | EZNC_FILE_DISKFREE_NODRIVE | 0x8003069B | Drive does not exist. |
| 215. | EZNC_FILE_DISKFREE_FILESYSTEM | 0x80030643 | File system error. |
| 216. | EZNC_FILE_DISKFREE_NAMELENGTH | 0x80030648 | File name is too long. |
| 217. | EZNC_FILE_DISKFREE_ILLEGALNAME | 0x80030648 | Invalid file name format. |
| 218. | EZNC_PCFILE_DISKFREE_NODIR | 0x800306A2 | Directory does not exist (personal computer only). |
| 219. | EZNC_PCFILE_DISKFREE_NODRIVE | 0x800306A8 | Drive does not exist (personal computer only). |
| 220. | EZNC_FILE_DRVLIST_DATASIZE | 0x80030701 | Application does not fit into prepared buffer. |
| 221. | EZNC_FILE_DRVLIST_READ | 0x80030794 | Drive information read error. |
| 222. | EZNC_ENET_ALREADYOPEN | 0x82020001 | Already open. |
| 223. | EZNC_ENET_NOTOPEN | 0x82020002 | Not open. |
| 224. | EZNC_ENET_CARDNOTEXIST | 0x82020004 | Card does not exist. |
| 225. | EZNC_ENET_BADCHANNEL | 0x82020006 | Invalid channel number. |
| 226. | EZNC_ENET_BADFD | 0x82020007 | Invalid file descriptor. |
| 227. | EZNC_ENET_NOTCONNECT | 0x8202000A | Not connected. |
| 228. | EZNC_ENET_NOTCLOSE | 0x8202000B | Not closed. |
| 229. | EZNC_ENET_TIMEOUT | 0x82020014 | Time-out. |
| 230. | EZNC_ENET_DATAERR | 0x82020015 | Invalid data. |
| 231. | EZNC_ENET_CANCELED | 0x82020016 | Terminated by cancel request. |
| 232. | EZNC_ENET_ILLEGALSIZE | 0x82020017 | Invalid packet size. |
| 233. | EZNC_ENET_TASKQUIT | 0x82020018 | Terminated due to by end of task. |
| 234. | EZNC_ENET_UNKNOWNFUNC | 0x82020032 | Invalid command. |
| 235. | EZNC_ENET_SETDATAERR | 0x82020033 | Invalid setting data. |
| 236. | EZNC_READ_CACHE_REGIST | 0x80060001 | Invalid data read cache. |
| 237. | EZNC_READ_CACHE_ADDR | 0x80060090 | Invalid address. |
| 238. | EZNC_READ_CACHE_SECT | 0x80060091 | Invalid section number. |
| 239. | EZNC_READ_CACHE_SUBSECT | 0x80060092 | Invalid subsection number. |
| 240. | EZNC_READ_CACHE_DATATYPE | 0x80060097 | Invalid data type. |
| 241. | EZNC_READ_CACHE_DATA | 0x8006009A | Invalid data range. |
| 242. | EZNC_READ_CACHE_READ | 0x8006009D | Data is not readable. |
| 243. | EZNC_READ_CACHE_WRITEONLY | 0x8006009F | Invalid data type. |
| 244. | EZNC_READ_CACHE_AXIS | 0x800600A0 | Invalid axis specification. |
| 245. | EZNC_FS_OPEN_FILE_MALLOC | 0x80070140 | Work area cannot be allocated. |
| 246. | EZNC_FS_OPEN_FILE_OPEN | 0x80070142 | File cannot be opened. |
| 247. | EZNC_FS_OPEN_FILE_BUSY | 0x80070147 | File cannot be opened (during operation). |
| 248. | EZNC_FS_OPEN_FILE_NAMELENGTH | 0x80070148 | File path is too long. |
| 249. | EZNC_FS_OPEN_FILE_NOTSUPPORTED | 0x80070149 | Not supported (CF not supported). |
| 250. | EZNC_FS_OPEN_FILE_ALREADYOPEN | 0x80070192 | Already open. |

| No. | Error code | Number | Description |
|---|---|---|---|
| 251. | EZNC_FS_OPEN_FILE_FILEFULL | 0x80070199 | Maximum number of open files exceeded. |
| 252. | EZNC_FS_OPEN_FILE_ALREADYOPEN | 0x80070192 | Already open. |
| 253. | EZNC_FS_OPEN_FILE_SORT | 0x8007019F | Cannot open while sorting tool data. |
| 254. | EZNC_FS_OPEN_FILE_SAFE_NOPASSWD | 0x800701B0 | Safety password not authenticated. |
| 255. | EZNC_FS_CLOSE_FILE_NOTOPEN | 0x80070290 | File is not open. |
| 256. | EZNC_FS_CREATE_FILE_MALLOC | 0x80070340 | Work area cannot be allocated. |
| 257. | EZNC_FS_CREATE_FILE_BUSY | 0x80070347 | File cannot be created (during operation). |
| 258. | EZNC_FS_CREATE_FILE_NAMELENGTH | 0x80070348 | File path is too long. |
| 259. | EZNC_FS_CREATE_FILE_NOTSUPPORTED | 0x80070349 | Not supported (CF not supported). |
| 260. | EZNC_FS_CREATE_FILE_ALREADYOPEN | 0x80070392 | Already created. |
| 261. | EZNC_FS_CREATE_FILE_CREATE | 0x80070393 | File cannot be created. |
| 262. | EZNC_FS_CREATE_FILE_FILEFULL | 0x80070399 | Maximum number of open files exceeded. |
| 263. | EZNC_FS_CREATE_FILE_NODRIVE | 0x8007039B | Drive does not exist. |
| 264. | EZNC_FS_READ_FILE_NOTOPEN | 0x80070490 | File is not open. |
| 265. | EZNC_FS_READ_FILE_READ | 0x80070494 | File information read error. |
| 266. | EZNC_FS_WRITE_FILE_NOTSUPPORTED | 0x80070549 | Write is not available. |
| 267. | EZNC_FS_WRITE_FILE_NOTOPEN | 0x80070590 | File is not open. |
| 268. | EZNC_FS_WRITE_FILE_WRITE | 0x80070595 | File write error. |
| 269. | EZNC_FS_REMOVE_FILE_REMOVEERR | 0x80070740 | File deletion error. |
| 270. | EZNC_FS_REMOVE_FILE_NOFILE | 0x80070742 | File does not exist. |
| 271. | EZNC_FS_REMOVE_FILE_BUSY | 0x80070747 | File cannot be deleted (during operation). |
| 272. | EZNC_FS_REMOVE_FILE_NAMELENGTH | 0x80070748 | File path is too long. |
| 273. | EZNC_FS_REMOVE_FILE_NOTSUPPORTED | 0x80070749 | Not supported (CF not supported). |
| 274. | EZNC_FS_REMOVE_FILE_ALREADYOPEN | 0x80070792 | File is already open. |
| 275. | EZNC_FS_REMOVE_FILE_NODRIVE | 0x8007079B | Drive does not exist. |
| 276. | EZNC_FS_RENAME_FILE_NOFILE | 0x80070842 | File does not exist. |
| 277. | EZNC_FS_RENAME_FILE_NOTRENAME | 0x80070843 | File cannot be renamed. |
| 278. | EZNC_FS_RENAME_FILE_NAMELENGTH | 0x80070848 | File path is too long. |
| 279. | EZNC_FS_RENAME_FILE_NOTSUPPORTED | 0x80070849 | Not supported (CF not supported). |
| 280. | EZNC_FS_RENAME_FILE_ALREADYOPEN | 0x80070892 | File is already open. |
| 281. | EZNC_FS_RENAME_FILE_FILEFULL | 0x80070899 | Maximum number of open files exceeded. |
| 282. | EZNC_FS_RENAME_FILE_NODRIVE | 0x8007089B | Drive does not exist. |
| 283. | EZNC_FS_IOCTL_FILE_FUNCTION | 0x80070944 | Invalid command (not supported). |
| 284. | EZNC_FS_IOCTL_FILE_NOTOPEN | 0x80070990 | Not open. |
| 285. | EZNC_FS_IOCTL_FILE_READ | 0x80070994 | Read error. |
| 286. | EZNC_FS_IOCTL_FILE_WRITE | 0x80070995 | Write error. |
| 287. | EZNC_FS_IOCTL_FILE_DATASIZE | 0x80070996 | Application does not fit into prepared buffer. |
| 288. | EZNC_FS_IOCTL_FILE_DATATYPE | 0x80070997 | Invalid data type. |
| 289. | EZNC_FS_IOCTL_FILE_NOTSUPPORTED | 0x80070949 | Not supported (CF not supported). |
| 290. | EZNC_FS_OPEN_DIR_MALLOC | 0x80070A40 | Work area cannot be allocated. |
| 291. | EZNC_FS_OPEN_DIR_BUSY | 0x80070A47 | Directory cannot be opened (during operation). |
| 292. | EZNC_FS_OPEN_DIR_NAMELENGTH | 0x80070A48 | File path is too long. |
| 293. | EZNC_FS_OPEN_DIR_NOTSUPPORTED | 0x80070A49 | Not supported (CF not supported). |
| 294. | EZNC_FS_OPEN_DIR_NODIR | 0x80070A91 | Directory does not exist. |
| 295. | EZNC_FS_OPEN_DIR_NOTOPEN | 0x80070A92 | Already open. |
| 296. | EZNC_FS_OPEN_DIR_FILEFULL | 0x80070A99 | Maximum number of open directories exceeded. |
| 297. | EZNC_FS_OPEN_DIR_NODRIVE | 0x80070A9B | Drive does not exist. |
| 298. | EZNC_FS_READ_DIR_NOTOPEN | 0x80070B90 | Directory is not open. |
| 299. | EZNC_FS_READ_DIR_NODIR | 0x80070B91 | Directory does not exist. |
| 300. | EZNC_FS_READ_DIR_DATASIZE | 0x80070B96 | Application does not fit into prepared buffer. |

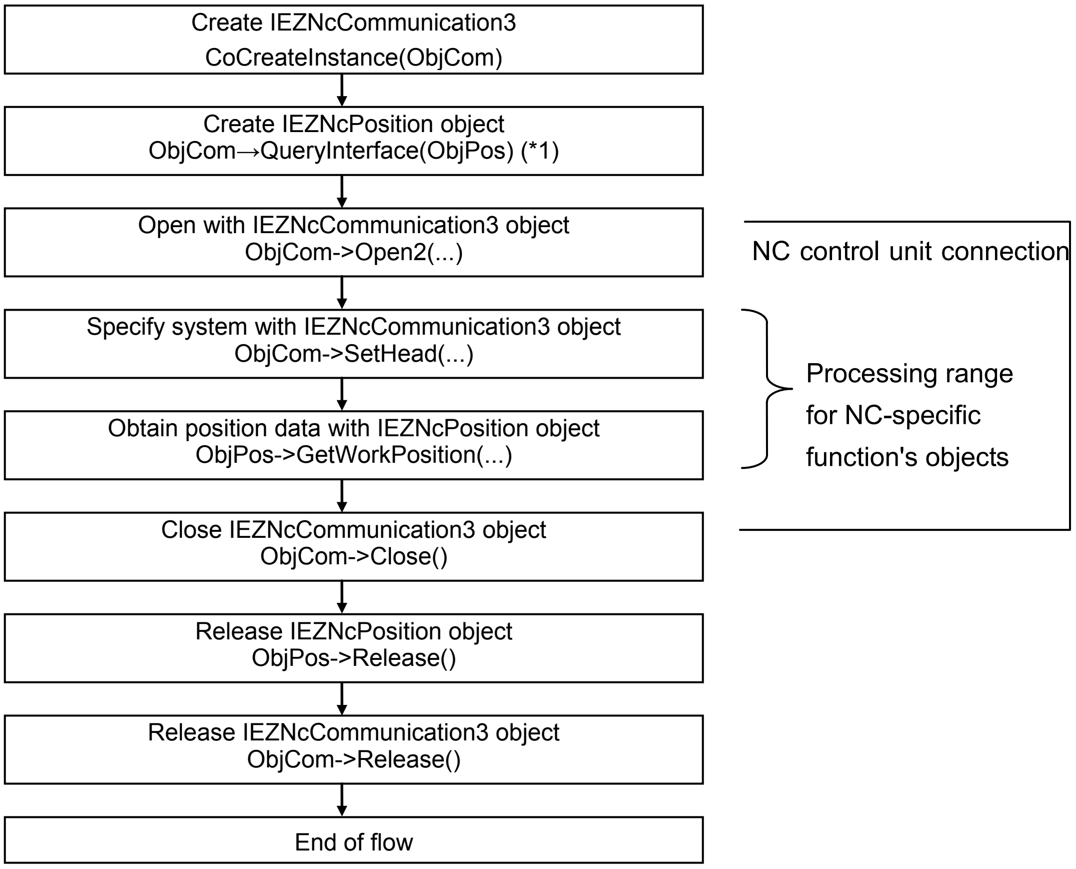
| No. | Error code | Number | Description |
|---|---|---|---|
| 301. | EZNC_FS_CLOSE_DIR_NOTOPEN | 0x80070D90 | Directory is not open. |
| 302. | EZNC_FS_STAT_FILE_NAMELENGTH | 0x80070E48 | File path is too long. |
| 303. | EZNC_FS_STAT_FILE_NOTSUPPORTED | 0x80070E49 | Supported (CF not supported). |
| 304. | EZNC_FS_STAT_FILE_STATERR | 0x80070E94 | File information read error. |
| 305. | EZNC_FS_STAT_FILE_FILEFULL | 0x80070E99 | Maximum number of open files exceeded. |
| 306. | EZNC_FS_STAT_FILE_NODRIVE | 0x80070E9B | Drive does not exist. |
| 307. | EZNC_FS_FSTAT_FILE_NAMELENGTH | 0x80070F48 | File path is too long. |
| 308. | EZNC_FS_FSTAT_FILE_NOTSUPPORTED | 0x80070F49 | Not supported (CF not supported). |
| 309. | EZNC_FS_FSTAT_FILE_STATERR | 0x80070F94 | File information read error. |
| 310. | EZNC_FS_FSTAT_FILE_NOTOPEN | 0x80070F90 | File is not open. |
| 311. | EZNC_FS_FSTAT_FILE_NODRIVE | 0x80070F9B | Drive does not exist. |
| 312. | EZNC_FS_IOCTL_UOPEN_FORMAT | 0x8007099C | Formatting canceled because of invalid SRAM open parameter. |
| 313. | Error codes output by NC control unit. | 0xF00000FF | Invalid argument. |
| 314. | Error codes output by NC control unit. | 0xFFFFFFFF | Data is not readable/writable. |
| 315. | Error codes output by EZSocket (EZSocketPc) for MELSEC programmable controllers (C70 only) | 0x01XXXXXX<br>0x02XXXXXX<br>0x03XXXXXX<br>0x04XXXXXX<br>0x10XXXXXX | For details, refer to the following manuals:<br>• EZSocket Standard, Reference Manual (for MELSEC)(BAD-801Q013)<br>• EZSocket Pro-FX CPU-supported Edition, Reference Manual (for MELSEC)(BAD-801Q025) |

# 4. API OPERATING PROCEDURE

## 4.1 API Operating Procedure

This section provides instructions and procedures for using the product.

Use the product by performing the following steps:



Create IEZNcCommunication3
CoCreateInstance(ObjCom)
Create IEZNcPosition object
ObjCom→QueryInterface(ObjPos) (*1)
Open with IEZNcCommunication3 object
ObjCom->Open2(...)
NC control unit connection
Specify system with IEZNcCommunication3 object
ObjCom->SetHead(...)
Processing range
for NC-specific
function's objects
Obtain position data with IEZNcPosition object
ObjPos->GetWorkPosition(...)
Close IEZNcCommunication3 object
ObjCom->Close()
Release IEZNcPosition object
ObjPos->Release()
Release IEZNcCommunication3 object
ObjCom->Release()
End of flow

## 4.2 Initialization for Enabling OLE/COM Interface

The product uses the OLE/COM interface. Thus, the VC++ project must support OLE/COM. For the project that was created without OLE/COM being enabled, it is possible to enable OLE/COM by modifying appropriate parts of the two files created by VC++, as shown below.

Note that a project name is *Project* in the explanation below.

Project.cpp

```
BOOL CProjectApp::InitInstance()
{
    // Initialize OLE library.
    if (!AfxOleInit())
    {
          // Show error message.
            return FALSE;
    }

     // The rest is omitted.
}
```

Stdafx.h

```
// stdafx.h: Describes standard system include files
//                  or project specific include files that are used frequently
//                  but changed infrequently.
//
#define VC_EXTRALEAN            // Excludes rarely-used stuff from Windows headers.
                                //
#include <afxwin.h>             // MFC core and standard components
#include <afxext.h>             // MFC extensions
#include <afxdisp.h>            // MFC OLE/COM
#ifndef _AFX_NO_AFXCMN_SUPPORT
#include <afxcmn.h>       // MFC support for Windows Common Controls
#endif // _AFX_NO_AFXCMN_SUPPORT
```

## 4.3 Object Creation

The product uses the OLE/COM interface; therefore, objects must be created/released in the thread where OLE/COM initialization was performed. This is not a matter of concern if your program is single-threaded.

In the sample below, the application is for displaying position data, and it is display-centric and single-threaded. Thus, objects are created when the View window is created, and released when the window is closed.

First, create an IEZNcCommunication3 communication object by using CoCreateInstance in the COM library. Then, from the created communication object, create an IEZNcPosition object and other objects by using QueryInterface. The following shows how to create IEZNcCommunication3 and IEZNcPosition objects.

Table 4-1 Creation of IEZNcCommunication3 object

| Creation of IEZNcCommunication3 communication object | |
|---|---|
| Calling procedure | CLSID clsid;<br>IEZNcCommunication pComm;<br>HRESULT hr = CLSIDFromProgID(L"EZSocketNc.EZNcCommunication",&clsid); *1<br>hr = CoCreateInstance( clsid,<br>NULL,<br>CLSCTX_INPROC_SERVER,<br>IID_IEZNcCommunication3,<br>(VOID**)&pComm); |
| Return value | S_OK is returned if the object is successfully created, and if not, another value is returned. |
| Function | Creates a communication object and returns its address in the parameter pComm. |

*1 Refer to *2 in "1.8.1 VC++ program flow (1)".

Table 4-2 Creation of IEZNcPosition object

| Creation of IEZNcPosition parameter object | |
|---|---|
| Calling procedure | IEZNcPosition pPos;<br>HRESUTL hr = pComm->QueryInterface(IID_IEZNcPosition,(void**)&pPos); |
| Return value | S_OK is returned if the object is successfully created, and if not, another value is returned. |
| Function | Creates a parameter object and returns its address in the parameter pPos. |

## 4.4 Include Files

To use the product, include the following header files in the project as necessary.

| | |
|---|---|
| #include "EZSocketNc.h" | ……… Header file for method definitions |
| #include "EZSocketNcStr.h" | ……… Header file for structure definitions (*) |
| #include "EZSocketNcDef.h" | ……… Header file for miscellaneous definitions |
| #include "EZSocketNcErr.h" | ……… Header file for error definitions |
| #include "EasysocketDef.h" | ……… Header file for miscellaneous definitions (*) |

(*) These are necessary when the product is used in the C70.

## 4.5 Overview of VB Programming of Automation Interface

This section explains programming with Microsoft Visual Basic (hereinafter referred to as "VB"). The VC++ programs and VB programs can be written in a similar flow; therefore, VB can be used to create a prototype of your application and verify it at an early stage.

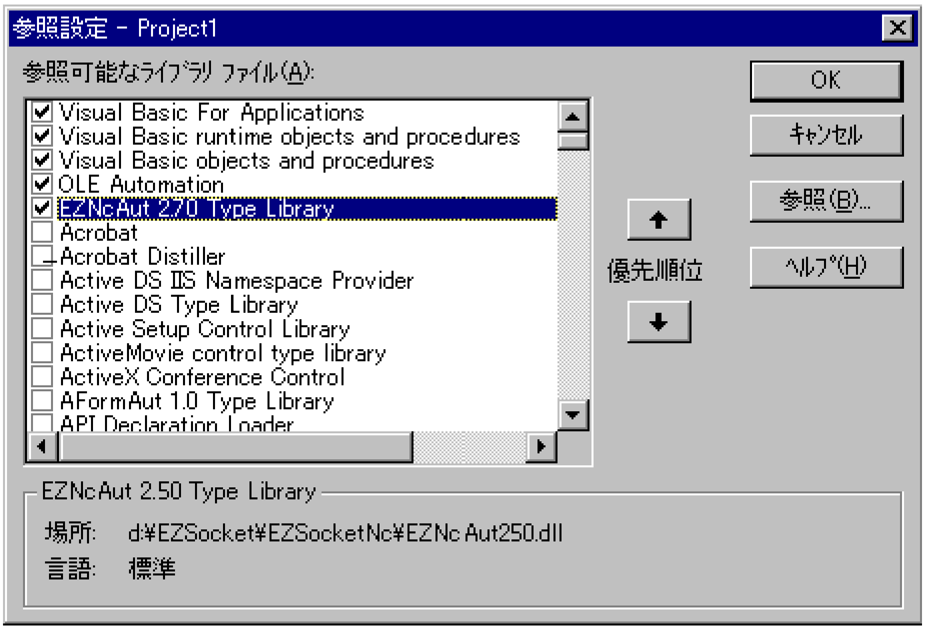
VB's functions that assist programming also help make programming efficient.

### 4.5.1 Using OLE automation interface with VB

(1) Setting references: Check (select) object libraries.

This section describes a way to enable early binding by setting references. Setting references will enable VB's object browsing function.

(Install the product prior to this procedure.)



参照設定 - Project1

参照可能なﾗｲﾌﾞﾗﾘ ﾌｧｲﾙ(A):

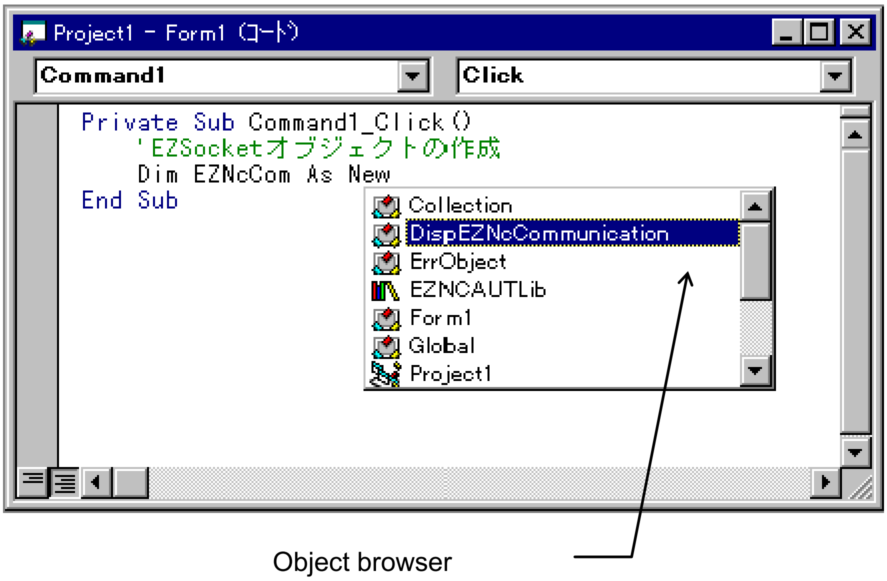
- ☑ Visual Basic For Applications
- ☑ Visual Basic runtime objects and procedures
- ☑ Visual Basic objects and procedures
- ☑ OLE Automation
- ☑ EZNcAut 2.70 Type Library
- □ Acrobat
- □ Acrobat Distiller
- □ Active DS IIS Namespace Provider
- □ Active DS Type Library
- □ Active Setup Control Library
- □ ActiveMovie control type library
- □ ActiveX Conference Control
- □ AFormAut 1.0 Type Library
- □ API Declaration Loader

EZNcAut 2.50 Type Library

場所: d:¥EZSocket¥EZSocketNc¥EZNcAut250.dll

言語: 標準

(2) Object browser: In your programming window, select the EZNcCommunication object.



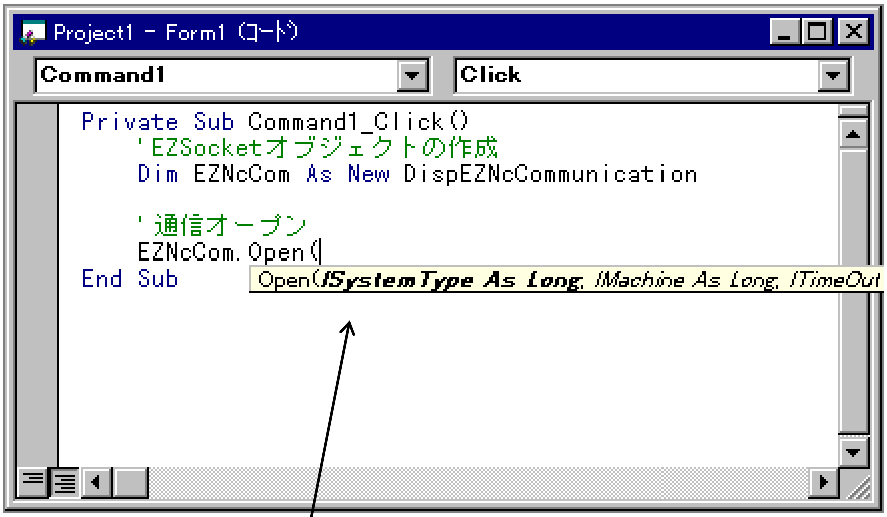
Project1 - Form1 (ｺｰﾄﾞ)

Command1 | Click

```
Private Sub Command1_Click()
    'EZSocketオブジェクトの作成
    Dim EZNcCom As New
End Sub
```

Object browser

(3) Method browser: In your programming window, select a method for the EZNcCom object and check its argument.



```
Private Sub Command1_Click()
    'EZSocketオブジェクトの作成
    Dim EZNcCom As New DispEZNcCommunication

    '通信オープン
    EZNcCom.Open(
End Sub
```

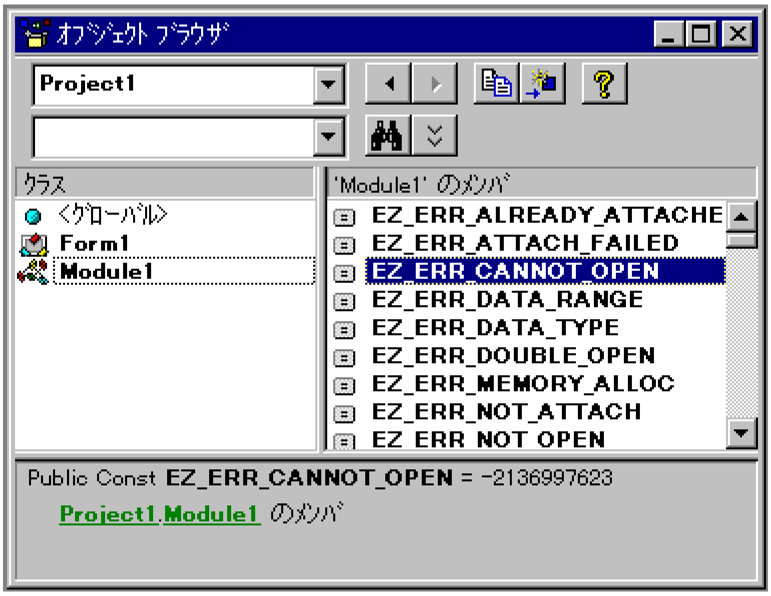
Open(**lSystemType As Long**, lMachine As Long, lTimeOut

Method browser

(4) Module file: Sets definitions and error code references.

Add a module file so that the definitions of the product and error codes can be used in the VB. Select "Project", "Add a standard module", and then "Existing file" to add EZNcDef.bas, EZNcErr.bas, and EZComErr.bas module files to the project.

Definitions and error codes can now be referred to easily through VB's object browser function.



Public Const **EZ_ERR_CANNOT_OPEN** = -2136997623
Project1.Module1 のﾒﾝﾊﾞ

### 4.5.2 VB program flow (1)

This section shows the flow of the program that uses early binding. Reference setting in the product is required.

```
Private Sub Command1_Click()
    'Create object.
    Dim EZNcCom As New DispEZNcCommunication

    'Open communication.
    Dim lRet As Long
    lRet = EZNcCom.Open2(EZNC_SYS_MAGICBOARD64, 1, 1)
    If lRet <> 0 Then GoTo Error_Proc

    '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.
    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 object.
    Set EZNcCom = Nothing
End Sub
```

### 4.5.3 VB program flow (2)

This section shows the flow of the program that uses late binding. Reference setting in the product is not required. Note that the object browser function with VB cannot be used.

```
Private Sub Command1_Click()
    'Create object.
    Dim EZNcCom As Object
    Set EZNcCom = CreateObject("EZNcAut.DispEZNcCommunication","10.20.123.12")

    'Open communication.
    Dim lRet As Long
    lRet = EZNcCom.Open2(EZNC_SYS_MAGICBOARD64, 1, 1)
    If lRet <> 0 Then GoTo Error_Proc

    '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.
    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 object.
    Set EZNcCom = Nothing
End Sub
```

Specify the connection destination's IP address or domain name.

(Note) For the first argument setting of CreateObject(), refer to *2 in "1.8.1 VC++ program flow (1)".

# 5. APPLICATION INSTALLATION PROCEDURE

## 5.1 Overview

To redistribute the application that uses the product and to run it on other computers, it is necessary to copy the user-developed software modules as well as files contained in the product to the computers, and set them properly in their system registry.

This section provides the guideline and the procedure for these steps.

There are two methods to redistribute the product. Choose one that suits your application environment.

(1) Method to use the redistribution installer contained in the enclosed DVD-ROM.

(2) Method to use your own installer created according to the redistribution procedure.

**<<Tips and Precautions for selecting a method>>**

The method (1) above is time saving in that the installer does not need to be created. It is still necessary to add a mechanism to execute the redistribution installer from your application's installer.

For the method (2), the installer created according to the redistribution procedure can be directly and suitably embedded to your application's installer. This method is suitable especially when it is difficult to embed the redistribution installer in your application's installer.

Please be aware that the product may be used by more than one application. When installing/uninstalling your application, please make sure to follow the instructions and procedures specified in this document, to avoid causing problems to the operation of other applications.

**<<Installation specifications for different platforms>>**

The installation specifications of the product are different for different platforms. The table shows the differences in the specifications.

Table 5-1 Installation specifications for different platforms

| Operating environment<br>Specifications | x86 platform | x64 platform |
|---|---|---|
| | EZSocket (32-bit) | EZSocket (32-bit) |
| Destination of installation | Recommended folder (Any folder may be specified.)<br>%ProgramFiles%\EZSocket | Recommended folder (*1)<br>%ProgramFiles%\EZSocket |
| Actual destination of installation | c:\Program Files<br>∟ \EZSocket<br>* This applies when recommended values are specified. | c:\Program Files (x86)<br>∟ \EZSocket<br>* This applies when recommended values are specified. |
| Destination registry | HKEY_LOCAL_MACHINE<br>∟ SOFTWARE<br>∟ MITSUBISHI<br>∟ EZSocketNc | HKEY_LOCAL_MACHINE<br>∟ SOFTWARE<br>∟ Wow6432Node<br>∟ MITSUBISHI<br>∟ EZSocketNc |

(*1) For x64 platform, install the product in the recommended folder.

## 5.2 Distribution Method with Redistribution Installer

This section explains the redistribution method that uses the redistribution installer contained in the enclosed DVD-ROM.

Distribution is easy with the redistribution installer because it has been created in compliance with "5.3 Terms of Redistribution".

### 5.2.1 Location where redistribution installer is stored

The redistribution installer that can be embedded in your product is stored in the following folder on the enclosed DVD-ROM:

EZSocketNc\RedistributableInstaller

### 5.2.2 Destination where redistribution installer is installed

The product is installed in the location specified in the INI file described in the section 5.2.3. Any location may be specified. (Note that this applies only when the product is installed on the target computer for the first time. If the product already exists on the computer, it must be installed in the existing directory and the new installation overwrites the old one.)

For x64 platform, install the product in the recommended folder, C:\Program Files (x86)\EZSocket.

### 5.2.3 Specifications for redistribution installer INI file

This file is used for interaction between your product's installer and the redistribution installer.

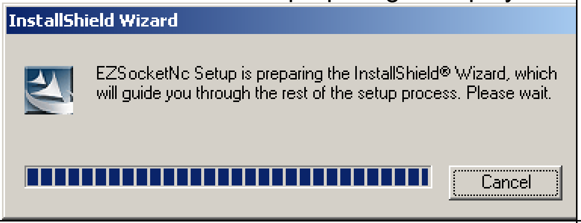
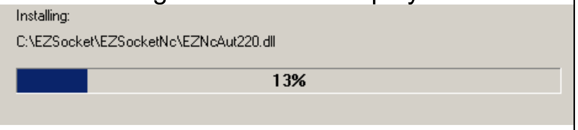
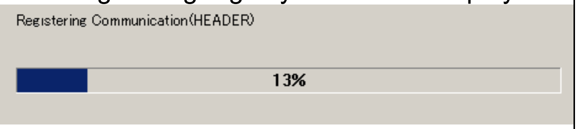
| Section | Key | Description | I/O |
|---|---|---|---|
| USER | Name | User name. | IN |
| | Company | Company name. | |
| SETUP | Target | Installation destination folder.<br>The recommended folder is %ProgramFiles%\EZSocket. If this key is not specified, the product is installed in the recommended folder.<br>Example) C:\Program Files\EZSocket<br>Specify the full path of the folder. (Environment variables cannot be used.) | |
| ERROR | ERROR | Error flag<br>0: No error<br>1: Error occurred | OUT |
| | DETAIL | Error details<br>0: Other error<br>1: Insufficient capacity in installation destination<br>* When ERROR=0, this key is not set. | |
| DOINSTALL | START | 1: Start installation | |
| | END | 1: End installation | |

### 5.2.4 Processing flow and specifications of redistribution installer

This section explains the processing flow and specifications of the redistribution installer.

Make sure to perform thorough operation check when embedding the redistribution installer in your product.

| No. | Processing flow | Specifications |
|---|---|---|
| 1 | Your product's installer is executed. | Your product's installer should:<br><br>Create an EZSNCSET.INI file in the folder that can be managed by your product's installer.<br>The specifications of the EZSNCSET.INI file are as follows:<br><br>[Specifications of EZSNCSET.INI]<br>[USER]<br>Name=user name (maximum 256 bytes)<br>Company=user's company name (maximum 256 bytes)<br>[SETUP]<br>Target=path to installation destination<br><br>[USER] section:<br>Register the Name value under Name in Table 5-2.<br>Register the Company value under Organization in Table 5-2.<br>[SETUP] section:<br>Register the Target value to InstallPath in Table 5-2.<br>However, if InstallPath is already registered, prioritize it.<br>The product is installed in (Target)\EZSocketNc.<br>It is recommended that the following folder be specified as Target:<br>x86 platform:<br>Target=C:\Program Files\EZSocket<br>x64 platform:<br>Target=C:\Program Files (x86)\EZSocket<br><br>[Example of EZSNCSET.INI]<br>[USER]<br>Name=Taro Mitsubishi<br>Company=Mitsubishi Electric Corporation<br>[SETUP]<br>Target=C:\Program Files\EZSocket |
| 2 | Your product's installer executes the redistribution installer. | Execute the redistribution installer's Setup.exe, which is stored on your product's media (for example, DVD-ROM), with the following command line:<br>Setup.exe△full path to where EZSNCSET.INI is located<br>where the symbol △ means a space.<br>Example) Setup.exe C:\temp<br>Place EZSNCSET.INI in the C:\temp folder. |

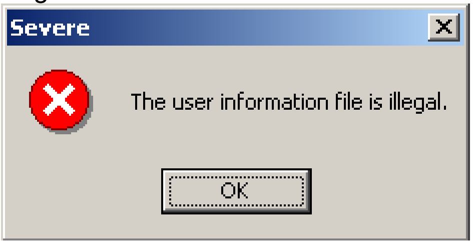
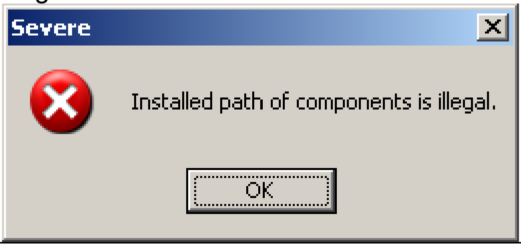
| No. | Processing flow | Specifications |
|---|---|---|
| 3 | The window that indicates that the redistribution installer is preparing is displayed.<br><br>InstallShield Wizard<br>EZSocketNc Setup is preparing the InstallShield® Wizard, which will guide you through the rest of the setup process. Please wait.<br> | The installation preparation window is displayed.<br>If the language of the operating system is other than Japanese, the display language is English. |
| 4 | Make sure that your product's installer refers to the START value of the [DOINSTALL] section of EZSNCSET.INI, and that the redistribution installer has started. (*1) | Set the installation start flag in the EZSNCSET.INI.<br><br>Specifications of [EZSNCSET.INI]<br>[USER]<br>Name=user name (maximum 256 bytes)<br>Company=user's company name (maximum 256 bytes)<br>[SETUP]<br>Target=EZSocket installation folder<br>[DOINSTALL]<br>START=1<br><br>[DOINSTALL] section:<br>The redistribution installer sets the START value to 1 when the installation is started, and the END value to 1 when the installation is<br>completed. |
| 5 | The "Installing...:" window is displayed.<br><br>Installing:<br>C:\EZSocket\EZSocketNc\EZNcAut220.dll<br> | Perform the installation.<br>If the language of the operating system is other than Japanese, the display language is English. |
| 6 | The "Registering registry" window is displayed.<br><br>Registering Communication(HEADER)<br> | Register registry information required for this product according to the installation procedure.<br>If the language of the operating system is other than Japanese, the display language is English. |
| 7 | When the registry registration is completed, the window closes and the installation finishes. | When the registry registration is completed, the "Registering registry" window closes and the installation finishes. |

| | | |
|---|---|---|
| 8 | Make sure that your product's installer refers to the END value of the [DOINSTALL] section of EZSNCSET.INI, and that the redistribution installer is completed. (*1)<br>Also, check the ERROR and DETAIL values in the [ERROR] section of EZSNCSET.INI, and perform post-installation processes.<br>If the redistribution installer is abnormally terminated, resolve the error status, and then execute the installer by following the procedure again. For common errors, refer to 5.2.5 Troubleshooting. | After the installer is finished, record results in EZSNCSET.INI.<br>Specifications of [EZSNCSET.INI]<br>[USER]<br>Name=user name (maximum 256 bytes)<br>Company=user's company name (maximum 256 bytes)<br>[SETUP]<br>Target=Installation destination folder<br>[DOINSTALL]<br>START=1<br>END=1<br>[ERROR]<br>Error=0 *0=completed successfully, 1=terminated abnormallv<br>[ERROR] section:<br>The Error value is set to 0 when the installation is successfully completed, and set to 1 when it is abnormally terminated.<br>When ERROR=1, set error details to the DETAIL key. |
| 9 | Your product's installer deletes EZSNCSET.INI. | EZSNCSET.INI must be deleted at the end because it is a common file. |

(*1) Refer to "5. 2. 6 Precautions".

### 5.2.5 Troubleshooting

This section explains how to handle errors that may occur in the installer.

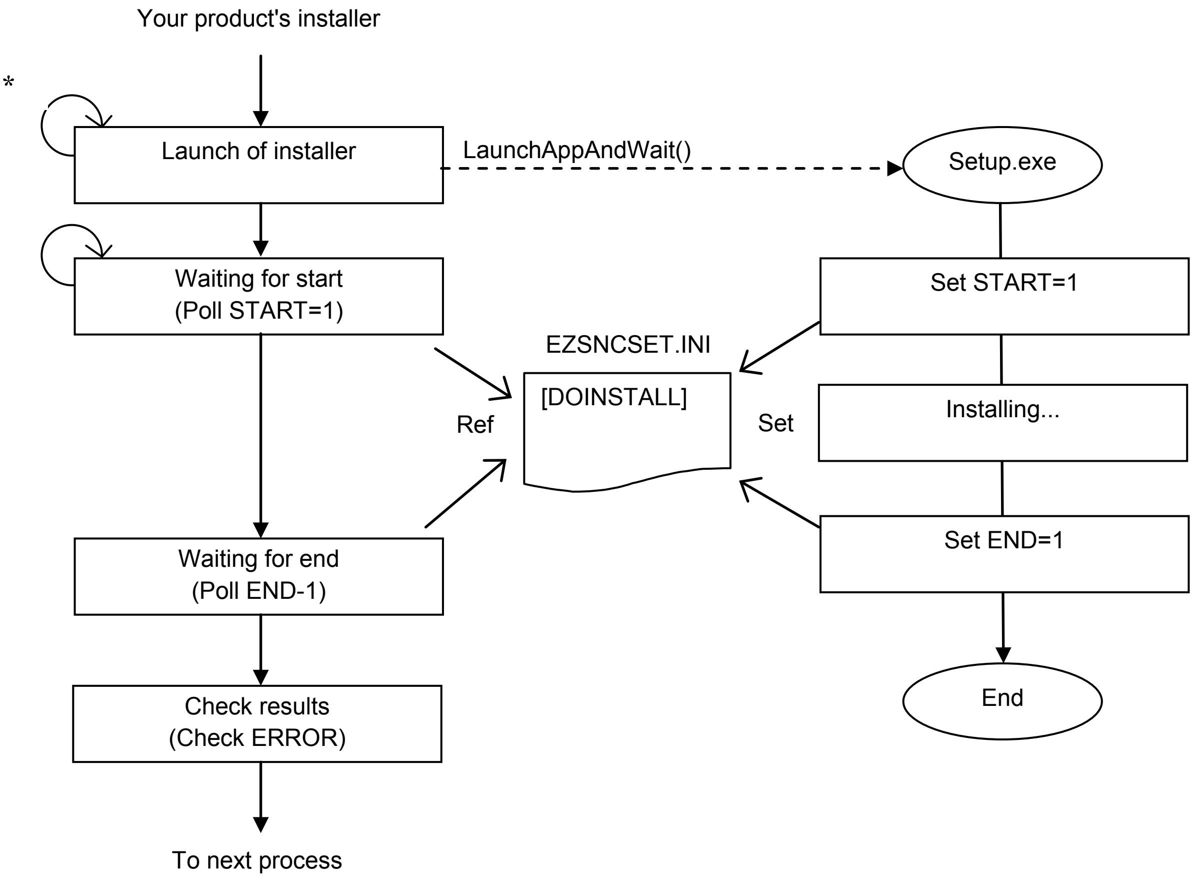
| No. | Error case | Action |
|---|---|---|
| 1. | If EZSNCSET.INI is invalid,<br>the window below is displayed. Click [OK] to close.<br>If the language of the operating system is other than Japanese, the display language is English.<br><br>Severe<br>The user information file is illegal.<br> | [Possible cause]<br>1) User name or user company name cannot be obtained.<br>2) User name/user company name exceeds 256-byte limit.<br><br>[Action]<br>EZSNCSET.INI<br>[ERROR]<br>Error=1<br>Check the contents of EZSNCSET.INI, and set them correctly. Or, place EZSNCSET.INI in the specified folder. |
| 2. | If the path to the installation destination is incorrect,<br>the window below is displayed. Click [OK] to close.<br>If the language of the operating system is other than Japanese, the display language is English.<br><br>Severe<br>Installed path of components is illegal.<br> | [Possible cause]<br>The read path to the installation destination is incorrect.<br>[Action]<br>EZSNCSET.INI<br>[ERROR]<br>Error=1<br>The path specified in Target of EZSNCSET.INI is incorrect. Set the path correctly. |

### 5.2.6 Precautions

**Starting of redistribution installer**

If your product's installer uses the LaunchAppAndWait() function to start the redistribution installer, it may give a return value before the installation of the product is completed. To avoid this, it is necessary to have your product's installer monitor the completion of the redistribution installer.

For this monitoring, use the START and END values in the [DOINSTALL] section of EZSNCSET.INI. For the specifications of the INI file, refer to section 5. 2. 3.



* The installer cannot set the START value in the [DOINSTALL] section if the cancel button is pressed on the InstallShield initial screen. Make sure to set time-outs for waiting for start.

## 5.3 Terms of Redistribution

This section describes the terms of redistribution of the product.

### 5.3.1 Redistributable modules

Modules that can be redistributed are as follows:

- Redistributable files contained in the product

Upon installation of the product on your development machine, these files are installed on the hard disk of the machine. To make sure that the files of the correct version are distributed when redistributing the product, copy the product from the installation DVD-ROM, not from the hard disk, and then embed it in a disk for distribution.

### 5.3.2 Redistributable files

The following files that are stored on the installation DVD-ROM are redistributable. Make sure to redistribute both the custom and automation interfaces. The following files need to be same version; otherwise, an error may occur.

\Lib\EZSocketNc.dll: DLL for custom interface
\Lib\EZNcAutxxx.dll: DLL for automation interface
(where xxx in the file name represents a version-specific number.)
\Lib\CommServer: Related folder
\Lib\Parameter: Related folder
\Lib\Ini\melcfg.ini: Initialization file

## 5.4 Installation Procedure

### 5.4.1 Version upgrade of redistributable files

The redistributable files with the same name but with different versions (older or newer) may be distributed by different applications. In such cases, make sure that the newer version overwrites the older version, not the other way around. Normally the setup program performs version check. If the application does not have a setup program, the user needs to manually check the version before embedding the redistribution program.

(Note 1) If the initialization file melcfg.ini already exists on the personal computer, do not overwrite it however old it may be.

(Note 2) The DLL file name of the automation interface is different for different versions. Do not delete the automation interface file that has already been installed. It will cause the application compatible with the automation interface that has already been installed to fail-to start.

### 5.4.2 x86 platform

#### 5.4.2.1 Installation directory for files

When installing the product for the first time, it can be installed in any directory. For the second or later installation, it must be installed in the directory where the product already exists so that more than one product does not exists on the same computer.

To meet this requirement, follow the procedure below.

(1) Determine if it is the first-time installation or not

Check if the following registry key exists and the correct path to the installation directory is registered as its data.

Registry key: HKEY_LOCAL_MACHINE\SOFTWARE\MITSUBISHI\EZSocketNc\CurrentVersion\InstallPath
Data (Example): C:\Program Files\EZSocket\EZSocketNc

If the correct path name is registered as registry key data, the product is considered to have been installed once or more, and if not, for instance, the registry key does not exist, or the path name is not registered, the product is considered to have never been installed for the first time. Do not add "\" to the end of the path, or related files cannot be read.

(2) Installing for the first time

Have the user specify the installation directory through a dialog or other appropriate method. Register the directory in the registry as installation directory. Other registry settings also need to be configured at this time. For details about registry settings, refer to "5.4.2.3 Registry settings".

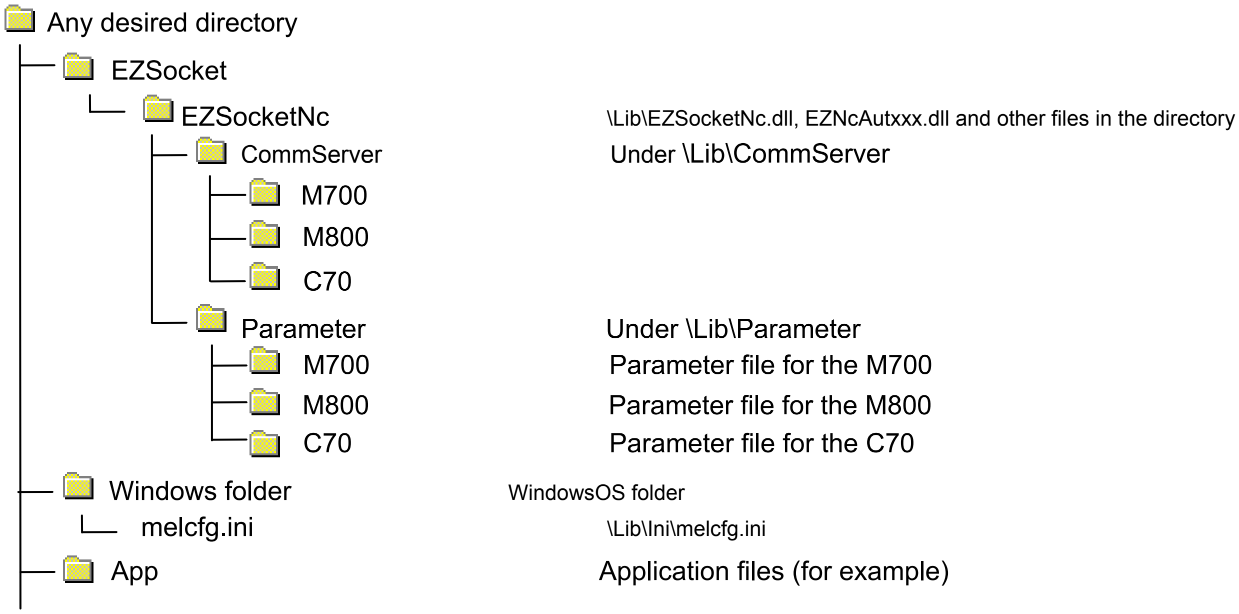
(3) Installing from the second time onward

Install the product in the installation directory that is registered to the registry. If the installation directory does not exist, create one. When copying files, make sure not to overwrite the new version with the old version.

5.4.2.2 Configuration of installation directory

The configuration of the installation directory is shown below:

Copy all files stored on the installation DVD-ROM to the directory as follows:



| Installation directory | Path to the installation files in the installation DVD-ROM |
|---|---|
| Any desired directory | |
| EZSocket | |
| EZSocketNc | \Lib\EZSocketNc.dll, EZNcAutxxx.dll and other files in the directory |
| CommServer | Under \Lib\CommServer |
| M700 | |
| M800 | |
| C70 | |
| Parameter | Under \Lib\Parameter |
| M700 | Parameter file for the M700 |
| M800 | Parameter file for the M800 |
| C70 | Parameter file for the C70 |
| Windows folder | WindowsOS folder |
| melcfg.ini | \Lib\Ini\melcfg.ini |
| App | Application files (for example) |

### 5.4.2.3 Registry settings

Registries required for the product to run are listed below:

When installing the product, create the registry structure as shown, and register data accordingly.

Table 5-2 List of registries

| Key | Name | Type | Data | Remarks |
|---|---|---|---|---|
| HKEY_LOCAL_MACHINE | | | | |
| └ SOFTWARE | | | | |
| └ MITSUBISHI | | | | |
| └ EZSocketNc | | | | |
| ├ CurrentVersion | Description | Character string | "EZSocketNc" | Fixed data. |
| | Organization | Character string | User-specified company name | Register the company name specified by the user at the time of installation. |
| | Name | Character string | User-specified user name | Register the user name specified by the user at the time of installation. |
| | MajorVersion | DWORD value | Version | |
| | MinorVersion | Character string | Version | |
| | InstallPath (Note 1) | Character string | "User-specified directory \EZSocketNc" | Register the path specified by the user at the time of installation. |
| ├ Custom | FileVersion (Note 2) | Character string | Date of EZSocketNc.dll file | YYYY-MM-DD format. |
| | EZSocketNcName | Character string | “EZSocketNc.dll” | Fixed data. |
| └ Automation | FileVersion (Note 2) | Character string | Date of EZNcAut.dll file | YYYY-MM-DD format. |
| | EZSocketNcName | Character string | “EZNcAutxxx.dll” | xxx are numeric characters. |

(Note 1) The data to be registered to "InstallPath" must be "drive: directory specified by the user at the time of package installation + \EZSocketNc".

(Note 2) After copying installation files to the HD, get the time stamp of the specified file, and register this data as "FileVersion".

### 5.4.2.4 System environment variable settings

The system environment variable required for the product to run is shown below.

When installing the product, register it as an additional system environment variable.

The default value is shown in the list below. If the file is not in the specified path, change the path.

Table 5-3 List of system environment variable

| Model | System environment variable (default value) |
|---|---|
| M700/M800 | PATH=installation path of the product<br>(Example: C:\EZSocket\EZSocketNc) |

### 5.4.2.5 COM information registry settings

For EZSocketNc.dll and EZNcAutxxx.dll stored in the installation directory, COM information must be registered to the registry. To register the information, use the redistributable REGSVR32.EXE command, which is shipped with Microsoft Visual C++, at the time of installation. The information is registered as follows:

```
REGSVR32 /s installation directory\EZSocketNc.dll
REGSVR32 /s installation directory\EZNcAutxxx.dll
```

### 5.4.2.6 Precautions for uninstallation

The product may be used by more than one application. If that is the case, and if the product is deleted by uninstalling one of the applications that are installed, the operation of the remaining applications will be affected. To avoid this, do not delete the files and the registry of the product by uninstalling the application in which the product is embedded.

### 5.4.3 x64 platform

#### 5.4.3.1 Installation directory for files

Install the files to %ProgramFiles%\EZSocket.
Register the installation destination directory to the registry as installation directory. Other registry settings also need to be configured at this time. When copying files, make sure not to overwrite the new version with the old version.

Registry key:

HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\MITSUBISHI\EZSocketNc\CurrentVersion\InstallPath

Data (example): C:\Program Files (x86)\EZSocket\EZSocketNc\

#### 5.4.3.2 Configuration of Installation directory

The configuration of the installation directory is shown below:

Copy all files stored on the installation DVD-ROM to the directory as follows:

Installation directory    Path to the installation files in the installation DVD-ROM

```
%ProgramFiles%
├── EZSocket
│    └── EZSocketNc              \Lib\EZSocketNc.dll, EZNcAutxxx.dll and other files in the directory
│         ├── CommServer         Under \Lib\CommServer
│         │    ├── M700
│         │    ├── M800
│         │    └── C70
│         └── Parameter          Under \Lib\Parameter
│              ├── M700          Parameter file for the M700
│              ├── M800          Parameter file for the M800
│              └── C70           Parameter file for the C70
├── Windows folder           WindowsOS folder
│    └── melcfg.ini              \Lib\Ini\melcfg.ini
├── App                          Application files (for example)
│
```

5.4.3.3 Registry settings

Registries required for the product to run are listed below:
When installing the product, create the registry structure as shown, and register data accordingly.

Table 5-4 List of registries

| Key | Name | Type | Data | Remarks |
|---|---|---|---|---|
| HKEY_LOCAL_MACHINE | | | | |
| └ SOFTWARE | | | | |
| └ Wow6432Node | | | | |
| └ MITSUBISHI | | | | |
| └ EZSocketNc | | | | |
| ├CurrentVersion | Description | Character string | "EZSocketNc" | Fixed data. |
| | Organization | Character string | User-specified company name | Register the company name specified by the user at the time of installation. |
| | Name | Character string | User-specified user name | Register the user name specified by the user at the time of installation. |
| | MajorVersion | DWORD value | Version | |
| | MinorVersion | Character string | Version | |
| | InstallPath (Note 1) | Character string | "%ProgramFiles%directory\EZSocket\EZSocketNc" | Register the path specified by the user at the time of installation. |
| ├Custom | FileVersion (Note 2) | Character string | Date of EZSocketNc.dll file | YYYY-MM-DD format. |
| | EZSocketNcName | Character string | “EZSocketNc.dll” | Fixed data. |
| └Automation | FileVersion (Note 2) | Character string | Date of EZNcAut.dll file | YYYY-MM-DD format. |
| | EZSocketNcName | Character string | “EZNcAutxxx.dll” | xxx are numeric characters. |

(Note 1) The data to be registered to "InstallPath" must be "%ProgramFiles%\EZSocket\EZSocketNc".

(Note 2) After copying installation files to the HD, get the time stamp of the specified file, and register this data as "FileVersion".

5.4.3.4 System environment variable settings

The system environment variable required for the product to run is shown below.
When installing the product, register it as an additional system environment variable.
The default value is shown in the list below. If the file is not in the specified path, change the path.

Table 5-5 List of system environment variable

| Model | System environment variable (default value) |
|---|---|
| M700/M800 | PATH=installation path of the product<br>(Example: C:\Program Files (x86)\EZSocket\EZSocketNc) |

#### 5.4.3.5 COM information registry settings

For EZSocketNc.dll and EZNcAutxxx.dll stored in the installation directory, COM information must be registered to the registry. To register the information, use the redistributable REGSVR32.EXE command, which is shipped with Microsoft Visual C++, at the time of installation. The information is registered as follows:

```
REGSVR32 /s installation directory\EZSocketNc.dll
REGSVR32 /s installation directory\EZNcAutxxx.dll
```

#### 5.4.3.6 Precautions for uninstallation

The product may be used by more than one application. If that is the case, and if the product is deleted by uninstalling one of the applications that are installed, the operation of the remaining applications will be affected. To avoid this, do not delete the files and the registry of the product by uninstalling the application in which the product is embedded.

# 6. SAMPLE APPLICATION

## 6.1 Overview of the Sample Application

The sample application that uses this product is provided with compilable project files for Visual C++ Version 6.0 and Visual Basic Version 6.0. The macro sample program using the OLE interface macros that allow custom interfaces to be called easily is also provided. The OLE interface macros are provided as samples.

The sample application includes the following:

- 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 section explains the sample application for Visual C++ Version 6.0 using this product.

### 6.2.1 Operating requirements

The sample application operates in the following system configuration:

| | |
|---|---|
| Operating systems | Windows 2000, Windows XP |
| Compiler | Microsoft Visual C++ Version 6.0 |
| Controller | Mitsubishi CNC C70,<br>Mitsubishi CNC M700/M700V/M70/M70V, M800/M80 |
| H/W | Personal computer on which the operating systems, compiler, and controllers above can be operated |

### 6.2.2 Installation and uninstallation

This section explains installation and uninstallation of the sample application.

For installation of operating systems and VC++ other than the product as well as operations of hardware, refer to the respective instruction manuals.

(1) Installation

The sample application is created in the samples folder when this product is installed.

The sample application has the subfolders with respective project names and each contains its source code and execution file. The sample application includes the Visual C++ 6.0 project workspace files. Opening the corresponding project workspace file enables Visual C++ to open the project.

(2) Uninstallation

To uninstall the sample application, delete the subfolder with the project name or delete the samples folder.

### 6.2.3 Executing the sample application

This section explains execution of the sample application.

The execution file is stored under the Debug folder or the Release folder in the sample application folder. To open the position data display application, execute **Position.exe**.

For instructions for using the position data display application, refer to the following sections.

Note that this sample application is a monitor application for the Mitsubishi CNC. Operations such as operation search and cycle start are required for the computerized numerical controller. For details on the operation methods, refer to the instruction manuals.

### 6.2.4 Function list

This section explains the functions of the sample application.

The position data display application monitors specified position data and displays values obtained as counters.

Table 6-1 Position data display application function list

| | | |
|---|---|---|
| [File] | [Exit application] | Ends the position data display application. |
| [Edit] | [Position data] | Edits the position data type to be displayed.<br>• Current position<br>• Workpiece coordinate position<br>• Machine position<br>• Command remaining distance |
| [Display] | [Refresh cycle] | Edits the refresh cycle for the display. |
| [Communication] | [Communication selection] | Selects a communication target.<br>• CNC C70<br>• CNC M700M (*1)<br>• CNC M700L (*2)<br>• CNC M800M(*3)<br>• CNC M800L(*4) |
| | [Execution] | Starts/Stops communication. |
| [Help] | [Version information] | Displays version information of the position data display application. |

(*1) A communication target is the Mitsubishi CNC machining center system M700/M700V/M70/M70V.

(*2) A communication target is the Mitsubishi CNC lathe system M700/M700V/M70/M70V.

(*3) A communication target is the Mitsubishi CNC machining center system M800/M80.

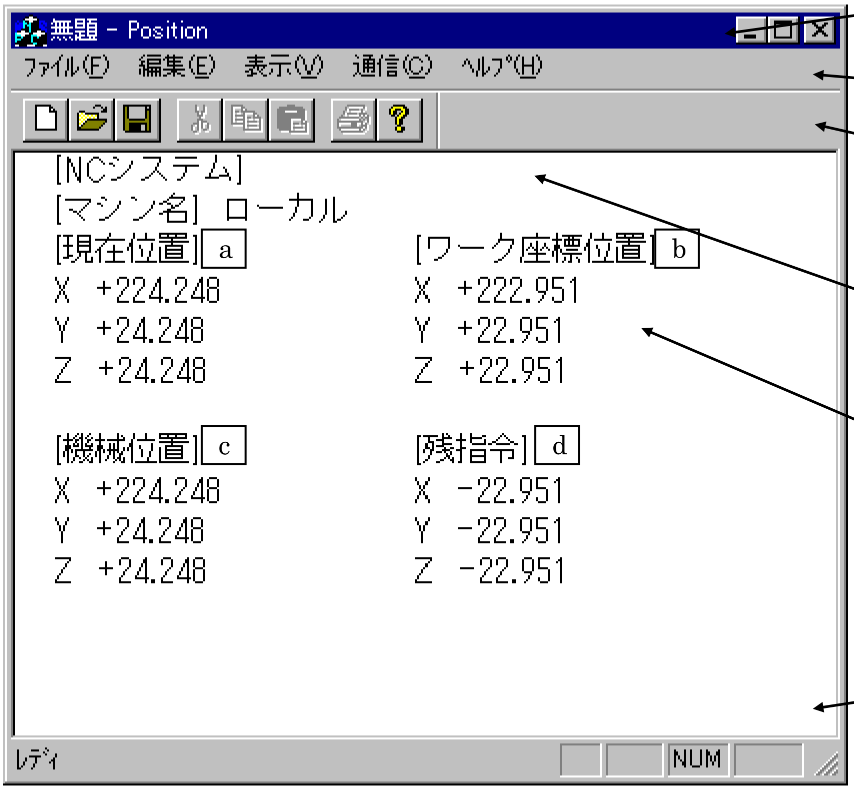
(*4) A communication target is the Mitsubishi CNC lathe system M800/M80

### 6.2.5 Screen structure and functions

This section describes the screen structure for the position data display application and the functions for each menu item.

(1) Basic screen structure

The basic screen is shown below:



```
[NCシステム]
[マシン名] ローカル
[現在位置]              [ワーク座標位置]
X  +224.248             X  +222.951
Y  +24.248              Y  +22.951
Z  +24.248              Z  +22.951

[機械位置]              [残指令]
X  +224.248             X  -22.951
Y  +24.248              Y  -22.951
Z  +24.248              Z  -22.951
```

1) Title bar
(displays the application title)

2) Menu

3) Docking toolbar
(Display/hide is selectable and the file function is not available in the sample.)

4) Communication target display area
Area for the connected machine name

5) Screen display area
a. Displays the current position
b. Displays the workpiece position
c. Displays the machine position
d. Displays the command remaining distance

6) Status line
(Display/hide is selectable.)

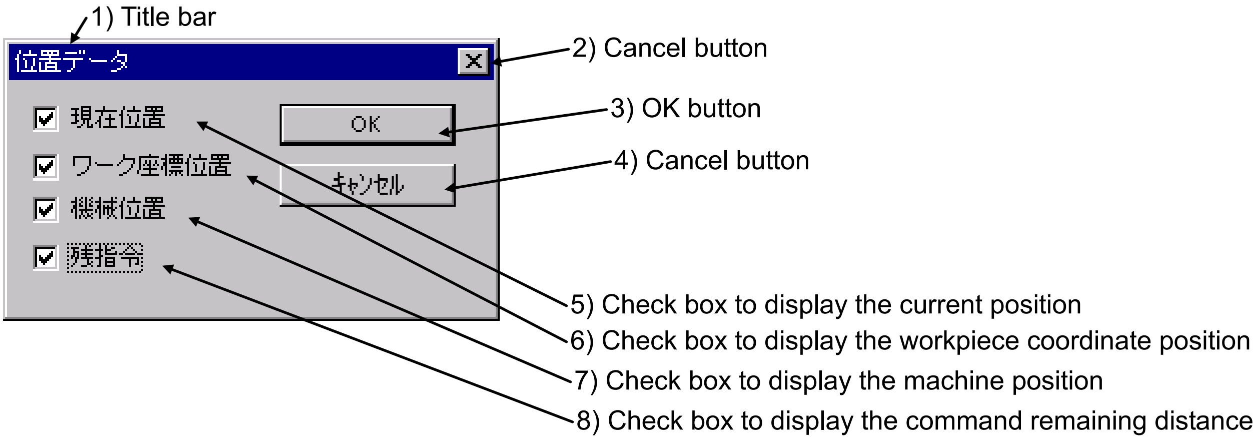
(2) File function

In the position data display application, the available file function is the exit of the application only. There is no file selection function.

(3) Edit function

a. Position data function dialog

Selects the position data types to be displayed.



1) Title bar

2) Cancel button

3) OK button

4) Cancel button

5) Check box to display the current position

6) Check box to display the workpiece coordinate position

7) Check box to display the machine position

8) Check box to display the command remaining distance

1) to 4): Explanation omitted. The following explanation is also omitted.

5) Check box to display the current position:

Selects whether to display or hide the relative position to the position at a completion of the dog type zero point return or to the preset position configured by G92/origin set/counter set.

6) Check box to display the workpiece coordinate position:

Selects whether to display or hide the coordinate position in the current workpiece coordinate system.

7) Check box to display the machine position:

Selects whether to display or hide the coordinate position for each axis in the basic machine coordinate system.

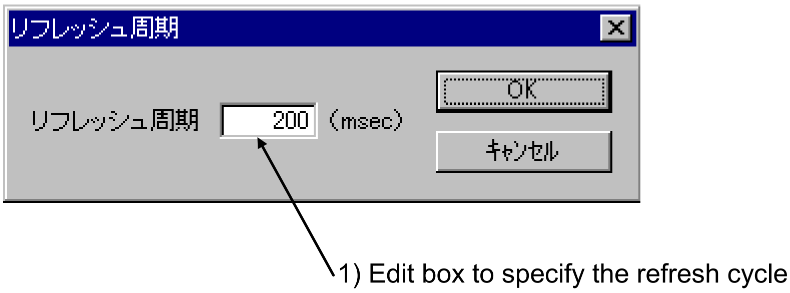
8) Check box to display the command remaining distance:

Selects whether to display or hide the remaining distance for the travel command being executed.

(4) Display function

Refresh cycle function dialog

Sets the refresh cycle for the screen display.



リフレッシュ周期 200 (msec)

1) Edit box to specify the refresh cycle

1) Edit box to specify the refresh cycle:

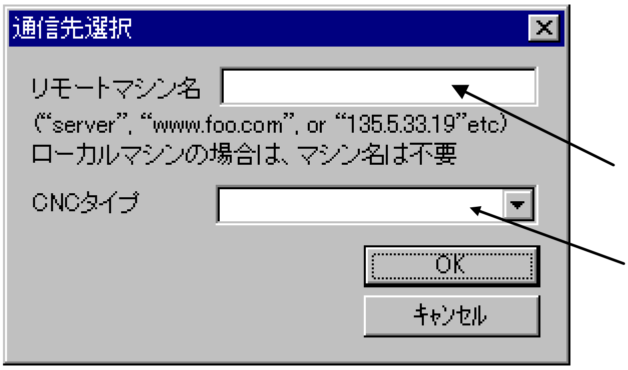
Specifies the refresh cycle for the position data displayed on the screen.

The range is from 200 to 10000 (ms).

(5) Communication function

Communication selection function dialog

Selects a communication target.



("server", "www.foo.com", or "135.5.33.19"etc)
ローカルマシンの場合は、マシン名は不要

1) Combo box for the remotely connected machine

2) Combo box for communication target selection (*1)

1) Combo box for the remotely connected machine:

Sets the machine name of a personal computer equipped with the NC.

Allows the domain name and IP address to be specified.

2) Combo box for the communication target selection:

Sets an NC control module communication target.

Selection range is as follows: MELDASMAGIC64, MELDAS6x5M, MELDAS6x5L, MELDASC6/C64, CNC C70, CNC M700M, CNC M700L, CNC M800M, CNC M800L.

(6) Execution function

Connects/disconnects the selected communication target.

If connection fails, an error message is output and the message box appears.

(7) Version display

"Help - Version display" displays the dialog box for version information of the position data display application.

### 6.2.6 Setting project workspaces

This section explains how to set the project workspace used for creating the position data display application. The application project configuration is as follows.

Table 6-2 Project configuration

| Setting item | Setting value |
|---|---|
| Application type | SDI (Single Document Interface). |
| Database support | Not supported. |
| Automation support | Supported. |
| OLE compound document support | Not supported. |
| Functions to be embedded into the application | Docking toolbar.<br>Initial status bar.<br>3D control. |
| MAPI support | Not supported. |
| Windows Sockets support | Not supported. |
| Number of the latest files to be displayed. | 4 files. (The default value is applied to the advanced settings.) |

### 6.2.7 IEZNcCommunication object

The IEZNcCommunication3 object is used for connection to the communication circuit.
This sample application uses the following methods:

Open2()　……. Open line method
Close() ……… Line disconnect method
SetHead()………………… System specification method

### 6.2.8 IEZNcPosition object

The IEZNcPosition object executes position information acquisition for the opened NC control module. This sample application uses the following methods:

GetWorkPosition()……… Workpiece coordinate position acquisition method
GetMachinePosition()…　Machine position acquisition method
GetCurrentPosition()…　Current position acquisition method
GetDistance()…………… Command remaining distance acquisition method

## 6.3 Monitoring Application

This section explains the sample application for Visual Basic Version 6.0 using this product.

### 6.3.1 Operating requirements

The sample application operates in the following system configuration:

| | |
|---|---|
| Operating systems | Windows 2000, Windows XP |
| Compiler | Microsoft Visual Basic Version 6.0 |
| Controller | Mitsubishi CNC C70,<br>Mitsubishi CNC M700/M700V/M70/M70V, M800/M80 |
| H/W | Personal computer on which the operating systems, compiler, and controllers above can be operated |

### 6.3.2 Installation and uninstallation

This section explains installation and uninstallation of the sample application.

For installation of operating systems and VB other than the product as well as operations of hardware, refer to the respective instruction manual.

(1) Installation

The sample application is created in the samples folder when this product is installed.

The sample application has the subfolders with respective project names and each contains its source code and execution file. The sample application includes the Visual Basic 6.0 project workspace files. Opening the corresponding project workspace file enables Visual Basic to open the project.

(2) Uninstallation

To uninstall the sample application, delete the subfolder with the project name or delete the samples folder.

### 6.3.3 Executing the sample application

This section explains execution of the sample application.

The execution file is stored under the sample application folder. To open the monitoring application, execute **EZNcAutSample.exe**.

For instruction for using the monitoring application, refer to the following sections.

Note that this sample application is a monitor application for the computerized numerical controller. Operations such as operation search and cycle start are required for the computerized numerical controller. For details on the operation methods, refer to the instruction manuals.

### 6.3.4 Function list

This section explains the functions of the sample application.

The monitoring application monitors the currently-running NC program and the current position, and displays the obtained values as counters.

Table 6-3 Monitoring application function list

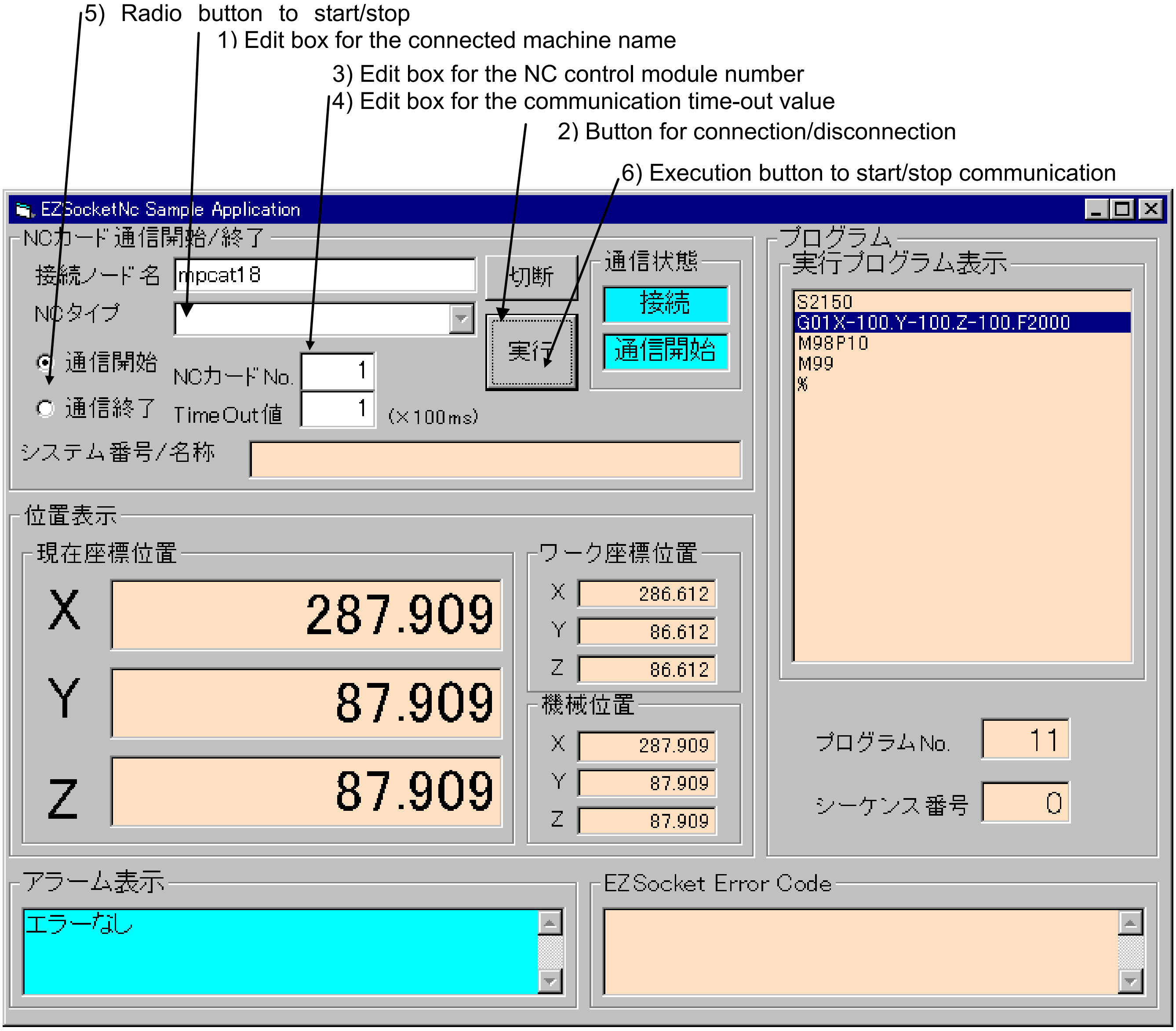
| Function item | Overview |
|---|---|
| NC control module communication start/stop | Sets NC control module communication parameters and starts/stops communication.<br>• Sets the NC control module number.<br>• Sets communication time-out value.<br>• Executes communication.<br>• Displays the NC system version. |
| Position display | Reads the position data.<br>• Displays the current coordinate position.<br>• Displays the workpiece coordinate position.<br>• Displays the machine coordinate position. |
| Alarm display | Displays the current alarm. |
| Program display | Displays the currently-running program and the ongoing line position.<br>• Displays the currently-running program list<br>• Displays the running block position and sequence number. |
| Error display | • Displays the API error codes for this product. |

### 6.3.5 Screen structure and functions

This section describes the screen structure for the position data display application and the functions for each menu item.

(1) Basic screen structure

The basic screen is shown below:


5) Radio button to start/stop
1) Edit box for the connected machine name
3) Edit box for the NC control module number
4) Edit box for the communication time-out value
2) Button for connection/disconnection
6) Execution button to start/stop communication
EZSocketNc Sample Application
切断
実行

NCカード通信開始/終了

接続ノード名 mpcat18

NCタイプ

● 通信開始

○ 通信終了

NCカードNo. 1

TimeOut値 1 (×100ms)

通信状態: 接続 / 通信開始

システム番号/名称

プログラム

実行プログラム表示

```
S2150
G01X-100.Y-100.Z-100.F2000
M98P10
M99
%
```

位置表示

現在座標位置

| | |
|---|---|
| X | 287.909 |
| Y | 87.909 |
| Z | 87.909 |

ワーク座標位置

| | |
|---|---|
| X | 286.612 |
| Y | 86.612 |
| Z | 86.612 |

機械位置

| | |
|---|---|
| X | 287.909 |
| Y | 87.909 |
| Z | 87.909 |

プログラムNo. 11

シーケンス番号 0

アラーム表示: エラーなし

EZSocket Error Code

1) Edit box for the connected machine name

Sets the machine name of a personal computer equipped with the NC.

Allows the domain name and IP address to be specified.

2) Button for connection/disconnection

Connects/disconnects a specified machine.

3) Edit box for the NC control module number

Sets the NC control module number. The NC control module number is determined when setting up the NC control module.

4) Edit box for the communication time-out value

Sets the communication time-out value for the NC control module.

5) Radio button to start/stop communication

Sets to start or stop communication with the NC control module.

6) Execution button to start/stop communication

Executes a communication start or stop after setting 1) to 3).

### 6.3.6 Setting project workspaces

This section explains how to set the project workspace used for creating the position data display application. The application project configuration is as follows.

This project uses late binding to call methods of this product.

Table 6-4 Project configuration

| Setting item | Setting value |
|---|---|
| Application type | Standard EXE |
| Addition of the standard module | Select EZNcDef.bas or EZNcErr.bas from [Add standard module] in [Project]. |

# 7. CONSOLE PROGRAM SAMPLE

## 7.1 Console Program to Connect Mitsubishi CNC C70 (via Ethernet)

```
//
// Simple sample program for 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));

    // COM initialization
    hr = CoInitialize(NULL);
    if( S_OK != hr ){
        wprintf(L"Failed in CoInitialize!\n");
        return 0;
    }

    setlocale( LC_ALL, "Japanese" );
    IEZNcCommunication3     *pIEZNcCom = NULL;      // Communication object
    IEZNcFile6              *pIEZNcFile = NULL;     // File object

    //
    // EZNcCommunication object   creation
    // For the first argument of CLSIDFromProgID(), refer to *2 in "1.8.1 VC++ program flow (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;
    }

    //
    // EZNcFile object creation
    //
    if(pIEZNcCom->QueryInterface(IID_IEZNcFile6, (void**)&pIEZNcFile) != S_OK){
        wprintf(L"EZSocket is not installed!\n");
        goto END;
    }

    //
    // Open parameter setting
    //
    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;
    }

    // IEZNcCommunication3 open
    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;
    }

    //
    // File search
    //
    hr = pIEZNcFile->FindDir2(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->FindNextDir2(&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;
    }
    //
```

```
        // File read
        //
        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;
        }

        // IEZNcCommunication3 close
        if(pIEZNcCom != NULL){
            pIEZNcCom->Close(&lRet);
        }

        // Object release
        if(pIEZNcFile != NULL){
            pIEZNcFile->Release();
            pIEZNcFile = NULL;
        }
        if(pIEZNcCom != NULL){
            pIEZNcCom->Release();
            pIEZNcCom = NULL;
        }

        // COM library release
        CoUninitialize();
        return 0;
}
```

# Revision History

<table>
<tr><th>Date of revision</th><th>Manual No.</th><th>Revision details</th></tr>
<tr><td>Dec.2013</td><td>IB-1501209-A</td><td>First edition created.</td></tr>
<tr><td>Jan. 2014</td><td>IB-1501209-B</td><td>
1. Changed the compatible NC models<br>
Changed the compatible NC models to be M700 and C70<br>
Corrected the description so that only the compatible models are given in each item.<br><br>
2. Added the descriptions and corrections on C70<br>
Added the compatible methods<br>
Added the description and caution on I/F in accordance with the added methods<br>
Added a C70 sample program<br><br>
3. Added the description on the compatibility with 64bit OS<br>
Added the description and caution on x64 platform<br><br>
4. Corrected the method versions
<table>
<tr><th>Before correction</th><th>After correction</th></tr>
<tr><td>IEZNcFile5<br>FindDir<br>FindNextDir</td><td>IEZNcFile6<br>FindDir2<br>FindNextDir2</td></tr>
<tr><td>IEZNcATC2<br>GetMNGReady</td><td>IEZNcATC3<br>GetMNGReady2</td></tr>
<tr><td>IEZNcParameter2<br>GetParameter<br>SetParameter</td><td>IEZNcParameter3<br>GetParameter2<br>SetParameter2</td></tr>
<tr><td>IEZNcSubFunciton<br>ChangeInit</td><td>IEZNcSubFunciton2<br>ChangeInit2</td></tr>
</table>
Applied these version changes also to the other parts of this manual.<br><br>
5. Error codes<br>
Added the related error codes to the I/F description<br>
Revised the list of error codes<br><br>
6. Others<br>
Corrected errors, modified the layout, etc.
</td></tr>
<tr><td>Jul. 2014</td><td>IB-1501209-C</td><td>
1. Corrected the method versions
<table>
<tr><th>Before correction</th><th>After correction</th></tr>
<tr><td>IEZNcParameter3<br>GetParameter2<br>SetParameter2</td><td>IEZNcParameter3<br>GetParameter3<br>SetParameter3</td></tr>
</table>
2. Others<br>
Corrected errors.
</td></tr>
<tr><td>Sept. 2015</td><td>IB-1501208-D</td><td>
1. Added information related to M800 series<br>
2. Added and corrected information for M700 series and C70 series<br>
3. Others<br>
Corrected errors, modified the layout, etc.
</td></tr>
</table>

# Global Service Network

## AMERICA

### MITSUBISHI ELECTRIC AUTOMATION INC. (AMERICA FA CENTER)

**Central Region Service Center**
500 CORPORATE WOODS PARKWAY, VERNON HILLS, ILLINOIS 60061, U.S.A.
TEL: +1-847-478-2500 / FAX: +1-847-478-2650

**Michigan Service Satellite**
ALLEGAN, MICHIGAN 49010, U.S.A.
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
CINCINATTI, OHIO 45201, U.S.A.
TEL: +1-847-478-2500 / FAX: +1-847-478-2650

**Minnesota Service Satellite**
ROGERS, MINNESOTA 55374, U.S.A.
TEL: +1-847-478-2500 / FAX: +1-847-478-2650

**West Region Service Center**
16900 VALLEY VIEW AVE., LAMIRADA, CALIFORNIA 90638, U.S.A.
TEL: +1-714-699-2625 / FAX: +1-847-478-2650

**Northern CA Satellite**
SARATOGA, CALIFORNIA 95070, U.S.A.
TEL: +1-714-699-2625 / FAX: +1-847-478-2650

**Pennsylvania Service Satellite**
PITTSBURG, PENNSYLVANIA 15644, U.S.A.
TEL: +1-732-560-4500 / FAX: +1-732-560-4531

**Connecticut Service Satellite**
TORRINGTON, CONNECTICUT 06790, U.S.A.
TEL: +1-732-560-4500 / FAX: +1-732-560-4531

**South Region Service Center**
1845 SATTELITE BOULEVARD STE. 450, DULUTH, GEORGIA 30097, U.S.A.
TEL +1-678-258-4529 / FAX +1-678-258-4519

**Texas Service Satellites**
GRAPEVINE, TEXAS 76051, U.S.A.
TEL: +1-678-258-4529 / FAX: +1-678-258-4519
HOUSTON, TEXAS 77001, U.S.A.
TEL: +1-678-258-4529 / FAX: +1-678-258-4519

**Tennessee Service Satellite**
Nashville, Tennessee, 37201, U.S.A.
TEL: +1-678-258-4529 / FAX: +1-678-258-4519

**Florida Service Satellite**
WEST MELBOURNE, FLORIDA 32904, U.S.A.
TEL: +1-678-258-4529 / FAX: +1-678-258-4519

**Canada Region Service Center**
4299 14TH AVENUE MARKHAM, ONTARIO L3R OJ2, CANADA
TEL: +1-905-475-7728 / FAX: +1-905-475-7935

**Canada Service Satellite**
EDMONTON, ALBERTA T5A 0A1, CANADA
TEL: +1-905-475-7728 FAX: +1-905-475-7935

**Mexico Region Service Center**
MARIANO ESCOBEDO 69 TLALNEPANTLA, 54030 EDO. DE MEXICO
TEL: +52-55-3067-7500 / FAX: +52-55-9171-7649

**Monterrey Service Satellite**
MONTERREY, N.L., 64720, MEXICO
TEL: +52-81-8365-4171

## BRAZIL

### MELCO CNC do Brasil Comércio e Serviços S.A

**Brazil Region Service Center**
ACESSO JOSE SARTORELLI, KM 2.1 CEP 18550-000, BOITUVA-SP, BRAZIL
TEL: +55-15-3363-9900 / FAX: +55-15-3363-9911

## EUROPE

### MITSUBISHI ELECTRIC EUROPE B.V.

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: + 49-711-770598-123 / FAX: +49-711-770598-141

**France Service Center DEPARTEMENT CONTROLE NUMERIQUE**
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 DEPARTEMENT CONTROLE NUMERIQUE**
120, ALLEE JACQUES MONOD 69800 SAINT PRIEST FRANCE
TEL: +33-1-41-02-83-13 / FAX: +33-1-49-01-07-25

**Italy Service Center**
VIALE COLLEONI, 7 - CENTRO DIREZIONALE COLLEONI PALAZZO SIRIO INGRESSO 1
20864 AGRATE BRIANZA (MB), ITALY
TEL: +39-039-6053-342 / FAX: +39-039-6053-206

**Italy (Padova) Service Satellite**
VIA G. SAVELLI, 24 - 35129 PADOVA, ITALY
TEL: +39-039-6053-342 / FAX: +39-039-6053-206

**U.K. Branch**
TRAVELLERS LANE, HATFIELD, HERTFORDSHIRE, AL10 8XB, U.K.
TEL: +49-2102-486-0 / FAX: +49-2102-486-5910

**Spain Service Center**
CTRA. DE RUBI, 76-80-APDO. 420
08173 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: +48-12-630-4701

**Mitsubishi Electric Turkey A.Ş Ümraniye Şubesi**
**Turkey Service Center**
ŞERIFALI MAH. NUTUK SOK. NO.5 34775
ÜMRANIYE, ISTANBUL, TURKEY
TEL: +90-216-526-3990 / FAX: +90-216-526-3995

**Czech Republic Service Center**
KAFKOVA 1853/3, 702 00 OSTRAVA 2, 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

**MITSUBISHI ELECTRIC EUROPE B.V. (SCANDINAVIA)**
**Sweden Service Center**
HAMMARBACKEN 14 191 49 SOLLENTUNA, SWEDEN
TEL: +46-8-6251000 / FAX: +46-8-966877

**Bulgaria Service Center**
4 A.LYAPCHEV BOUL., POB 21, BG-1756 SOFIA, BULGARIA
TEL: +359-2-8176009 / FAX: +359-2-9744061

**Ukraine (Kharkov) Service Center**
APTEKARSKIY LANE 9-A, OFFICE 3, 61001 KHARKOV, UKRAINE
TEL: +380-57-732-7774 / FAX: +380-57-731-8721

**Ukraine (Kiev) Service Center**
4-B, M. RASKOVOYI STR., 02660 KIEV, UKRAINE
TEL: +380-44-494-3355 / FAX: +380-44-494-3366

**Belarus Service Center**
OFFICE 9, NEZAVISIMOSTI PR.177, 220125 MINSK, BELARUS
TEL: +375-17-393-1177 / FAX: +375-17-393-0081

**South Africa Service Center**
5 ALBATROSS STREET, RHODESFIELD, KEMPTON PARK 1619, GAUTENG, SOUTH AFRICA
TEL: +27-11-394-8512 / FAX: +27-11-394-8513

## 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

**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**
17 & 17A, JALAN IMPIAN EMAS 5/5, TAMAN IMPIAN EMAS, 81300 SKUDAI, JOHOR MALAYSIA.
TEL: +60-7-557-8218 / FAX: +60-7-557-3404

**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

## VIETNAM

### MITSUBISHI ELECTRIC VIETNAM CO.,LTD

**Vietnam (Ho Chi Minh) Service Center**
UNIT 01-04, 10TH FLOOR, VINCOM CENTER 72 LE THANH TON STREET, DISTRICT 1,
HO CHI MINH CITY, VIETNAM
TEL: +84-8-3910 5945 / FAX: +84-8-3910 5946

**Vietnam (Hanoi) Service Satellite**
6th Floor, Detech Tower, 8 Ton That Thuyet Street, My Dinh 2 Ward, Nam Tu Liem District, Hanoi,Vietnam
TEL: +84-4-3937-8075 / FAX: +84-4-3937-8076

## INDONESIA

### PT. MITSUBISHI ELECTRIC INDONESIA

**Indonesia Service Center ( Cikarang Office )**
JL.Kenari Raya Blok G2-07A Delta Silicon 5, Lippo Cikarang-Bekasi 17550, INDONESIA
TEL: +62-21-2961-7797 / FAX: +62-21-2961-7794

## THAILAND

### MITSUBISHI ELECTRIC FACTORY AUTOMATION (THAILAND) CO.,LTD

**Thailand Service Center**
12TH FLOOR, SV.CITY BUILDING, OFFICE TOWER 1, NO. 896/19 AND 20 RAMA 3 ROAD,
KWAENG BANGPONGPANG, KHET YANNAWA, BANGKOK 10120,THAILAND
TEL: +66-2-682-6522-31 / FAX: +66-2-682-6020

## INDIA

### MITSUBISHI ELECTRIC INDIA PVT. LTD.

**India Service Center**
2nd FLOOR, TOWER A & B, DLF CYBER GREENS, DLF CYBER CITY,
DLF PHASE-III, GURGAON 122 002, HARYANA, INDIA
TEL: +91-124-4630 300 / FAX: +91-124-4630 399
**Ludhiana satellite office**
**Jamshedpur satellite office**

**India (Pune) Service Center**
EMERALD HOUSE, EL-3, J-BLOCK, MIDC BHOSARI. PUNE – 411 026, MAHARASHTRA, INDIA
TEL: +91-20-2710 2000 / FAX: +91-20-2710 2100
**Baroda satellite office**
**Mumbai satellite office**

**India (Bangalore) Service Center**
PRESTIGE EMERALD, 6TH FLOOR, MUNICIPAL NO. 2,
LAVELLE ROAD, BANGALORE - 560 043, KAMATAKA, INDIA
TEL: +91-80-4020-1600 / FAX: +91-80-4020-1699
**Chennai satellite office**
**Coimbatore satellite office**

## OCEANIA

### MITSUBISHI ELECTRIC AUSTRALIA LTD.

**Australia 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 (CHINA) LTD. (CHINA FA CENTER)

**China (Shanghai) Service Center**
1-3,5-10,18-23/F, NO.1386 HONG QIAO ROAD, CHANG NING QU,
SHANGHAI 200336, CHINA
TEL: +86-21-2322-3030 / FAX: +86-21-2308-3000
**China (Ningbo) Service Dealer**
**China (Wuxi) Service Dealer**
**China (Jinan) Service Dealer**
**China (Hangzhou) Service Dealer**
**China (Wuhan) Service Satellite**

**China (Beijing) Service Center**
9/F, OFFICE TOWER 1, HENDERSON CENTER, 18 JIANGUOMENNEI DAJIE,
DONGCHENG DISTRICT, BEIJING 100005, CHINA
TEL: +86-10-6518-8830 / FAX: +86-10-6518-8030
**China (Beijing) Service Dealer**

**China (Tianjin) Service Center**
UNIT 2003, TIANJIN CITY TOWER, NO 35 YOUYI ROAD, HEXI DISTRICT,
TIANJIN 300061, CHINA
TEL: +86-22-2813-1015 / FAX: +86-22-2813-1017
**China (Shenyang) Service Satellite**
**China (Changchun) Service Satellite**

**China (Chengdu) Service Center**
ROOM 407-408, OFFICE TOWER AT SHANGRI-LA CENTER, NO. 9 BINJIANG DONG ROAD,
JINJIANG DISTRICT, CHENGDU, SICHUAN 610021, CHINA
TEL: +86-28-8446-8030 / FAX: +86-28-8446-8630

**China (Shenzhen) Service Center**
ROOM 2512-2516, 25/F., GREAT CHINA INTERNATIONAL EXCHANGE SQUARE, JINTIAN RD.S.,
FUTIAN DISTRICT, SHENZHEN 518034, CHINA
TEL: +86-755-2399-8272 / FAX: +86-755-8218-4776
**China (Xiamen) Service Dealer**
**China (Dongguan) Service Dealer**

## KOREA

### MITSUBISHI ELECTRIC AUTOMATION KOREA CO., LTD. (KOREA FA CENTER)

**Korea Service Center**
8F, Gangseo Hangang Xi-tower, 401 Yangcheon-ro, Gangseo-gu, Seoul 157-801, KOREA
TEL: +82-2-3660-9602 / FAX: +82-2-3664-8668

**Korea Taegu Service Satellite**
4F KT BUILDING, 1630 SANGYEOK-DONG, BUK-KU, DAEGU 702-835, KOREA
TEL: +82-53-382-7400 / FAX: +82-53-382-7411

## TAIWAN

### MITSUBISHI ELECTRIC TAIWAN CO., LTD. (TAIWAN FA CENTER)

**Taiwan (Taichung) Service Center (Central Area)**
NO.8-1, INDUSTRIAL 16TH RD., TAICHUNG INDUSTRIAL PARK, SITUN DIST.,
TAICHUNG CITY 40768, TAIWAN R.O.C.
TEL: +886-4-2359-0688 / FAX: +886-4-2359-0689

**Taiwan (Taipei) Service Center (North Area)**
10F, NO.88, SEC.6, CHUNG-SHAN N. RD., SHI LIN DIST., TAIPEI CITY 11155, TAIWAN R.O.C.
TEL: +886-2-2833-5430 / FAX: +886-2-2833-5433

**Taiwan (Tainan) Service Center (South Area)**
11F-1., NO.30, ZHONGZHENG S. ROAD, YONGKANG DISTRICT, TAINAN CITY 71067, TAIWAN, R.O.C.
TEL: +886-6-252-5030 / FAX: +886-6-252-5031

## 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

MITSUBISHI Communication Software for CNC

MITSUBISHI ELECTRIC CORPORATION
HEAD OFFICE : TOKYO BLDG.,2-7-3 MARUNOUCHI,CHIYODA-KU,TOKYO 100-8310,JAPAN

| | |
|---|---|
| MODEL | FCSB1224W000 |
| MODEL CODE | – |
| Manual No. | IB-1501209 |

Made in JAPAN

Specifications are subject to change without notice.