MELSOFT

Engineering Software

MX Component for Android Version 1
Reference Manual

-SW1DNC-ACTAND-B

SAFETY PRECAUTIONS

(Read these precautions before using this product.)

Before using this product, please read this manual carefully and pay full attention to safety to handle the product correctly. If products are used in a different way from that specified by manufacturers, the protection function of the products may not work properly.

The precautions given in this manual are concerned with this product only. For the safety precautions for the programmable controller system, refer to the user's manual for the CPU module used and the system manual, (MELSEC iQ-R Module Configuration Manual, QCPU User's Manual(Hardware Design, Maintenance and Inspection), MELSEC-L CPU Module User's Manual (Hardware Design, Maintenance and Inspection)).

In this manual, the safety precautions are classified into two levels: " ⚠ WARNING" and " ⚠ CAUTION".

⚠ **WARNING**	Indicates that incorrect handling may cause hazardous conditions, resulting in death or severe injury.
⚠ **CAUTION**	Indicates that incorrect handling may cause hazardous conditions, resulting in minor or moderate injury or property damage.

Under some circumstances, failure to observe the precautions given under " ⚠ CAUTION" may lead to serious consequences.

Observe the precautions of both levels because they are important for personal and system safety.

Make sure that the end users read this manual and then keep the manual in a safe place for future reference.

[Design Precautions]

⚠WARNING

● When data change or status control are performed from a device such as a tablet to a running programmable controller CPU, create an interlock circuit outside the programmable controller to ensure that the whole system always operates safely.
Since a wireless LAN is used for communication, the communication may not be performed properly depending on the environment. Ensure the communication method other than using this product for the situation when the wireless LAN communication cannot be established.

[Startup and Maintenance Precautions]

⚠CAUTION

● The online operations performed from a device such as a tablet to a running CPU module have to be executed after the manual has been carefully read and the safety has been ensured.
Improper operation may damage machines or cause accidents.

[Security Precautions]

⚠WARNING

● To maintain the security (confidentiality, integrity, and availability) of the programmable controller and the system against unauthorized access, denial-of-service (DoS) attacks, computer viruses, and other cyberattacks from external devices via the network, take appropriate measures such as firewalls, virtual private networks (VPNs), and antivirus solutions.

CONDITIONS OF USE FOR THE PRODUCT

(1) MELSEC programmable controller ("the PRODUCT") shall be used in conditions;

i) where any problem, fault or failure occurring in the PRODUCT, if any, shall not lead to any major or serious accident; and

ii) where the backup and fail-safe function are systematically or automatically provided outside of the PRODUCT for the case of any problem, fault or failure occurring in the PRODUCT.

(2) The PRODUCT has been designed and manufactured for the purpose of being used in general industries.
MITSUBISHI ELECTRIC SHALL HAVE NO RESPONSIBILITY OR LIABILITY (INCLUDING, BUT NOT LIMITED TO ANY AND ALL RESPONSIBILITY OR LIABILITY BASED ON CONTRACT, WARRANTY, TORT, PRODUCT LIABILITY) FOR ANY INJURY OR DEATH TO PERSONS OR LOSS OR DAMAGE TO PROPERTY CAUSED BY the PRODUCT THAT ARE OPERATED OR USED IN APPLICATION NOT INTENDED OR EXCLUDED BY INSTRUCTIONS, PRECAUTIONS, OR WARNING CONTAINED IN MITSUBISHI ELECTRIC USER'S, INSTRUCTION AND/OR SAFETY MANUALS, TECHNICAL BULLETINS AND GUIDELINES FOR the PRODUCT.
("Prohibited Application")

Prohibited Applications include, but not limited to, the use of the PRODUCT in;

- Nuclear Power Plants and any other power plants operated by Power companies, and/or any other cases in which the public could be affected if any problem or fault occurs in the PRODUCT.
- Railway companies or Public service purposes, and/or any other cases in which establishment of a special quality assurance system is required by the Purchaser or End User.
- Aircraft or Aerospace, Medical applications, Train equipment, transport equipment such as Elevator and Escalator, Incineration and Fuel devices, Vehicles, Manned transportation, Equipment for Recreation and Amusement, and Safety devices, handling of Nuclear or Hazardous Materials or Chemicals, Mining and Drilling, and/or other applications where there is a significant risk of injury to the public or property.

Notwithstanding the above restrictions, Mitsubishi Electric may in its sole discretion, authorize use of the PRODUCT in one or more of the Prohibited Applications, provided that the usage of the PRODUCT is limited only for the specific applications agreed to by Mitsubishi Electric and provided further that no special quality assurance or fail-safe, redundant or other safety features which exceed the general specifications of the PRODUCTs are required. For details, please contact the Mitsubishi Electric representative in your region.

(3) Mitsubishi Electric shall have no responsibility or liability for any problems involving programmable controller trouble and system trouble caused by DoS attacks, unauthorized access, computer viruses, and other cyberattacks.

INTRODUCTION

Thank you for purchasing the engineering software, MELSOFT series.

This manual describes the operations of MX Component for Android.

Before using this product, please read this manual carefully, and develop familiarity with the functions and performance of MX Component for Android to handle the product correctly.

CONTENTS

CONTENTS

5

RELEVANT MANUALS

Manual name [manual number]	Description	Available form
MX Component for Android Version 1 Reference Manual [SH-081498ENG] (this manual)	Explains the system configuration, operation methods, and methods of MX Component for Android.	e-Manual PDF

Point👉

e-Manual refers to the Mitsubishi Electric FA electronic book manuals that can be browsed using a dedicated tool.

e-Manual has the following features:
- Required information can be cross-searched in multiple manuals.
- Other manuals can be accessed from the links in the manual.
- Hardware specifications of each part can be found from the product figures.
- Pages that users often browse can be bookmarked.

TERMS

Unless otherwise specified, this manual uses the following terms.

Term	Description
Built-in Ethernet CPU	A generic term for CPU modules with an Ethernet port.
C Controller module	A generic term for MELSEC iQ-R series C Controller modules and MELSEC-Q series C Controller modules.
Engineering tool	A tool for setting, programming, debugging, and maintaining programmable controllers. A generic term for GX Works2, GX Works3, and Setting/monitoring tools for the C Controller module.
Ethernet module	A generic term for MELSEC iQ-R series-compatible EN71, MELSEC-Q series-compatible E71, and MELSEC-L series-compatible E71.
FX5CPU	A generic term for FX5UCPU and FX5UCCPU.
GX Works2	A generic product name for SWnDND-GXW2 and SWnDNC-GXW2. ('n' indicates its version.)
GX Works3	A generic product name for SWnDND-GXW3. ('n' indicates its version.)
LCPU	A generic term for L02SCPU, L02SCPU-P, L02CPU, L02CPU-P, L06CPU, L06CPU-P, L26CPU, L26CPU-P, L26CPU-BT, and L26CPU-PBT.
MELSEC iQ-R series C Controller module	R12CCPU-V
MELSEC iQ-R series CPU	A generic term for MELSEC iQ-R series programmable controller CPUs, C Controller modules, and motion CPUs.
MELSEC iQ-R series motion CPU	A generic term for R16MTCPU and R32MTCPU.
MELSEC iQ-R series-compatible EN71	RJ71EN71 (when using the Ethernet function)
MELSEC-L series-compatible E71	LJ71E71-100
MELSEC-Q series C Controller module	A generic term for Q12DCCPU-V (Extended mode), Q24DHCCPU-V, and Q24DHCCPU-LS.
MELSEC-Q series CPU	A generic term for the MELSEC-Q series programmable controller CPUs, C Controller modules, and motion CPUs.
MELSEC-Q series motion CPU	A generic term for Q172DCPU, Q173DCPU, Q172DSCPU, and Q173DSCPU.
MELSEC-Q series-compatible E71	A generic term for QJ71E71-100, QJ71E71-B5, and QJ71E71-B2.
Motion CPU	A generic term for MELSEC iQ-R series motion CPUs and MELSEC-Q series motion CPUs.
MX Component for Android	A generic product name for SWnDNC-ACTAND-B. ('n' indicates its version.)
Programmable controller CPU	A generic term for RCPUs, FX5CPU, QCPUs, and LCPUs.
QCPU	A generic term for Q00JCPU, Q00UJCPU, Q00CPU, Q00UCPU, Q01CPU, Q01UCPU, Q02CPU, Q02HCPU, Q02PHCPU, Q02UCPU, Q03UDCPU, Q03UDECPU, Q03UDVCPU, Q04UDHCPU, Q04UDEHCPU, Q04UDVCPU, Q06HCPU, Q06PHCPU, Q06UDHCPU, Q06UDEHCPU, Q06UDVCPU, Q10UDHCPU, Q10UDEHCPU, Q12HCPU, Q12PHCPU, Q12PRHCPU, Q13UDHCPU, Q13UDEHCPU, Q13UDVCPU, Q20UDHCPU, Q20UDEHCPU, Q25HCPU, Q25PHCPU, Q25PRHCPU, Q26UDHCPU, Q26UDEHCPU, Q26UDVCPU, Q50UDEHCPU, and Q100UDEHCPU.
RCPU	A generic term for RnCPUs, RnENCPUs, and RnPCPUs.
RnCPU	A generic term for R04CPU, R08CPU, R16CPU, R32CPU, and R120CPU.
RnENCPU	A generic term for R04ENCPU, R08ENCPU, R16ENCPU, R32ENCPU, and R120ENCPU.
RnPCPU	A generic term for R08PCPU, R16PCPU, R32PCPU, and R120PCPU.
Setting/monitoring tools for the C Controller module	A generic product name for SW4PVC-CCPU.

Term	Description
Universal model QCPU	A generic term for Q00UJCPU, Q00UCPU, Q01UCPU, Q02UCPU, Q03UDCPU, Q03UDVCPU, Q03UDECPU, Q04UDHCPU, Q04UDVCPU, Q04UDPVCPU, Q04UDEHCPU, Q06UDHCPU, Q06UDVCPU, Q06UDPVCPU, Q06UDEHCPU, Q10UDHCPU, Q10UDEHCPU, Q13UDHCPU, Q13UDVCPU, Q13UDPVCPU, Q13UDEHCPU, Q20UDHCPU, Q20UDEHCPU, Q26UDHCPU, Q26UDVCPU, Q26UDPVCPU, Q26UDEHCPU, Q50UDEHCPU, and Q100UDEHCPU.

1 OVERVIEW

This chapter explains the features of MX Component for Android.

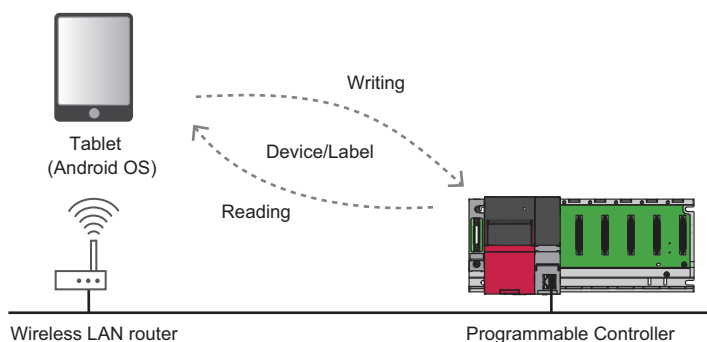## 1.1 What is MX Component for Android?

MX Component for Android is the library to read and write the values of devices and labels in a programmable controller from a tablet.

This product is for creating a user program to communicate with a programmable controller from Android OS.

Wireless communication performs between a device such as a tablet and wireless LAN router, and Ethernet communication performs to access a programmable controller.

By using MX Component for Android when creating an application, the time to create a program for communication with a programmable controller can be shortened.

It can also be used for the static library of Java.

1.2 Main Functions

MX Component for Android has the following main functions.

Creating a tablet application

An application for a tablet to read and write devices and labels in a programmable controller can easily be created without knowing the communication method for the programmable controller.

Programming by using labels (RCPUs only)

When the communication target is an RCPU, the devices in the programmable controller can be accessed by using labels[1]. Therefore, the program is easy to see and recreation of an application is not required even after the device configuration was changed.

 *1   Labels selected in "Access from External Device" in the global editor in GX Works3 can be used.

Creating a safe application for security

A safe application for security can be created by using this library since a remote password is encrypted[1] when communicating with a programmable controller in which the remote password has been set.

 *1   A remote password is encrypted in the following cases only.
    ·When accessing a MELSEC iQ-R series programmable controller
    ·When accessing a MELSEC-Q series CPU or an LCPU via an Ethernet module

Sample program

An application can be created by using the provided sample program as a reference. The sample program also is helpful to see how to use the library or correct an error etc.

MEMO

2 SYSTEM CONFIGURATION

This chapter explains the system configuration of MX Component for Android.

2.1 System Configuration

The following shows the system configurations for development and operation.

For development

Development personal computer (Windows OS)

Install the created application.

Tablet (Android OS)

Wireless LAN router

Programmable controller CPU/C Controller module/motion CPU

Built-in Ethernet CPU/ Ethernet module

For operation

Tablet (Android OS)

Wireless LAN router

Hub

Hub

Programmable controller CPU/C Controller module/motion CPU

Built-in Ethernet CPU/ Ethernet module
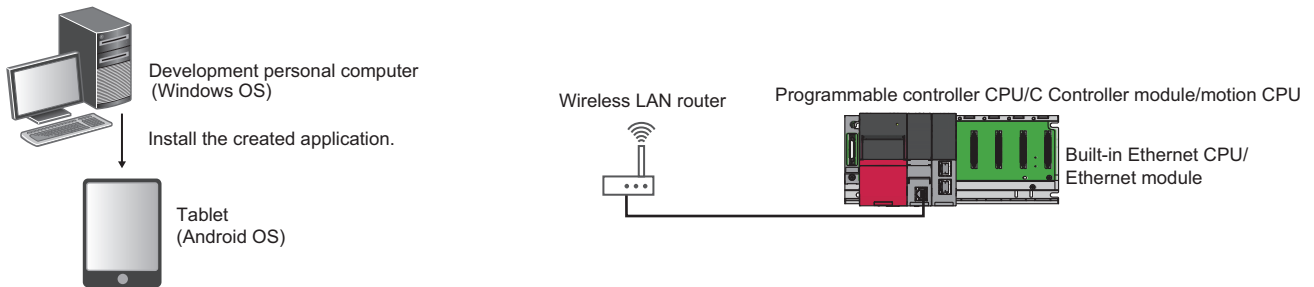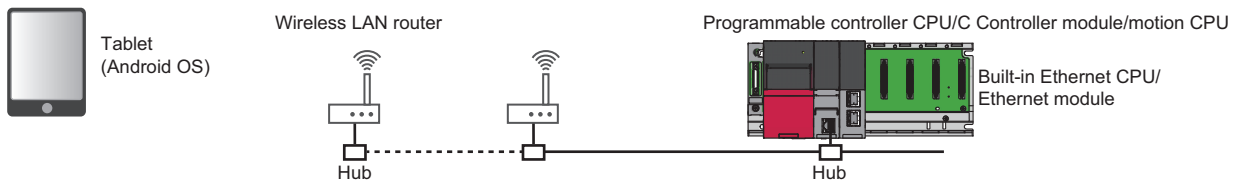
Considerations

■ Access route

A tablet accesses each module in a station which is directly connected with a wireless LAN router. It cannot access another station via an Ethernet module.

2.2 Configuration Devices

This section shows the usable and accessible CPU modules.

Usable CPU modules

Series	Module	Module name	Remarks
MELSEC iQ-R series	Programmable controller CPU	R04CPU, R04ENCPU, R08CPU, R08ENCPU, R08PCPU, R16CPU, R16ENCPU, R16PCPU, R32CPU, R32ENCPU, R32PCPU, R120CPU, R120ENCPU, R120PCPU	In RnPCPUs, the operation mode needs to be set to the process mode.
	C Controller module	R12CCPU-V	Communication via an Ethernet port of a C Controller module is not available. To communicate with a C Controller module, specify it as the other CPU, and route via a built-in Ethernet port of a programmable controller.
	Motion CPU	R16MTCPU, R32MTCPU	The PERIPHERAL I/F connector of a motion CPU cannot be used for communication. To communicate with a motion CPU, specify it as the other CPU, and route via a built-in Ethernet port of a programmable controller.
MELSEC iQ-F series[*1]	Programmable controller CPU	FX5UCPU, FX5UCCPU	—
MELSEC-Q series	Programmable controller CPU	Q00JCPU, Q00UJCPU, Q00CPU, Q00UCPU, Q01CPU, Q01UCPU, Q02CPU, Q02HCPU, Q02PHCPU, Q02UCPU, Q03UDCPU, Q04UDHCPU, Q06HCPU, Q06PHCPU, Q06UDHCPU, Q10UDHCPU, Q12HCPU, Q12PHCPU, Q12PRHCPU, Q13UDHCPU, Q20UDHCPU, Q25HCPU, Q25PHCPU, Q25PRHCPU, Q26UDHCPU	An Ethernet module is required.
		Q03UDECPU, Q03UDVCPU, Q04UDEHCPU, Q04UDVCPU, Q06UDEHCPU, Q06UDVCPU, Q10UDEHCPU, Q13UDEHCPU, Q13UDVCPU, Q20UDEHCPU, Q26UDEHCPU, Q26UDVCPU, Q50UDEHCPU, Q100UDEHCPU	Communication with another CPU cannot be performed via a built-in Ethernet port of a programmable controller CPU. To communicate with another CPU, use its built-in Ethernet port or route via an Ethernet module managed by it.
	C Controller module	Q12DCCPU-V, Q24DHCCPU-V, Q24DHCCPU-LS	Communication with another CPU cannot be performed via a built-in Ethernet port of a C Controller module. To communicate with another CPU, use its built-in Ethernet port or route via an Ethernet module managed by it.
	Motion CPU	Q172DCPU, Q173DCPU, Q172DSCPU, Q173DSCPU	The PERIPHERAL I/F connector of a motion CPU cannot be used for communication. To communicate with a motion CPU, route via an Ethernet module managed by it.
MELSEC-L series	Programmable controller CPU	L02SCPU, L02SCPU-P	An Ethernet module is required.
		L02CPU, L02CPU-P, L06CPU, L06CPU-P, L26CPU, L26CPU-P, L26CPU-BT, L26CPU-PBT	—

*1   Use the firmware version '05' or later.

Accessible Ethernet modules

An Ethernet module is required in order to communicate to a CPU module with no Ethernet port.

Series	Module name
MELSEC iQ-R series[*1,*2]	RJ71EN71
MELSEC-Q series[*3]	QJ71E71-100, QJ71E71-B5, QJ71E71-B2
MELSEC-L series[*4]	LJ71E71-100

*1   Use the firmware version '02' or later.
*2   It is not applicable when "Q Compatible Ethernet" is set for the network type of a module parameter in GX Works3.
*3   Use the first five digits of the serial number are '15042' and function version is 'D' or later.
*4   Use the first five digits of the serial number are '15042' and function version is 'A' or later.

2.3 Operating Environment

The following table shows the operating environment of MX Component for Android.

Development environment

Item	Description	
Personal computer[1]	A personal computer on which Microsoft$^®$Windows$^®$ operates	
Operating system (Japanese/English)	• Windows 10 Pro • Windows 8.1 • Windows 8.1 Pro • Windows 7 Professional	
Development environment[2] and language	Development application	• Android Studio[3] 1.3 to 2.2 • Eclipse 3.0 to 4.5
	Development language	Java$^®$[4]

*1   A CPU, required memory, and HDD free space need to follow the recommended specifications of the operating system, development environment and language.
*2   Operation by using the simulator included in the development environment is not guaranteed.
*3   Use Gradle version 2.5 to 2.10.
*4   Use Java Platform, Standard Edition 7 Development Kit ('JDK 7u60' or later).

Operating environment

Item	Description
Smartphone and tablet	A smartphone and a tablet on which Android$^®$ OS operates (wireless LAN-enabled type)[1]
Operating system	Android OS 4.2.X, 4.3.X, 4.4.X[2], 5.0.X, 5.1.X, 6.0.X, 7.0.X

*1   CPUs support only Arm$^®$. ('x86' is not supported.)
*2   When using Android OS 4.4.X, only 'Dalvik VM' is supported. ('ART VM' is not supported.)

2.4　Considerations

This section shows the considerations for using MX Component for Android.

Considerations for programming and various settings

■ Electric interruption and noise affection

On the communication by wireless LAN, some packets may be lost due to the electric interruption or noise affection. When creating a user program, include the resend processing and the line-reopen processing according to the system.

■ Erroneous data reception at timeout

Erroneous data may be received if timeout has occurred while connecting to an Ethernet port of an FX5CPU or a QCPU. Create a program to include the line-reopen processing.

■ Functions and applications that automatically disconnect the Wi-Fi communication

When any of the following functions or applications is enabled on a device such as a tablet, some packets may be lost and a timeout error may occur due to the disconnection of the Wi-Fi communication.
If a timeout error has occurred, perform the close processing, then perform the reopen processing again. In another way, use after disabling the functions and applications.

- Battery saving function of Wi-Fi setting[1]
- Auto connect/disconnect function of Wi-Fi network[1]
- Auto disconnect application of Wi-Fi communication

[1]　Depending on the device being used, the setting method is different; "Wi-Fi Advanced" → "Wi-Fi optimization" or "Wi-Fi Advanced" → "Wi-Fi timer".

■ Wireless LAN switching

On the communication by wireless LAN, some packets may be lost and a timeout error may occur due to the wireless LAN switching[1]. In the environment where wireless LAN is switched, create a program to include the resend processing and the line-reopen processing according to the system.

[1]　Wireless LAN roaming, hand-over, and disconnection etc.

■ Considerations for Android specifications

- When a device such as a table is in sleep status or the application is running as a background task, Ethernet communication of the application using this product may be disconnected. In this case, perform the reopen processing as necessary.
- This product is the communication library to perform the Ethernet communication. In an operating system, there are some essential items and recommendations related to the tasks for communication. Create a program by following them. Example: Thread the communication processing and perform it in the background.

Considerations for using this product with other products

For the connection setting between a device such as a tablet and a wireless LAN router, set a protocol such as WPA2 to encrypt data.
Data may illegally be accessed if a protocol that does not encrypt data is selected for the connection settings.
Select a much safer security setting between a device such as a tablet and wireless LAN router.

3 USAGE

This chapter explains how to use MX Component for Android.

3.1 Installation

This section shows the procedure for using this library on the development environment of Android application (Android Studio or Eclipse).

Android Studio

Operating procedure

1. Copy the 'libs' folder from the provided CD to the same folder[*1] where 'build.gradle' file of the project using this library is stored.

 *1 'build.gradle' file is stored in the folder that is '(path saving a project)\(module name)'. 'app' is created as the default module.
Example: C:\MyApplication\app
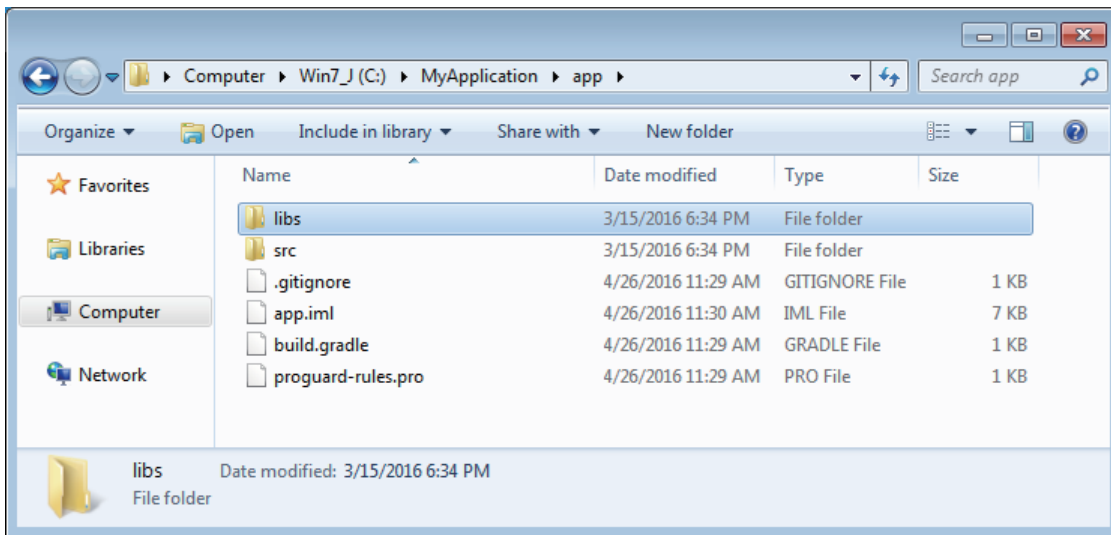
2. Copy the 'jni' folder from the provided CD to the folder, that is '(module name)\src\main', of the project using this library.
Example: C:\MyApplication\app\src\main
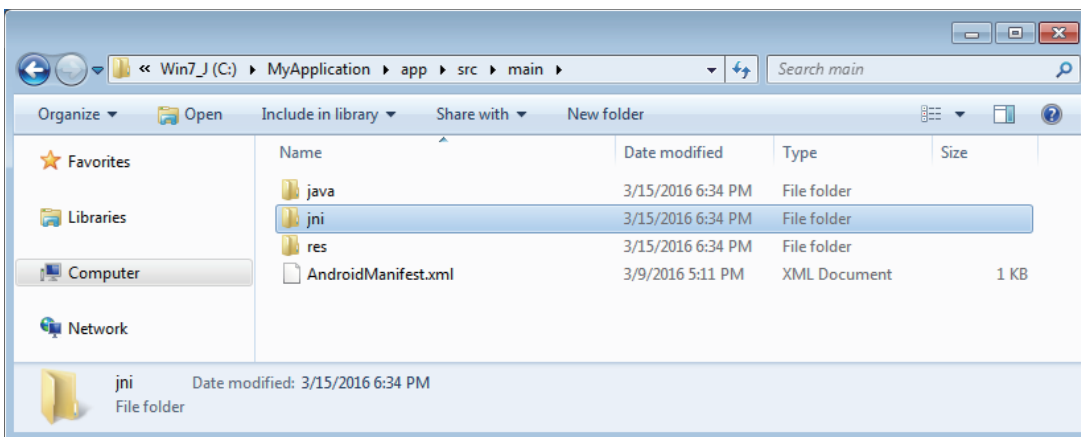
3. Delete the numbers[1] indicating the version information after the file extension 'so' of the following files in 'jni' folder.

Before deletion	After deletion
libinterface.so.1.01	libinterface.so
libMXComponent.so.1.01	libMXComponent.so
libSupport.so.1.02	libSupport.so

*1 For the numbers indicating the version information, refer to the following section.
　　☞ Page 23 Checking the version of MX Component for Android

4. Open the newly created project, then click the [Terminal] tab.
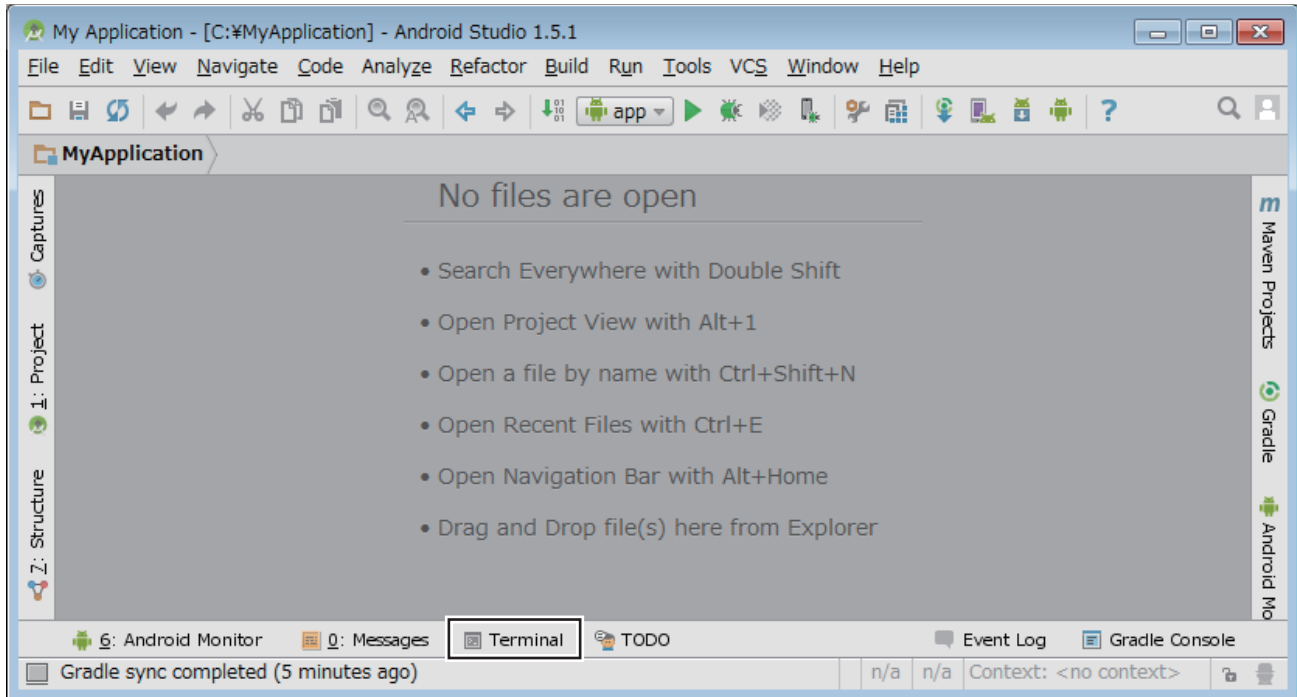
5. Change the current directory to the 'jni' folder with the 'cd' command.
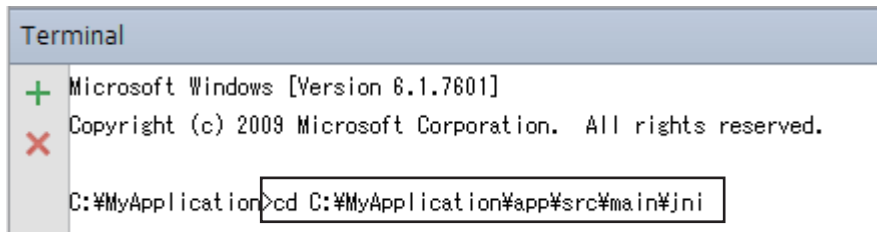
6. Execute the 'ndk-build' command.
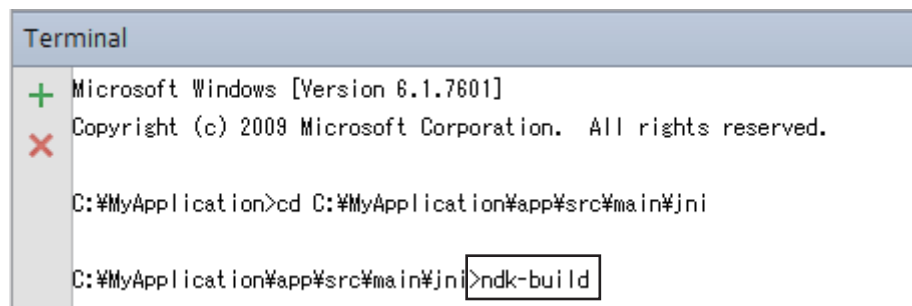
When the message ('ndk-build' is not recognized as an internal or external command, operable program or batch file.) appears, add the path of the folder to which Android NDK has been installed for "PATH" or "Path" in "Environment Variables" in Windows.
Make sure to restart Android Studio after setting it in "Environment Variables".

7. Open the 'build.gradle' file at the module level.

Example: C:\MyApplication\app\build.gradle
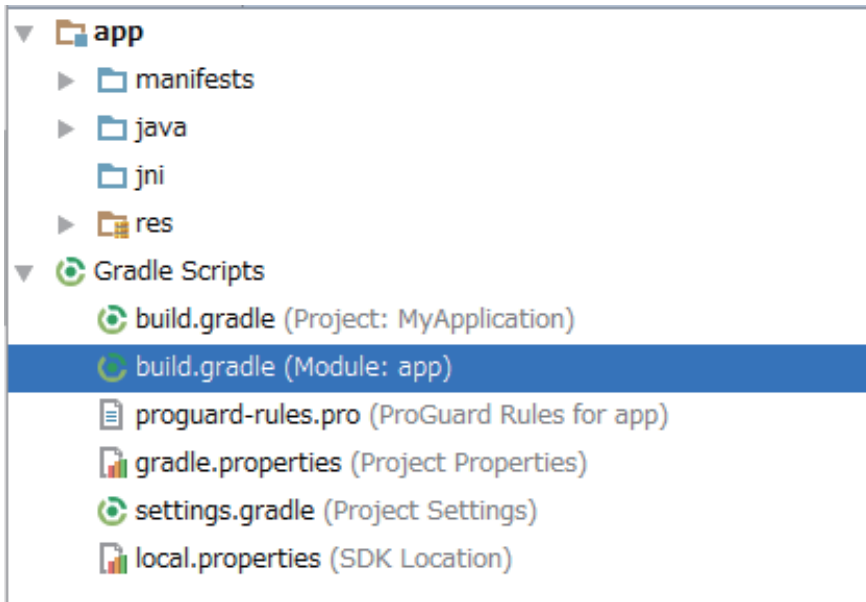
8. Create a program so that Gradle does not automatically call 'ndk-build'.

```
sourceSets{
    main{
        jni.srcDirs =[]
        jniLibs.srcDirs = ["src/main/jniLibs","src/main/libs"]
    }
}
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
}
```

Eclipse

Operating procedure

1. Copy the following folders from the provided CD to immediately under the project using this library.

- 'jni' folder
- 'libs' folder

Example: C:\workspace\sample
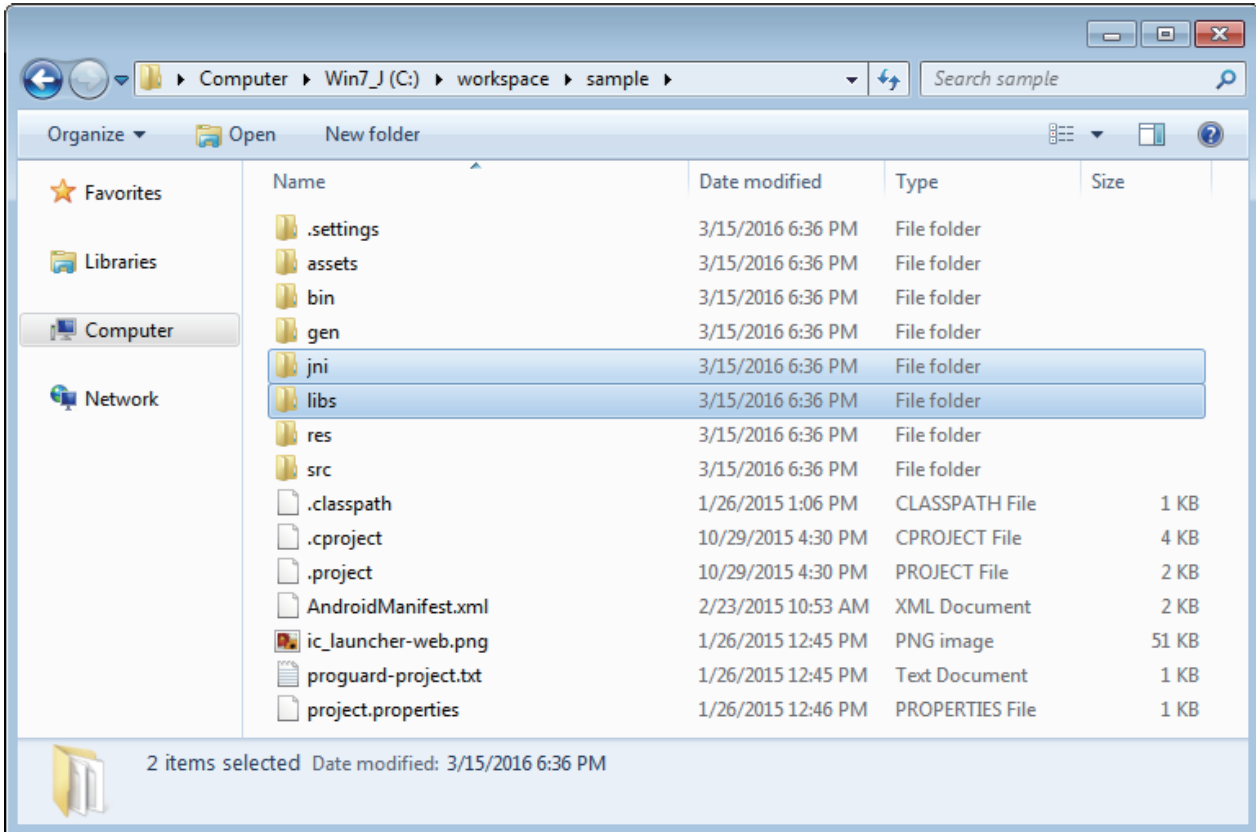
2. Delete the numbers[*1] indicating the version information after the file extension 'so' of the following files in 'jni' folder.

Before deletion	After deletion
libinterface.so.1.01	libinterface.so
libMXComponent.so.1.01	libMXComponent.so
libSupport.so.1.02	libSupport.so

*1 For the numbers indicating the version information, refer to the following section.
☞ Page 23 Checking the version of MX Component for Android

3.2 Creating a project

When creating a project, the following operations are required; importing the library, setting the permission, and reading the library.

Importing the library

This library can be imported as the following package name.

jp.co.melco.mxcomponent

Ex.
Eclipse

Setting the permission for the socket communication

Add the following statement in the "AndroidManifest.xml" to use the socket communication.

<uses-permission android:name="android.permission.INTERNET"></uses-permission>

Ex.

Eclipse
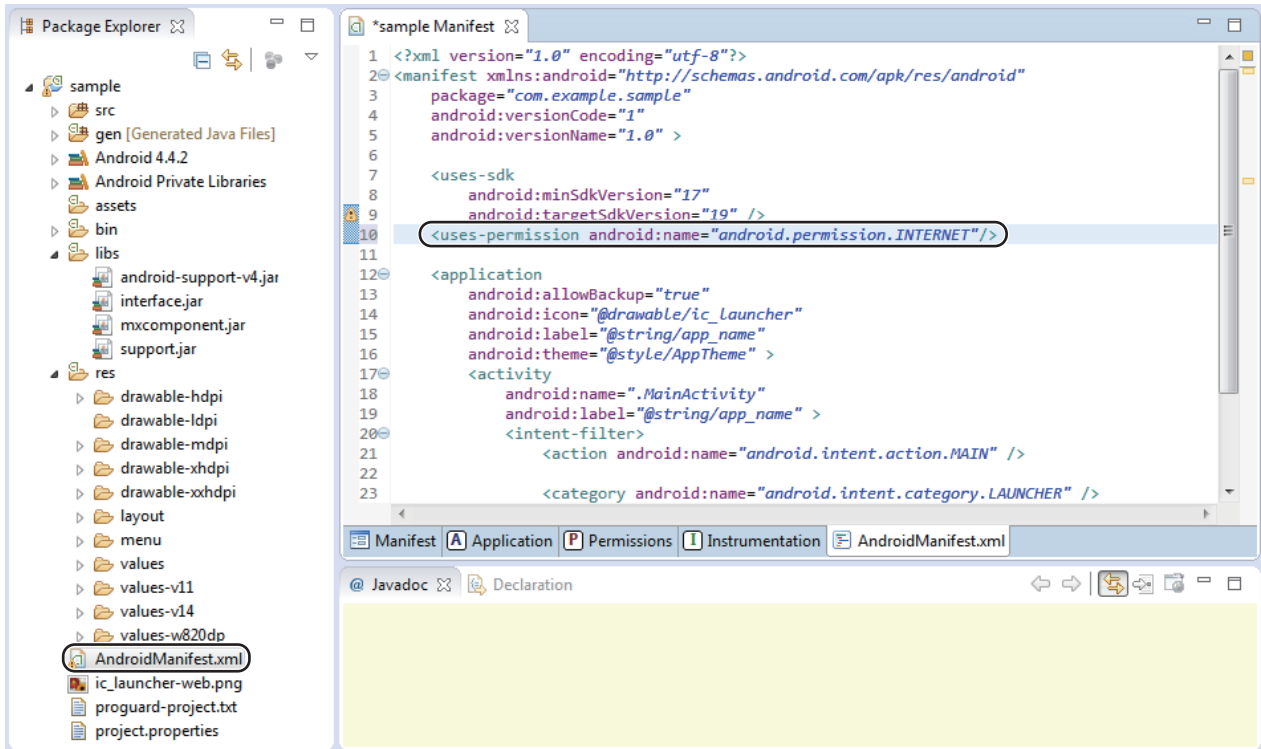
Reading the library

This library is a shared library.

Create a program to include the processing for reading the library by using Android Studio or Eclipse.

3.3 Update

For the information on how to obtain the update version of MX Component for Android, please consult your local Mitsubishi representative.

- The product version differs from the library version. For the product version, an alphabet is added at the end of the version number.

Update

The library can be updated by following the procedure below.

■ Android Studio

Operating procedure

1. Delete the numbers[1] indicating the version information after the file extension 'so' of the following files in 'jni' folder in the downloaded folder.

Before deletion	After deletion
libMXComponent.so.X.XX[1]	libMXComponent.so
libSupport.so.X.XX[1]	libSupport.so

*1 'X.XX' is the numbers indicating the version information. For the numbers, refer to the following section.
   ☞ Page 23 Checking the version of MX Component for Android

2. Overwrite the 'libs' folder in the downloaded folder to the 'libs' folder to which this library has been copied when being installed.

3. Overwrite the 'jni' folder in the downloaded folder to the 'jni' folder to which this library has been copied when being installed.

4. Open the project of Android Studio, then click the [Terminal] tab.
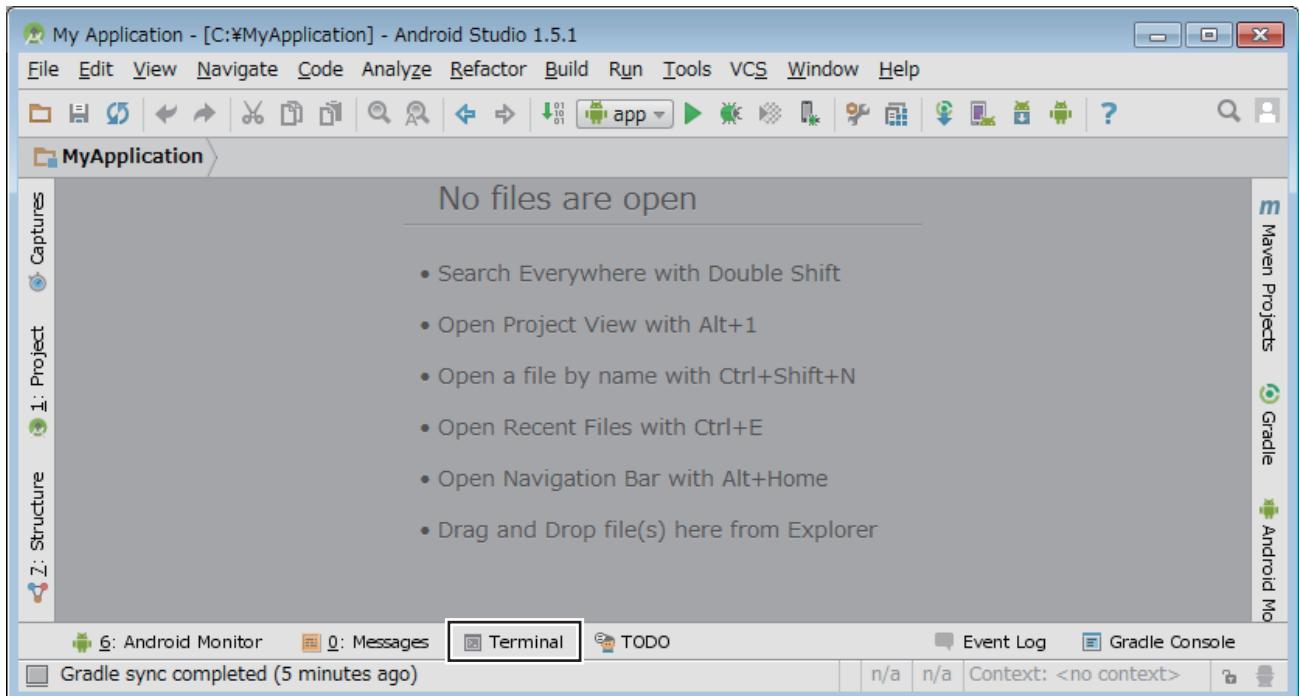
5. Change the current directory to the 'jni' folder with the 'cd' command.

```
Terminal
+  Microsoft Windows [Version 6.1.7601]
   Copyright (c) 2009 Microsoft Corporation.  All rights reserved.
×
   C:\MyApplication>cd C:\MyApplication\app\src\main\jni
```

6. Execute the 'ndk-build' command.

```
Terminal
+  Microsoft Windows [Version 6.1.7601]
   Copyright (c) 2009 Microsoft Corporation.  All rights reserved.
×

   C:\MyApplication>cd C:\MyApplication\app\src\main\jni

   C:\MyApplication\app\src\main\jni>ndk-build
```

■ Eclipse

Operating procedure

1. Delete the numbers[1] indicating the version information after the file extension 'so' of the following files in 'jni' folder in the downloaded folder.

Before deletion	After deletion
libMXComponent.so.X.XX[1]	libMXComponent.so
libSupport.so.X.XX[1]	libSupport.so

*1 'X.XX' is the numbers indicating the version information. For the numbers, refer to the following section.
☞ Page 23 Checking the version of MX Component for Android

2. Check that the 'libs' folder and 'jni' folder are included immediately under the project, then overwrite the folders. If those folders are not included, alter the folder configuration before overwriting.[1]

*1 When the folder configuration is altered, take the following actions.
Change the link that is set for the library to the altered folder in the setting of Eclipse.
Change the include path in the source code to the altered folder.

Precautions

Update may fail when it is performed with a target project of update opened. Update with the target project closed.

Checking the version of MX Component for Android

The version of this library can be checked by looking at the numbers following the file[1] extension 'so'.

   *1   For the version 1.02C or later, the file is stored in the 'jni' folder.

Ex.

The version of 'libSupport.so.1.01' is '1.01'.

Importing Eclipse projects to Android Studio

Import an Eclipse project into Android Studio.

Since this library uses Android NDK, perform the installation procedure from step 4 to 8 for Android Studio after the import.

(☞ Page 15 Android Studio)

3

3.4   Uninstallation

Delete the folders copied when this library was installed.

3.5 Communication method (open method)

Parameters to communicate with a tablet by Ethernet need to be to set to a module with an engineering tool.

To set the parameters, refer to the following sections.

Connection target module	Reference
RCPU	Page 25 For an RCPU
MELSEC iQ-R series-compatible EN71	Page 26 For a MELSEC iQ-R series-compatible EN71
FX5CPU	Page 26 For an FX5CPU
QCPU (Built-in Ethernet port)	Page 27 For a QCPU (built-in Ethernet port)
MELSEC-Q series-compatible E71	Page 27 For a MELSEC-Q series-compatible E71
MELSEC-Q series C Controller module	Page 27 For a MELSEC-Q series C Controller module
LCPU (Built-in Ethernet port)	Page 28 For an LCPU (built-in Ethernet port)
MELSEC-L series-compatible E71	Page 28 For an MELSEC-L series-compatible E71

Precautions

- To communicate to a CPU module with no Ethernet port, set the parameters for Ethernet communication to an Ethernet module.
- Same parameters for Ethernet communication are used for a motion CPU and a multiple CPU system. For the available combinations of CPU modules in the multiple CPU system configuration, refer to the manuals of each module.
- The setting for an RnENCPU differ depending on the Ethernet port being used.
  A port of the CPU part: the setting when the connection target is an RCPU
  A port of the network part: the setting when the connection target is a MELSEC iQ-R series-compatible EN71
- To communicate to a CPU module from multiple applications by using the multi-window of Android OS 7.0.X, set connections for each application.

For an RCPU

Set the following items in GX Works3 as follows:

■ MELSOFT Connection

Item	Setting	Setting screen
Enable/Disable Online Change	Enable All (SLMP)	"Parameter" ⇨ "(CPU model name of the project)" ⇨ "Module Parameter" ⇨ "Basic Settings" ⇨ "Own Node Settings" ⇨ "Enable/Disable Online Change" on the Navigation window
Communication Data Code	Binary	"Parameter" ⇨ "(CPU model name of the project)" ⇨ "Module Parameter" ⇨ "Basic Settings" ⇨ "Own Node Settings" ⇨ "Communication Data Code" on the Navigation window
Communication Method	MELSOFT Connection	"Parameter" ⇨ "(CPU model name of the project)" ⇨ "Module Parameter" ⇨ "Basic Settings" ⇨ "External Device Configuration" ⇨ "Communication Method" on the Navigation window

■ SLMP

Item	Setting	Setting screen
Enable/Disable Online Change	Enable All (SLMP)	"Parameter" ⇨ "(CPU model name of the project)" ⇨ "Module Parameter" ⇨ "Basic Settings" ⇨ "Own Node Settings" ⇨ "Enable/Disable Online Change" on the Navigation window
Communication Data Code	Binary	"Parameter" ⇨ "(CPU model name of the project)" ⇨ "Module Parameter" ⇨ "Basic Settings" ⇨ "Own Node Settings" ⇨ "Communication Data Code" on the Navigation window
Communication Method	SLMP	"Parameter" ⇨ "(CPU model name of the project)" ⇨ "Module Parameter" ⇨ "Basic Settings" ⇨ "External Device Configuration" ⇨ "Communication Method" on the Navigation window
Protocol	TCP	"Parameter" ⇨ "(CPU model name of the project)" ⇨ "Module Parameter" ⇨ "Basic Settings" ⇨ "External Device Configuration" ⇨ "Protocol" on the Navigation window
Host Station Port No.	Set the port number[1] of the CPU module.	"Parameter" ⇨ "(CPU model name of the project)" ⇨ "Module Parameter" ⇨ "Basic Settings" ⇨ "External Device Configuration" ⇨ "Port No." on the Navigation window

*1   For details on the port numbers, refer to the following manual.
      📖MELSEC iQ-R Ethernet User's Manual (Application)

For a MELSEC iQ-R series-compatible EN71

Set the following items in GX Works3 as follows:

■ MELSOFT Connection

Item	Setting	Setting screen
Enable/Disable Online Change	Enable All (SLMP)	"Parameter" ⇨ "RJ71EN71 (network type of the project[*1])" ⇨ "Module Parameter" ⇨ "Basic Settings" ⇨ "Own Node Settings" ⇨ "Enable/Disable Online Change" on the Navigation window
Communication Data Code	Binary	"Parameter" ⇨ "RJ71EN71 (network type of the project[*1])" ⇨ "Module Parameter" ⇨ "Basic Settings" ⇨ "Own Node Settings" ⇨ "Communication Data Code" on the Navigation window
Communication Method	MELSOFT Connection	"Parameter" ⇨ "RJ71EN71 (network type of the project[*1])" ⇨ "Module Parameter" ⇨ "Basic Settings" ⇨ "External Device Configuration" ⇨ "Communication Method" on the Navigation window

*1 It is not applicable when "Q Compatible Ethernet" is set for the network type of a module parameter in GX Works3.

■ SLMP

Item	Setting	Setting screen
Enable/Disable Online Change	Enable All (SLMP)	"Parameter" ⇨ "RJ71EN71 (network type of the project[*1])" ⇨ "Module Parameter" ⇨ "Basic Settings" ⇨ "Own Node Settings" ⇨ "Enable/Disable Online Change" on the Navigation window
Communication Data Code	Binary	"Parameter" ⇨ "RJ71EN71 (network type of the project[*1])" ⇨ "Module Parameter" ⇨ "Basic Settings" ⇨ "Own Node Settings" ⇨ "Communication Data Code" on the Navigation window
Communication Method	SLMP	"Parameter" ⇨ "RJ71EN71 (network type of the project[*1])" ⇨ "Module Parameter" ⇨ "Basic Settings" ⇨ "External Device Configuration" ⇨ "Communication Method" on the Navigation window
Protocol	TCP	"Parameter" ⇨ "RJ71EN71 (network type of the project[*1])" ⇨ "Module Parameter" ⇨ "Basic Settings" ⇨ "External Device Configuration" ⇨ "Protocol" on the Navigation window
Host Station Port No.	Set the port number[*2] of an Ethernet module.	"Parameter" ⇨ "RJ71EN71 (network type of the project[*1])" ⇨ "Module Parameter" ⇨ "Basic Settings" ⇨ "External Device Configuration" ⇨ "Port No." on the Navigation window

*1 It is not applicable when "Q Compatible Ethernet" is set for the network type of a module parameter in GX Works3.
*2 For details on the port numbers, refer to the following manual.
📖MELSEC iQ-R Ethernet User's Manual (Application)

For an FX5CPU

Set the following items in GX Works3 as follows:

Item	Setting	Setting screen
Communication Data Code	Binary	"Parameter" ⇨ "(CPU model name of the project)" ⇨ "Module Parameter" ⇨ "Ethernet Port" ⇨ "Basic Settings" ⇨ "Own Node Settings" ⇨ "Communication Data Code" on the Navigation window
Communication Method	SLMP	"Parameter" ⇨ "(CPU model name of the project)" ⇨ "Module Parameter" ⇨ "Ethernet Port" ⇨ "Basic Settings" ⇨ "External Device Configuration" ⇨ "Communication Method" on the Navigation window
Protocol	TCP	"Parameter" ⇨ "(CPU model name of the project)" ⇨ "Module Parameter" ⇨ "Ethernet Port" ⇨ "Basic Settings" ⇨ "External Device Configuration" ⇨ "Protocol" on the Navigation window
Host Station Port No.	Set the port number[*1] of the CPU module.	"Parameter" ⇨ "(CPU model name of the project)" ⇨ "Module Parameter" ⇨ "Ethernet Port" ⇨ "Basic Settings" ⇨ "External Device Configuration" ⇨ "Port No." on the Navigation window

*1 For details on the port numbers, refer to the following manual.
📖MELSEC iQ-F FX5 User's Manual (Ethernet Communication)

For a QCPU (built-in Ethernet port)

Set the following items in GX Works2 as follows:

Item	Setting	Setting screen
Protocol	TCP	"Parameter" ⇨ "PLC Parameter" ⇨ [Built-in Ethernet Port Setting] tab ⇨ "Open Setting" ⇨ "Protocol" on the Navigation window
Open System	MC Protocol	"Parameter" ⇨ "PLC Parameter" ⇨ [Built-in Ethernet Port Setting] tab ⇨ "Open Setting" ⇨ "Open System" on the Navigation window
Host Station Port No.	Set the port number[*1] of the CPU module.	"Parameter" ⇨ "PLC Parameter" ⇨ [Built-in Ethernet Port Setting] tab ⇨ "Open Setting" ⇨ "Host Station Port No." on the Navigation window
Communication Data Code	Binary Code	"Parameter" ⇨ "PLC Parameter" ⇨ [Built-in Ethernet Port Setting] tab ⇨ "Communication Data Code" on the Navigation window
Enable online change (FTP, MC Protocol)	Select the checkbox of "Enable online change (FTP, MC Protocol)".	"Parameter" ⇨ "PLC Parameter" ⇨ [Built-in Ethernet Port Setting] tab ⇨ "Enable online change (FTP, MC Protocol)" on the Navigation window

*1 For details on the port numbers, refer to the following manual.
📖 QnUCPU User's Manual (Communication via Built-in Ethernet Port)

For a MELSEC-Q series-compatible E71

Set the following items in GX Works2 as follows:

Item	Setting	Setting screen
Protocol	TCP	"Parameter" ⇨ "Network Parameter" ⇨ "Ethernet/CC IE/MELSECNET" ⇨ Select "Ethernet" for network type ⇨ "Open Setting" ⇨ "Protocol" on the Navigation window
Open System	MELSOFT Connection	"Parameter" ⇨ "Network Parameter" ⇨ "Ethernet/CC IE/MELSECNET" ⇨ Select "Ethernet" for network type ⇨ "Open Setting" ⇨ "Open System" on the Navigation window
Communication Data Code	Binary Code	"Parameter" ⇨ "Network Parameter" ⇨ "Ethernet/CC IE/MELSECNET" ⇨ Select "Ethernet" for network type ⇨ "Operation Setting" ⇨ "Communication Data Code" on the Navigation window
Enable Online Change	Select the checkbox of "Enable Online Change".	"Parameter" ⇨ "Network Parameter" ⇨ "Ethernet/CC IE/MELSECNET" ⇨ Select "Ethernet" for network type ⇨ "Operation Setting" ⇨ "Enable Online Change" on the Navigation window

For a MELSEC-Q series C Controller module

Set the following items in Setting/monitoring tools for the C Controller module as follows:

Item	Setting	Setting screen
Protocol	TCP	• Q12DCCPU-V "Parameter" ⇨ "CCPU Parameter" ⇨ [Built-in Ethernet port(CH1 and CH2) open settings] tab ⇨ "Open Setting" ⇨ "Protocol" on the Navigation window
		• Q24DHCCPU-V and Q24DHCCPU-LS "Parameter" ⇨ "CCPU Parameter" ⇨ [System Ethernet port (S CH1) settings] tab ⇨ "Open Setting" ⇨ "Protocol" on the Navigation window
Open System	MC Protocol	• Q12DCCPU-V "Parameter" ⇨ "CCPU Parameter" ⇨ [Built-in Ethernet port(CH1 and CH2) open settings] tab ⇨ "Open Setting" ⇨ "Open System" on the Navigation window
		• Q24DHCCPU-V and Q24DHCCPU-LS "Parameter" ⇨ "CCPU Parameter" ⇨ [System Ethernet port (S CH1) settings] tab ⇨ "Open Setting" ⇨ "Open System" on the Navigation window
Host Station Port No.	Set the port number[*1] of the C Controller module.	• Q12DCCPU-V "Parameter" ⇨ "CCPU Parameter" ⇨ [Built-in Ethernet port(CH1 and CH2) open settings] tab ⇨ "Open Setting" ⇨ "Host Station Port No." on the Navigation window
		• Q24DHCCPU-V and Q24DHCCPU-LS "Parameter" ⇨ "CCPU Parameter" ⇨ [System Ethernet port (S CH1) settings] tab ⇨ "Open Setting" ⇨ "Host Station Port No." on the Navigation window

*1 For details on the port numbers, refer to the following manual.
📖 Setting/Monitoring Tools for the C Controller Module Version 4 Operating Manual

For an LCPU (built-in Ethernet port)

Set the following items in GX Works2 as follows:

Item	Setting	Setting screen
Protocol	TCP	"Parameter" ⇨ "PLC Parameter" ⇨ [Built-in Ethernet Port Setting] tab ⇨ "Open Setting" ⇨ "Protocol" on the Navigation window
Open System	MC Protocol	"Parameter" ⇨ "PLC Parameter" ⇨ [Built-in Ethernet Port Setting] tab ⇨ "Open Setting" ⇨ "Open System" on the Navigation window
Host Station Port No.	Set the port number[1] of the CPU module.	"Parameter" ⇨ "PLC Parameter" ⇨ [Built-in Ethernet Port Setting] tab ⇨ "Open Setting" ⇨ "Host Station Port No." on the Navigation window
Communication Data Code	Binary Code	"Parameter" ⇨ "PLC Parameter" ⇨ [Built-in Ethernet Port Setting] tab ⇨ "Communication Data Code" on the Navigation window
Enable online change (FTP, MC Protocol)	Select the checkbox of "Enable online change (FTP, MC Protocol)".	"Parameter" ⇨ "PLC Parameter" ⇨ [Built-in Ethernet Port Setting] tab ⇨ "Enable online change (FTP, MC Protocol)" on the Navigation window

*1 For details on the port numbers, refer to the following manual.
📖 MELSEC-L CPU Module User's Manual (Built-In Ethernet Function)

For an MELSEC-L series-compatible E71

Set the following items in GX Works2 as follows:

Item	Setting	Setting screen
Protocol	TCP	"Parameter" ⇨ "Network Parameter" ⇨ "Ethernet/CC IE Field" ⇨ Select "Ethernet" for network type ⇨ "Open Setting" ⇨ "Protocol" on the Navigation window
Open System	MELSOFT Connection	"Parameter" ⇨ "Network Parameter" ⇨ "Ethernet/CC IE Field" ⇨ Select "Ethernet" for network type ⇨ "Open Setting" ⇨ "Open System" on the Navigation window
Communication Data Code	Binary Code	"Parameter" ⇨ "Network Parameter" ⇨ "Ethernet/CC IE Field" ⇨ Select "Ethernet" for network type ⇨ "Operation Setting" ⇨ "Communication Data Code" on the Navigation window
Enable Online Change	Select the checkbox of "Enable Online Change".	"Parameter" ⇨ "Network Parameter" ⇨ "Ethernet/CC IE Field" ⇨ Select "Ethernet" for network type ⇨ "Operation Setting" ⇨ "Enable Online Change" on the Navigation window

4 ACCESSIBLE DEVICES

This chapter explains the accessible devices with this library.

4.1 Accessible Device List

The following table shows the applicable devices for reading and writing devices of this library.

Programmable controller CPU

○: Accessible, ×: Not accessible

Category	Device name[1],[2]		Symbol	Type	Notation	Access target[3]		
						RCPU	FX5CPU	QCPU/ LCPU
System device	Special relay		SM	Bit	Decimal	○	○	○
	Special register		SD	Word	Decimal	○	○	○
User device	Input		X	Bit	Hexadecimal	○	×	○
					Octal	×	○	×
	Output		Y		Hexadecimal	○	×	○
					Octal	×	○	×
	Internal relay		M		Decimal	○	○	○
	Latch relay		L		Decimal	○	○	○
	Annunciator		F		Decimal	○	○	○
	Edge relay		V		Decimal	○	×	○
	Link relay		B		Hexadecimal	○	○	○
	Data register		D	Word	Decimal	○	○	○
	Link register		W		Hexadecimal	○	○	○
	Timer (T)	Contact	TS	Bit	Decimal	×	×	×
		Coil	TC		Decimal	×	×	×
		Current value	TN	Word	Decimal	○	○	○
	Long timer (LT)	Contact	LTS	Bit	Decimal	×	×	×
		Coil	LTC		Decimal	×	×	×
		Current value	LTN	Double Word	Decimal	○	×	×
	Retentive timer (ST)	Contact	STS	Bit	Decimal	×	×	×
			SS		Decimal	×	×	×
		Coil	STC		Decimal	×	×	×
		Current value	STN	Word	Decimal	○	×	○
			SN		Decimal	×	○	×
	Long retentive timer (LST)	Contact	LSTS	Bit	Decimal	×	×	×
		Coil	LSTC		Decimal	×	×	×
		Current value	LSTN	Double Word	Decimal	○	×	×
	Counter (C)	Contact	CS	Bit	Decimal	×	×	×
		Coil	CC		Decimal	×	×	×
		Current value	CN	Word	Decimal	○	○	○
	Long counter (LC)	Contact	LCS	Bit	Decimal	×	×	×
		Coil	LCC		Decimal	×	×	×
		Current value	LCN	Double Word	Decimal	○	×	×
	Link special relay		SB	Bit	Hexadecimal	○	○	○
	Link special register		SW	Word	Hexadecimal	○	○	○
	Step relay		S	Bit	Decimal	×	○	×
	Direct access input		DX		Hexadecimal	×	×	×
	Direct access output		DY		Hexadecimal	×	×	×

Category	Device name[1,2]		Symbol	Type	Notation	Access target[3]		
						RCPU	FX5CPU	QCPU/ LCPU
Index register			Z	Word	Decimal	○	○	○
			LZ	Double Word	Decimal	○	○	×
File register[4,5]			R	Word	Decimal	○	○	○
			ZR		Decimal	○	×	○
Extended data register			D	Word	Decimal	×	×	○[6]
Extended link register			W	Word	Hexadecimal	×	×	○[6]
Link direct device[7,8]	Link input		J□\X	Bit	Hexadecimal	○	×	○
	Link output		J□\Y		Hexadecimal	○	×	○
	Link relay		J□\B		Hexadecimal	○	×	×
	Link special relay		J□\SB		Hexadecimal	○	×	○
	Link register		J□\W	Word	Hexadecimal	○	×	○
	Link special register		J□\SW		Hexadecimal	○	×	○
Module access device[7]	Module access device (RCPU/ FX5CPU)/Intelligent function module device (QCPU/LCPU)		U□\G[9]	Word	Decimal	○	×	○
CPU buffer memory access device[7]	CPU buffer memory access device (RCPU)/Cyclic transmission area device (QCPU)		U3E□\G	Word	Decimal	○	×	×
	Fixed cycle communication area access device		U3E□\HG		Decimal	○	×	×
Refresh data register	Refresh data register		RD	Word	Decimal	○	×	×

*1   Local devices can not be accessed.
*2   Digit specification, bit specification, and index modification specification are not available.
*3   Even though the applicability of devices indicates ○ (accessible) in the list, some devices may not be accessed due to the restrictions of methods. For details, refer to the considerations of each method.
*4   When the file register consists of multiple blocks, use the device codes in the sequential number access method.
*5   In GX Works3, select "Use Common File Register in All Programs" in the CPU parameter. In GX Works2, select "Use the following file" in the PLC parameter.
*6   Only universal model QCPUs and LCPUs support the devices.
*7   Some methods are restricted since the devices are classified as the device extension specification.
*8   For the link direct devices (J□), specify a network number in hexadecimal.
*9   For the module access device (U□\G), specify the I/O number of the target module.
    Example: U20\G1000 (when the I/O number is '200H' and the address is '1000'.)

C Controller module

Numbers in the access target column indicate the CPU modules: (1) MELSEC iQ-R series C Controller module, (2) MELSEC-Q series C Controller module

○: Accessible, ×: Not accessible

Category	Device[*1,*2]	Symbol	Type	Notation	Access target[*3]	
					(1)	(2)
System device	Special relay[*4]	SM	Bit	Decimal	○	○
	Special register[*4]	SD	Word	Decimal	○	○
User device	Input	X	Bit	Hexadecimal	○	○
	Output	Y		Hexadecimal	○	○
	Internal relay[*4]	M		Decimal	○	○
	Link relay	B		Hexadecimal	○	○
	Data register[*4]	D	Word	Decimal	○	○
	Link register	W		Hexadecimal	○	○
File register		ZR	Word	Decimal	×	×
Link direct device	Link input	J□\X	Bit	Hexadecimal	×	×
	Link output	J□\Y		Hexadecimal	×	×
	Link relay	J□\B		Hexadecimal	×	×
	Link special relay	J□\SB		Hexadecimal	×	×
	Link register	J□\W	Word	Hexadecimal	×	×
	Link special register	J□\SW		Hexadecimal	×	×
Module access device	Module access device(MELSEC iQ-R series)/Intelligent function module device(MELSEC-Q series)	U□\G	Word	Decimal	×	×
CPU buffer memory access device	CPU buffer memory access device (MELSEC iQ-R series)/Cyclic transmission area device (MELSEC-Q series)	U3E□\G	Word	Decimal	×	×
	Fixed cycle communication area access device	U3E□\HG		Decimal	×	×

*1 Local devices can not be accessed.
*2 Digit specification, bit specification, and index modification specification are not available.
*3 Even though the applicability of devices indicates ○ (accessible) in the list, some devices may not be accessed due to the restrictions of methods. For details, refer to the considerations of each method.
*4 The Q12DCCPU-V of which the first five digits of the serial number are '12042' is accessible.

4

Motion CPU

Numbers in the access target column indicate the CPU modules: (1) MELSEC iQ-R series motion CPU, (2) MELSEC-Q series motion CPU

○: Accessible, ×: Not accessible

Category	Device name[*1],[*2]	Symbol	Type	Notation	Access target[*3]	
					(1)	(2)
System device	Special relay	SM	Bit	Decimal	○	○
	Special register	SD	Word	Decimal	○	○
User device	Input	X	Bit	Hexadecimal	○	○
	Output	Y		Hexadecimal	○	○
	Internal relay	M		Decimal	○	○
	Annunciator	F		Decimal	○	○
	Link relay	B		Hexadecimal	○	○
	Data register	D	Word	Decimal	○	○
	Link register	W		Hexadecimal	○	○
	Motion register	#		Decimal	×	×
Module access device[*4]	Module access device	U□\G[*5]	Word	Decimal	○	×
CPU buffer memory access device[*4]	CPU buffer memory access device	U3E□\G	Word	Decimal	○	×
	Fixed cycle communication area access device	U3E□\HG		Decimal	○	×

*1 Local devices can not be accessed.

*2 Digit specification, bit specification, and index modification specification are not available.

*3 Even though the applicability of devices indicates ○ (accessible) in the list, some devices may not be accessed due to the restrictions of methods. For details, refer to the considerations of each method.

*4 Some methods are restricted since the devices are classified as the device extension specification.

*5 For the module access device (U□\G), specify the I/O number of the target module.
   Example: U20\G1000 (when the I/O number is '200H' and the address is '1000'.)

4.2 Considerations for Devices and Labels

This section shows the considerations for using devices and labels.
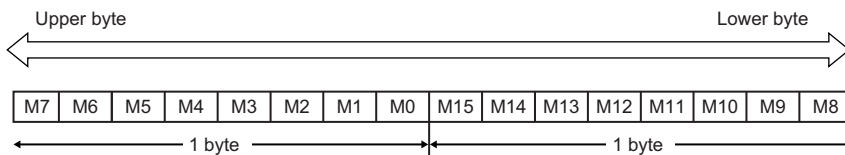
Considerations for bit devices

The place of memory needs to be considered when reading or writing bit devices in a batch. Bit devices are stored in the memory in a word unit. Note these points when acquiring or setting the bit values of target devices.
The byte-order of Android is the big endian.

Ex.
When reading one point (one word) from M0

Upper byte								Lower byte							
M7	M6	M5	M4	M3	M2	M1	M0	M15	M14	M13	M12	M11	M10	M9	M8
			1 byte								1 byte				

The bit devices are stored in order from upper bytes.

Considerations for using labels

Data types of labels

The following table shows the variable types of this library (Java) corresponding to the data types of labels.
The labels of which the data types are timer, retentive timer, counter, long timer, long retentive timer, or long counter are treated as the structure type. For details, refer to the following section.

☞ Page 35 Data types of labels treated as the structure type

Data type of a label	Variable type in this library (Java)	Definition name
Bit	Boolean	MEL_MX_LABEL_DATATYPE_BIT
Word [Signed]	Integer	MEL_MX_LABEL_DATATYPE_WORD
Word [Unsigned]/Bit String [16-bit]	Integer	MEL_MX_LABEL_DATATYPE_UNSIGNED_WORD
Double Word [Signed]	Long	MEL_MX_LABEL_DATATYPE_DOUBLE_WORD
Double Word [Unsigned]/Bit String [32-bit]	Long	MEL_MX_LABEL_DATATYPE_UNSIGNED_DOUBLE_WORD
FLOAT (Single Precision)	Float	MEL_MX_LABEL_DATATYPE_FLOAT_SINGLE
FLOAT (Double Precision)	Double	MEL_MX_LABEL_DATATYPE_FLOAT_DOUBLE
String (32)	String	MEL_MX_LABEL_DATATYPE_STRING_ASCII
String [Unicode] (32)	String	MEL_MX_LABEL_DATATYPE_STRING_UNICODE
Time	Long	MEL_MX_LABEL_DATATYPE_TIME

4

Specification methods of label names

When reading and writing labels, the label names registered in the CPU module need to be specified.

The applicable labels are as shown below.

■ Simple data type

Specify the data type described in the following section.

☞ Page 33 Data types of labels

■ Array type

Up to three dimensions of an array label can be specified.

Specify a label name and indexes of the array element (decimal).

For the data type of an array label, only the simple data type can be specified.

Ex.

When specifying up to three dimensions: Label1[10,10,10]

■ Structure type

Specify a structure label.

Specify all elements by connecting each element name of the structure with the period.

For the last element, the simple data type or array type can only be specified.

The whole structure cannot be specified by specifying the structure name only.

Ex.

Applicable: stLabel1.stLabel2.bMember1 (The simple data type label is specified for the last element.)

Inapplicable: stLabel1.stLabel2 (The structure label is specified for the last element.)

Precautions

- If the specific conditions of label name length are not satisfied, the label may not be specified. For details on the conditions, refer to each method (considerations).
- Digit specification, bit specification, and index modification specification are not available. Inapplicable label names are as follows;

Label type	Specification example
Bit specification of a label	Label.3
Digit specification of a label	K4Label
Specification of index modification	LabelZ0
Specification of multiple CPU No.2	U3E1\Label

- An alias name cannot be specified.

Data types of labels treated as the structure type

Labels of the following data types are treated as the structure type that includes a contact, coil, and current value for its elements. Note the specification method of label names when reading and writing the labels.

- Timer
- Retentive timer
- Counter
- Long timer
- Long retentive timer
- Long counter

Ex.

When reading the current value of the label name 'TimerLabel' in which the long retentive timer is assigned, specify the data type as follows;

TimerLabel.N

The following shows the members of the data types treated as the structure type.

Data name	Member name	Data type
Contact	S	Bit
Coil	C	Bit
Current value	N	Timer, retentive timer, counter: Word (Unsigned)
		Long timer, long retentive timer, long counter: Double Word (Unsigned)

5 METHODS

This chapter explains the procedure and functions of methods provided by the library.

Procedure for using methods

The following flow chart shows the basic order when reading and writing devices and labels by using the methods of MX Component for Android.
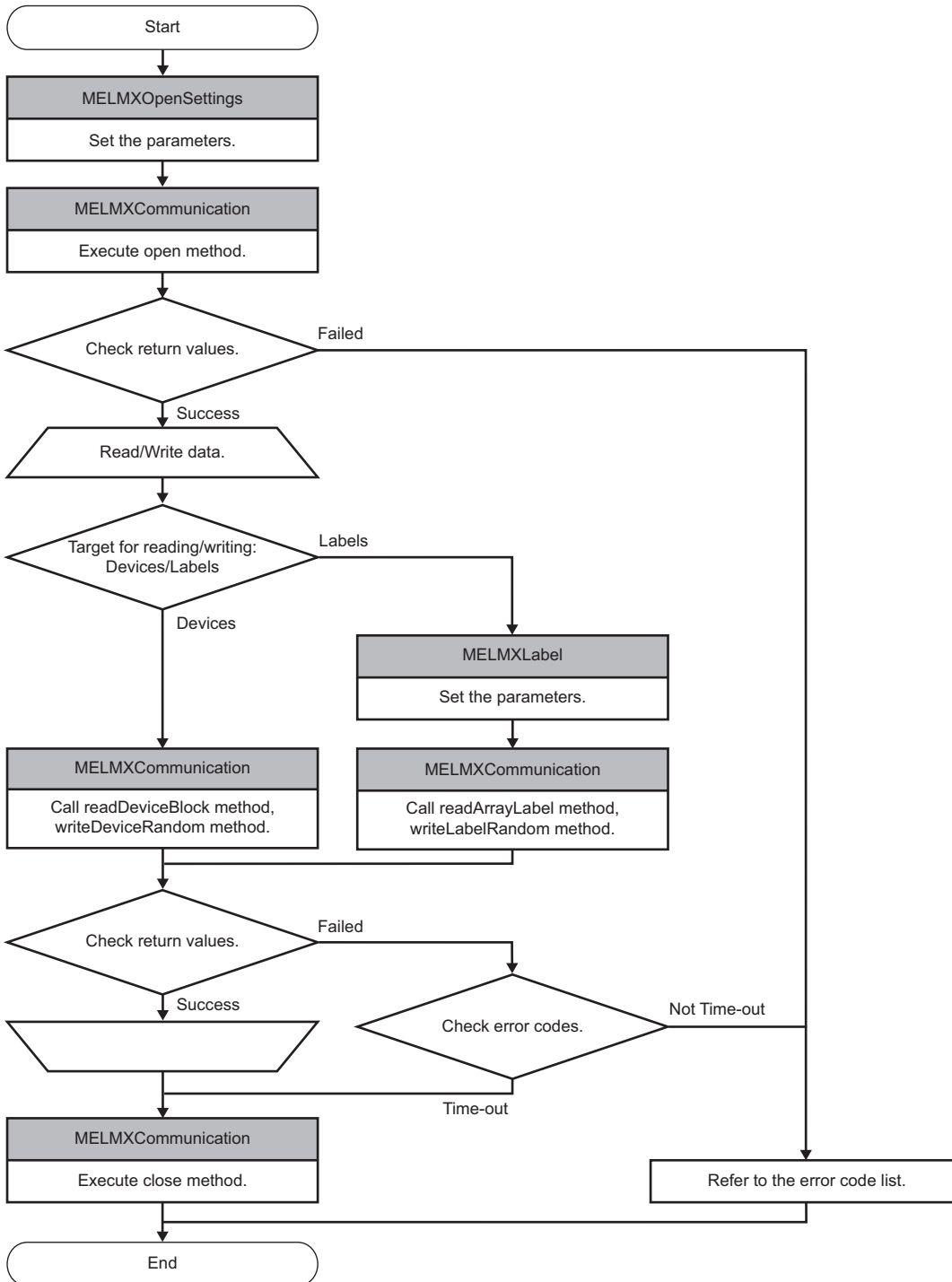
To use a method in a program, open the communication line at the beginning of the program, and close it at the end of the program.

5.1 Method List

The following table shows the method list.

Class	Method	Function	Reference
MELMxCommunication	MELMxCommunication	Constructor	Page 38 MELMxComminication (Constructor)
	open	To open the communication line	Page 38 open (To open the communication line)
	close	To close the communication line	Page 39 close (To close the communication line)
	readDeviceBlock	To read devices in a batch	Page 40 readDeviceBlock (To read devices in a batch)
	writeDeviceBlock	To write devices in a batch	Page 41 writeDeviceBlock (To write devices in a batch)
	readDeviceRandom	To read devices randomly	Page 42 readDeviceRandom (To read devices randomly)
	writeDeviceRandom	To write devices randomly	Page 43 writeDeviceRandom (To write devices randomly)
	readArrayLabel	To read data by specifying an array label	Page 44 readArrayLabel (To read data by specifying an array label)
	writeArrayLabel	To write data by specifying an array label	Page 44 writeArrayLabel (To write data by specifying an array label)
	readLabelRandom	To read data by specifying multiple labels	Page 45 readLabelRandom (To read data by specifying multiple labels)
	writeLabelRandom	To write data by specifying multiple labels	Page 45 writeLabelRandom (To write data by specifying multiple labels)
MELMxLabel	MELMxLabel	Constructor	Page 50 MELMxLabel (Constructor)
	getValues	To acquire label data	Page 50 getValues (To acquire label data)
	getName	To acquire a label name	Page 50 getName (To acquire a label name)
	getDataType	To acquire the data type of a label	Page 51 getDataType (To acquire the data type of a label)
	setBitLabel	To set the label data of Bit type	Page 51 setBitLabel (To set the label data of Bit type)
	setWordLabel	To set the label data of Word (Signed)	Page 51 setWordLabel (To set the label data of Word (Signed))
	setUnsignedWordLabel	To set the label data of Word (Unsigned)	Page 52 setUnsignedWordLabel (To set the label data of Word (Unsigned))
	setDoubleWordLabel	To set the label data of Double Word (Signed)	Page 52 setDoubleWordLabel (To set the label data of Double Word (Signed))
	setUnsignedDoubleWordLabel	To set the label data of Double Word (Unsigned)	Page 52 setUnsignedDoubleWordLabel (To set the label data of Double Word (Unsigned))
	setFloatSingleLabel	To set the label data of FLOAT (Single Precision)	Page 53 setFloatSingleLabel (To set the label data of FLOAT (Single Precision))
	setFloatDoubleLabel	To set the label data of FLOAT (Double Precision)	Page 53 setFloatDoubleLabel (To set the label data of FLOAT (Double Precision))
	setAsciiStringLabel	To set the label data of ASCII character string	Page 53 setAsciiStringLabel (To set the label data of ASCII character string)
	setUnicodeStringLabel	To set the label data of Unicode character string	Page 54 setUnicodeStringLabel (To set the label data of Unicode character string)
	setTimeLabel	To set the label data of Time	Page 54 setTimeLabel (To set the label data of Time)

5.2 Details of Methods

This section shows the details of each method.

MELMxCommunication class

It is the class to access a CPU module for reading and writing data.

Create multiple instances as necessary when performing communication with multiple CPU modules.

Details of methods

■ MELMxComminication (Constructor)

MELSEC iQ-R series	MELSEC iQ-F series	MELSEC-Q series	MELSEC-L series
○	○	○	○

| Item	Description
Class name	MELMxCommunication
Method definition	void MELMxCommuinication()
Argument	None
Return value	None
Function	It is the constructor of this class. Up to 10 instances can be created for this class. The operation is not guaranteed if 11 or more instances are created.
Reference	None
Considerations	None

■ open (To open the communication line)

MELSEC iQ-R series	MELSEC iQ-F series	MELSEC-Q series	MELSEC-L series
○	○	○	○

Item	Description	
Class name	MELMxCommunication	

Method definition	int open(
	    MELMxOpenSettings openSettings,
	    String remotePassword
	)	

Argument	[in]openSettings	Specify the instance of MELMxOpenSettings class that stores a setting value for connection. For details, refer to the following section. ☞ Page 46 MELMxOpenSettings class
	[in]remotePassword	Specify a password to unlock the remote password. Specify 'null' when any remote password has not been set in a module. For 'null', the unlock processing of a remote password is not performed.
Return value	MX_SUCCESS: Succeed Except for MX_SUCCESS: Failed (☞ Page 68 Error Code List)	
Function	Opens the communication line. The communication with one CPU module is enabled per one instance. When performing communication with multiple CPU modules, create multiple instances as necessary. When simultaneously accessing CPU modules by using multiple instances with TCP/IP, multiple Ethernet modules or ports of built-in Ethernet CPUs are required. One port is required for one instance. The communication between one port and two or more instances cannot be performed.	
Reference	close()	
Considerations	• Remote password A remote password is unlocked in the open method, and it is locked again when the close method is called. The operation related to a remote password is the same as that for an Ethernet module and a built-in Ethernet CPU. When 'null' is specified to 'remotePassword' (argument) even if a remote password has been set to an Ethernet module or a built-in Ethernet CPU, the error is detected with the method for reading and writing devices and labels (not with the open method).	

■ close (To close the communication line)

MELSEC iQ-R series	MELSEC iQ-F series	MELSEC-Q series	MELSEC-L series
○	○	○	○

Item	Description
Class name	MELMxCommunication
Method definition	int close()
Argument	None
Return value	MX_SUCCESS: Succeed Except for MX_SUCCESS: Failed (☞ Page 68 Error Code List)
Function	Disconnects the communication.
Reference	open()
Considerations	When the close method is executed, the disconnection request of TCP connection may not reach to a CPU module or an Ethernet module due to disconnection of the Ethernet cable on the module side or powering-OFF of the wireless router. In that case, the connection of the CPU module or Ethernet module side is retained. Wait until the connection is disconnected.

• Time until the connections for each connected module are disconnected

Connected module	Check item
RCPU	GX Works3: Module Parameter ⇨ [Application Settings] ⇨ [Timer Settings for Data Communication]
MELSEC iQ-R series-compatible EN71	
FX5CPU	
QCPU (Built-in Ethernet port)	Inapplicable (Timeout value of KeepAlive: 45 seconds)
LCPU (Built-in Ethernet port)	
MELSEC-Q series-compatible E71	GX Works2: Network Parameter Ethernet/CC IE Field ⇨ Network Type: Ethernet ⇨ [Initial Settings] ⇨ Timer Settings
MELSEC-L series-compatible E71	
MELSEC-Q series C Controller module	Inapplicable (Timeout value of KeepAlive: 30 seconds)

5

■ readDeviceBlock (To read devices in a batch)

MELSEC iQ-R series	MELSEC iQ-F series	MELSEC-Q series	MELSEC-L series
○	○	○	○

Item	Description	
Class name	MELMxCommunication	

Method definition	int readDeviceBlock(
	        String deviceName,
	        int size,
	        int[] readDatas
	        )	

Argument	[in]deviceName	Specify a device name which includes the start reading device number.
	[in]size	Specify a number of read points. Up to 960 points can be read.
	[out]readDatas	Read device values are stored.
Return value	MX_SUCCESS: Succeed Except for MX_SUCCESS: Failed (☞ Page 68 Error Code List)	
Function	Use this when reading sequential device values. Reads device values for the number of points specified to 'size' (argument) from the device specified to 'deviceName' (argument), and stores them to 'readDatas' (argument) in a batch.	
	Device specification method (when specifying a bit device)	Example: Read the data for three points (three words) from M0 deviceName="M0", size=3 ↓ Data storage example (one element of an array: 32-bit) readDatas[0]: M0 to M15 readDatas[1]: M16 to M31 readDatas[2]: M32 to M47 • Since the data of bit devices are stored in the elements of an array (32-bit) in a word unit, the upper 2-bytes are padded with '0'. • The data storage example of bit devices differs depending on the system environment. For details, refer to the following section. ☞ Page 33 Considerations for bit devices
	Device specification method (when specifying a word device)	Example: Read the data for three points from D0 deviceName="D0", size=3 ↓ Data storage example (one element of an array: 32-bit) readDatas[0]: D0 readDatas[1]: D1 readDatas[2]: D2 • Since the data is stored in the elements of an array (32-bit), the upper 2-bytes are padded with '0'.
Reference	open(), close(), writeDeviceBlock()	
Considerations	Device name	The following device names cannot be specified. An error occurs if specified. • Long timer: LTN (Current value) • Long retentive timer: LSTN (Current value) • Long counter: LCN (Current value) • Index register: LZ
	Device extension specification	The devices used the extension specification cannot be specified. (Example: 'U3E1\G0' cannot be specified.)
	Number of read points	The maximum number of read points is in the range that satisfies the following formula. Start reading device number + number of read points ≤ end device number
	Bit device specification	When specifying a bit device, only the multiples of 16 can be specified for a device number.
	Specify the number of elements equal or greater than that specified to 'size' (argument) for 'readDatas' (argument).	

■ writeDeviceBlock (To write devices in a batch)

MELSEC iQ-R series	MELSEC iQ-F series	MELSEC-Q series	MELSEC-L series
○	○	○	○

Item	Description	
Class name	MELMxCommunication	
Method definition	int writeDeviceBlock(　　String deviceName, 　　int size, 　　int[] writeDatas 　　)	
Argument	[in]deviceName	Specify a device name which includes the start writing device number.
	[in]size	Specify a number of write points. Up to 960 points can be written.
	[in]writeDatas	Specify an array of device values to be written.
Return value	MX_SUCCESS: Succeed Except for MX_SUCCESS: Failed (☞ Page 68 Error Code List)	
Function	Use this when writing sequential device values. Writes device values for the number of points specified to 'size' (argument) from the device specified to 'deviceName' (argument) in a batch. Specify device values, which is to be written, to 'writeDatas' (argument)	
	Device specification method (when specifying a bit device)	Example: Write the data for three points (three words) from M0 deviceName="M0", size=3 ↓ Data storage example (one element of an array: 32-bit) writeDatas[0]: M0 to M15 writeDatas[1]: M16 to M31 writeDatas[2]: M32 to M47 　• Since the data of bit devices are stored in the elements of an array (32-bit) in a word unit, the upper 2-bytes are padded with '0'. 　• The data storage example of bit devices differs depending on the system environment. For details, refer to the following section. 　☞ Page 33 Considerations for bit devices
	Device specification method (when specifying a word device)	Example: Write the data for three points from D0 deviceName="D0", size=3 ↓ Data storage example (one element of an array: 32-bit) writeDatas[0]: D0 writeDatas[1]: D1 writeDatas[2]: D2 　• Since the data is stored in the elements of an array (32-bit), the upper 2-bytes are padded with '0'.
Reference	open(), close(), readDeviceBlock()	
Considerations	Device name	The following device names cannot be specified. An error occurs if specified. 　• Long timer: LTN (Current value) 　• Long retentive timer: LSTN (Current value) 　• Long counter: LCN (Current value) 　• Index register: LZ
	Device extension specification	The devices used the extension specification cannot be specified. (Example: 'U3E1\G0' cannot be specified.)
	Number of write points	The maximum number of write points is in the range that satisfies the following formula. Start writing device number + number of write points ≤ end device number
	Bit device specification	When specifying a bit device, only the multiples of 16 can be specified for a device number.
	Specify the number of elements equal or greater than that specified to 'size' (argument) for 'writeDatas' (argument).	

5

■ readDeviceRandom (To read devices randomly)

MELSEC iQ-R series	MELSEC iQ-F series	MELSEC-Q series	MELSEC-L series
○	○	○	○

Item	Description	
Class name	MELMxCommunication	

Method definition	int readDeviceRandom(
	    String[] deviceNames,
	    int[] readDatas
	    )	

Argument	[in]deviceNames	Specify a device name, which includes a device number, in an array. In this method, the number of elements of an array is treated as the number of read points. Therefore, specifying the number of read points with an argument is not required. Up to 96 points can be read.
	[out]readDatas	Read device values are stored.
Return value	MX_SUCCESS: Succeed Except for MX_SUCCESS: Failed (☞ Page 68 Error Code List)	
Function	Use this when reading values by specifying multiple devices individually. Reads device values from the multiple devices specified to 'deviceNames' (argument), and stores them to each element of 'readDatas' (argument).	
	Device specification method	Example: Read the device values of M0, D10, and LZ20 deviceNames="M0", "D10", "LZ20" ↓ Data storage example (one element of an array: 32-bit) readDatas[0]: M0 (ON/OFF is judged in bit 0.) readDatas[1]: D10 readDatas[2]: LZ20
Reference	open(), close(), writeDeviceRandom()	
Considerations	• Devices and the devices used the extension specification cannot be specified together. (Example: 'U3E1\G0' and 'D0' are not acceptable to specify together.) When specifying the devices used the extension specification, specify that devices only to 'deviceNames' (argument). • Specify the number of elements equal or greater than that of read points to 'readDatas' (argument).	

■ writeDeviceRandom (To write devices randomly)

MELSEC iQ-R series	MELSEC iQ-F series	MELSEC-Q series	MELSEC-L series
○	○	○	○

Item	Description	
Class name	MELMxCommunication	

Method definition	int writeDeviceRandom(
	    String[] deviceNames,
	    int[] writeDatas
	    )	

Argument	[in]deviceNames	Specify a device name, which includes a device number, in an array. In this method, the number of elements of an array is treated as the number of write points. Therefore, specifying the number of write points with an argument is not required. The maximum number of write points is as follows: RCPU: 68 points QCPU/LCPU/FX5CPU: 80 points
	[in]writeDatas	Specify device values to be written.
Return value	MX_SUCCESS: Succeed Except for MX_SUCCESS: Failed (☞ Page 68 Error Code List)	
Function	Use this when writing by specifying multiple devices individually. Writes device values to the multiple devices specified to 'deviceNames' (argument). Specify device values, which is to be written, to each element of 'writeDatas'.	
	Device specification method	Example: Write the device values of M0, D10, and LZ20 deviceNames="M0", "D10", "LZ20" ↓ Data storage example (one element of an array: 32-bit) writeDatas[0]: M0 (ON/OFF is judged in bit 0.) writeDatas[1]: D10 writeDatas[2]: LZ20
Reference	open(), close(), readDeviceRandom()	
Considerations	• Devices and the devices used the extension specification cannot be specified together. (Example: 'U3E1\G0' and 'D0' are not acceptable to specify together.) When specifying the devices used the extension specification, specify that devices only to 'deviceNames' (argument). • Specify the number of elements equal or greater than that of write points to 'writeDatas' (argument). • When bit devices and word devices (including double word devices) are specified together, they are processed as respective packets. Therefore, the timing for writing the bit devices and word devices are different. • When bit devices and word devices (including double word devices) are specified together, the bit devices are written after writing the word devices. Therefore, the writing of bit devices may fail even if the word devices successfully are written. If the error occurs in this method, write them again after checking the error content.	

5

■ readArrayLabel (To read data by specifying an array label)

MELSEC iQ-R series	MELSEC iQ-F series	MELSEC-Q series	MELSEC-L series
○	×	×	×

Item	Description	
Class name	MELMxCommunication	
Method definition	int readArrayLabel(MELMxLabel label)	
Argument	[in/out]label	Specify a MELMxLabel object in which an array label to be read has been set. For a MELMxLabel object, set a label name, character string length to be stored in a label (when the data type is the string type only), and number of pieces of data by using the method for the label data setting in advance. After this method succeeded, the read label data is stored.
Return value	MX_SUCCESS: Succeed Except for MX_SUCCESS: Failed (☞ Page 68 Error Code List)	
Function	Use this when reading array label data Reads the array label data specified to 'label' (argument) for the number of array elements. The read label data is stored to 'label' (argument).	
Reference	open(), close(), writeArrayLabel()	
Considerations	• An array label of the following data types cannot be specified; Timer, retentive timer, counter, long timer, long retentive timer, long counter • Up to three dimensions of arrays can be specified. (Note that two-dimensional arrays and three-dimensional arrays, of which data types are bit, cannot be specified with this method.) • An error occurs if the data size to be read exceeds 1914 bytes. Specify the size 1914 bytes or less. • Specify the character string length for the label name up to 1906 bytes. • 4-byte characters of Unicode (Example: 𠂉 (U+20089)) are treated as 2 bytes and 2 characters. • Data of a simple type label can be read as an array label with one element. (Example: Label[0])	

■ writeArrayLabel (To write data by specifying an array label)

MELSEC iQ-R series	MELSEC iQ-F series	MELSEC-Q series	MELSEC-L series
○	×	×	×

Item	Description	
Class name	MELMxCommunication	
Method definition	int writeArrayLabel(MELMxLabel label)	
Argument	[in]label	Specify a MELMxLabel object in which an array label to be written has been set. For a MELMxLabel object, set a label name, character string length to be stored in a label (when the data type is the string type only), label data, and number of pieces of data by using the method for the label data setting in advance.
Return value	MX_SUCCESS: Succeed Except for MX_SUCCESS: Failed (☞ Page 68 Error Code List)	
Function	Use this when writing array label data. Writes the array label data specified to 'label' (argument) for the number of array elements. Specify label data to be written to 'label' (argument).	
Reference	open(), close(), readArrayLabel()	
Considerations	• An array label of the following data types cannot be specified; Timer, retentive timer, counter, long timer, long retentive timer, long counter • Up to three dimensions of arrays can be specified. (Note that two-dimensional arrays and three-dimensional arrays, of which data types are bit, cannot be specified with this method.) • The total value of the character string length and data to be written, which can be specified to a label name, is up to 1906 bytes. • 4-byte characters of Unicode (Example: 𠂉 (U+20089)) are treated as 2 bytes and 2 characters. • Data of a simple type label can be written as an array label with one element. (Example: Label[0])	

■ readLabelRandom (To read data by specifying multiple labels)

MELSEC iQ-R series	MELSEC iQ-F series	MELSEC-Q series	MELSEC-L series
○	×	×	×

Item	Description	
Class name	MELMxCommunication	
Method definition	int readLabelRandom(MELMxLabel[] labels)	
Argument	[in/out]labels	Specify multiple MELMxLabel objects, in which labels to be read have been set, in an array. For each MELMxLabel object, set a label name, character string length to be stored in a label (when the data type is the string type only), and number of pieces of data by using the method for the label data setting in advance. After this method succeeded, the read label data is stored.
Return value	MX_SUCCESS: Succeed Except for MX_SUCCESS: Failed (☞ Page 68 Error Code List)	
Function	Use this when reading the following label data. • Simple type label • Simple type label of structure elements Reads data from a label specified to 'labels' (argument). The read label data is stored in each element of 'labels' (argument).	
Reference	open(), close(), writeLabelRandom()	

Considerations	• The size of the label data to be read needs to satisfy the following condition formula.
	  Total number of (the size of read data + 2) ≤ 1912
	• The total length of label name needs to satisfy the following condition formula.
	  Total number of (the length of label name × 2 + 2) ≤ 1910
	• 4-byte characters of Unicode (Example: 𠂉 (U+20089)) are treated as 2 bytes and 2 characters.	

■ writeLabelRandom (To write data by specifying multiple labels)

MELSEC iQ-R series	MELSEC iQ-F series	MELSEC-Q series	MELSEC-L series
○	×	×	×

Item	Description	
Class name	MELMxCommunication	
Method definition	int writeLabelRandom(MELMxLabel[] labels)	
Argument	[in]labels	Specify multiple MELMxLabel objects, in which labels to be written have been set, in an array. For each MELMxLabel object, set a label name, character string length to be stored in a label (when the data type is the string type only), label data, and number of pieces of data by using the method for the label data setting in advance.
Return value	MX_SUCCESS: Succeed Except for MX_SUCCESS: Failed (☞ Page 68 Error Code List)	
Function	Use this when writing the following label data. • Simple type label • Simple type label of structure elements Writes data to a label specified to 'labels' (argument). Specify label data to be written to each element of 'labels' (argument).	
Reference	open(), close(), readLabelRandom()	
Considerations	• The total length of label name needs to specify the following condition formula. Total number of (the length of label name × 2 + the size of write data + 4) ≤ 1910 • 4-byte characters of Unicode (Example: 𠂉 (U+20089)) are treated as 2 bytes and 2 characters.	

MELMxOpenSettings class

It is the class to manage the following parameters when communicating.

The parameters can access directly as "Public field".

Details of fields

Field name	Data type	Description	Default value	Reference
unitType	Integer	Specify a communication target module.	UNIT_RETHER(0x1002)	Page 47 unitType
ioNumber	long	Specify the start I/O No. of an access target CPU module.	0x03FF	Page 47 ioNumber
cpuType	Integer	Specify an access target CPU module.	CPU_R04CPU(0x1001)	Page 47 cpuType
protocolType	Integer	Specify a communication protocol.	PROTOCOL_TCPIP(0x0005)	Page 49 protocolType
hostAddress	String	Specify the host name (IP address) of a connection target. Only IPv4 is supported.	1.1.1.1	—
portNumber	Integer	Specify the port number of a device such as a tablet.	0x0000	—
destinationPortNumber	Integer	Specify the port number of a connection target module.	5002	Page 49 destinationPortNumber
timeOut	Integer	Specify the timeout time (ms) for communication between a CPU module and a device such as a tablet.	10000(0x2710)	—
cpuTimeOut	Integer	Set the CPU monitoring timer. (Units: 250 ms) Example: 2500 ms for 10 The available range is 0 to 0xFFFF. Set '0' if the value that is outside of range has been set. When specifying '0', it will be an infinity wait. Fix '0' for an FX5CPU.	40	—

■ unitType

Connected module	Definition name	Value (Hexadecimal)
RCPU	UNIT_RETHER	0x1002
MELSEC iQ-R series-compatible EN71	UNIT_RJ71EN71	0x1001
FX5CPU	UNIT_FETHER	0x2001
QCPU (Built-in Ethernet port)	UNIT_QNETHER	0x002C
MELSEC-Q series C Controller module		
MELSEC-Q series-compatible E71	UNIT_QJ71E71	0x001A
LCPU (Built-in Ethernet port)	UNIT_LNETHER	0x0052
MELSEC-L series-compatible E71	UNIT_LJ71E71	0x005C

■ ioNumber

System Configuration	Access target CPU module	Value (Hexadecimal)
Single CPU system	Host CPU	0x03FF
Multiple CPU system[1]	CPU No.1	0x03E0
	CPU No.2	0x03E1
	CPU No.3	0x03E2
	CPU No.4	0x03E3

*1 In MELSEC-Q series, communication with another CPU cannot be performed via a built-in Ethernet port of a programmable controller CPU.
    To communicate with another CPU, use its built-in Ethernet port or route via an Ethernet module managed by it.

■ cpuType
• MELSEC iQ-R series CPU

Applicable CPU module	Definition name	Value (Hexadecimal)
R04CPU	CPU_R04CPU	0x1001
R08CPU	CPU_R08CPU	0x1002
R16CPU	CPU_R16CPU	0x1003
R32CPU	CPU_R32CPU	0x1004
R120CPU	CPU_R120CPU	0x1005
R04ENCPU	CPU_R04ENCPU	0x1008
R08ENCPU	CPU_R08ENCPU	0x1009
R16ENCPU	CPU_R16ENCPU	0x100A
R32ENCPU	CPU_R32ENCPU	0x100B
R120ENCPU	CPU_R120ENCPU	0x100C
R08PCPU	CPU_R08PCPU	0x1102
R16PCPU	CPU_R16PCPU	0x1103
R32PCPU	CPU_R32PCPU	0x1104
R120PCPU	CPU_R120PCPU	0x1105
R12CCPU-V	CPU_R12CCPU_V	0x1021
R16MTCPU	CPU_R16MTCPU	0x1011
R32MTCPU	CPU_R32MTCPU	0x1012

• FX5CPU

Applicable CPU module	Definition name	Value (Hexadecimal)
FX5UCPU	CPU_FX5UCPU	0x0210
FX5UCCPU	CPU_FX5UCCPU	0x0210

• MELSEC-Q series CPU

Applicable CPU module	Definition name	Value (Hexadecimal)
Q00JCPU	CPU_Q00JCPU	0x0030
Q00UJCPU	CPU_Q00UJCPU	0x0080
Q00CPU	CPU_Q00CPU	0x0031
Q00UCPU	CPU_Q00UCPU	0x0081
Q01CPU	CPU_Q01CPU	0x0032
Q01UCPU	CPU_Q01UCPU	0x0082
Q02(H)CPU	CPU_Q02CPU	0x0022
Q06HCPU	CPU_Q06CPU	0x0023
Q12HCPU	CPU_Q12CPU	0x0024
Q25HCPU	CPU_Q25CPU	0x0025
Q02PHCPU	CPU_Q02PHCPU	0x0045
Q06PHCPU	CPU_Q06PHCPU	0x0046
Q12PHCPU	CPU_Q12PHCPU	0x0041
Q25PHCPU	CPU_Q25PHCPU	0x0042
Q12PRHCPU	CPU_Q12PRHCPU	0x0043
Q25PRHCPU	CPU_Q25PRHCPU	0x0044
Q02UCPU	CPU_Q02UCPU	0x0083
Q03UDCPU	CPU_Q03UDCPU	0x0070
Q04UDHCPU	CPU_Q04UDHCPU	0x0071
Q06UDHCPU	CPU_Q06UDHCPU	0x0072
Q10UDHCPU	CPU_Q10UDHCPU	0x0075
Q13UDHCPU	CPU_Q13UDHCPU	0x0073
Q20UDHCPU	CPU_Q20UDHCPU	0x0076
Q26UDHCPU	CPU_Q26UDHCPU	0x0074
Q03UDECPU	CPU_Q03UDECPU	0x0090
Q04UDEHCPU	CPU_Q04UDEHCPU	0x0091
Q06UDEHCPU	CPU_Q06UDEHCPU	0x0092
Q10UDEHCPU	CPU_Q10UDEHCPU	0x0095
Q13UDEHCPU	CPU_Q13UDEHCPU	0x0093
Q20UDEHCPU	CPU_Q20UDEHCPU	0x0096
Q26UDEHCPU	CPU_Q26UDEHCPU	0x0094
Q50UDEHCPU	CPU_Q50UDEHCPU	0x0098
Q100UDEHCPU	CPU_Q100UDEHCPU	0x009A
Q03UDVCPU	CPU_Q03UDVCPU	0x00D1
Q04UDVCPU	CPU_Q04UDVCPU	0x00D2
Q06UDVCPU	CPU_Q06UDVCPU	0x00D3
Q13UDVCPU	CPU_Q13UDVCPU	0x00D4
Q26UDVCPU	CPU_Q26UDVCPU	0x00D5
Q12DCCPU-V	CPU_Q12DC_V	0x0058
Q24DHCCPU-V	CPU_Q24DHC_V	0x0059
Q24DHCCPU-LS	CPU_Q24DHC_LS	0x005B
Q172DCPU	CPU_Q172DCPU	0x0625
Q173DCPU	CPU_Q173DCPU	0x0626
Q172DSCPU	CPU_Q172DSCPU	0x062A
Q173DSCPU	CPU_Q173DSCPU	0x062B

- LCPU

Applicable CPU module	Definition name	Value (Hexadecimal)
L02SCPU	CPU_L02SCPU	0x00A3
L02SCPU-P	CPU_L02SCPU	0x00A3
L02CPU	CPU_L02CPU	0x00A1
L02CPU-P	CPU_L02CPU	0x00A1
L06CPU	CPU_L06CPU	0x00A5
L06CPU-P	CPU_L06CPU	0x00A5
L26CPU	CPU_L26CPU	0x00A4
L26CPU-P	CPU_L26CPU	0x00A4
L26CPU-BT	CPU_L26CPUBT	0x00A2
L26CPU-PBT	CPU_L26CPUBT	0x00A2

■ protocolType

Applicable protocol	Definition name	Value (Hexadecimal)
TCP/IP	PROTOCOL_TCPIP	0x0005

■ destinationPortNumber

When the communication method (open method)[1] is SLMP or MC protocol, specify the same port number as that of the connection target module.

When the communication method (open method)[1] is MELSOFT connection, specify the port number as follows:

Connected module	TCP/IP
RCPU	5007
MELSEC iQ-R series-compatible EN71	5002
MELSEC-Q series-compatible E71	5002
MELSEC-L series-compatible E71	5002

*1 For the communication method (open method), refer to the following section.
☞ Page 25 Communication method (open method)

MELMxLabel class

It is the class to manage the label data for a label name.

Use this class when using the methods for reading and writing labels (readArrayLabel/writeArrayLabel/readLabelRandom/writeLabelRandom of MELMxCommunication class).

Details of methods

■ MELMxLabel (Constructor)

Item	Description
Class name	MELMxLabel
Method definition	void MELMxLabel()
Argument	None
Return value	None
Function	It is the constructor of this class. The default value of label name is null character. Set "MEL_MX_LABEL_DATATYPE_BIT" for the default value of a data type, and set "false" for the default value of a value. The default value of a point is 1. The default value of the string data length for a label, of which data type is 'string [32] or 'string' [Unicode] (32), is 32 characters.
Reference	None
Considerations	None

■ getValues (To acquire label data)

| Item	Description
Class name	MELMxLabel
Method definition	Object[] getValues()
Argument	None
Return value	Label data
Function	Acquires label data that is managed in a instance of this class. • Use this class to acquire the label data which was read with 'readLabelRandom'. (☞ Page 45 readLabelRandom (To read data by specifying multiple labels)) Acquired data is the 'Object' instance. Before using the data, convert it to the type of a variable corresponding to the data type acquired with the method for acquiring a data type (getDataType). (☞ Page 51 getDataType (To acquire the data type of a label)) For details on the types of variables, refer to the following section. ☞ Page 33 Data types of labels
Reference	None
Considerations	When 4-byte characters of Unicode (Example: 𠂉(U+20089)) are included when label data is the Unicode character string, they are treated as 2 bytes and 2 characters.

■ getName (To acquire a label name)

| Item	Description
Class name	MELMxLabel
Method definition	String getName()
Argument	None
Return value	Label name
Function	Acquires a label name specified with the methods of label data settings.
Reference	MELMxLabel()
Considerations	When 4-byte characters of Unicode (Example: 𠂉(U+20089)) are included in a label name, they are treated as 2 bytes and 2 characters.

■ getDataType (To acquire the data type of a label)

Item	Description
Class name	MELMxLabel
Method definition	int getDataType()
Argument	None
Return value	Data type of a label
Function	Acquires the data type of a label. For details on the types of variables, refer to the following section. ☞ Page 33 Data types of labels
Reference	None
Considerations	None

■ setBitLabel (To set the label data of Bit type)

Item	Description	
Class name	MELMxLabel	

Method definition	void setBitLabel(
	    String name,
	    Boolean[] values,
	    int size
	    )	

Argument	[in]name	Specify a label name.
	[in]values	Specify label data.
	[in]size	Specify the number of pieces of label data. For a simple type label, specify '1'. For an array label, specify a number of elements.
Return value	None	
Function	Set a label name, type, and label data to be managed in a instance of this class. Use this method when using a label of which the data type is Bit. (☞ Page 33 Data types of labels)	
Reference	getValues()	
Considerations	None	

■ setWordLabel (To set the label data of Word (Signed))

Item	Description	
Class name	MELMxLabel	

Method definition	void setWordLabel(
	    String name,
	    Integer[] values,
	    int size
	    )	

Argument	[in]name	Specify a label name.
	[in]values	Specify label data.
	[in]size	Specify the number of pieces of label data. For a simple type label, specify '1'. For an array label, specify a number of elements.
Return value	None	
Function	Set a label name, type, and label data to be managed in a instance of this class. Use this method when using a label of which the data type is Word (Signed). (☞ Page 33 Data types of labels)	
Reference	getValues()	
Considerations	None	

■ setUnsignedWordLabel (To set the label data of Word (Unsigned))

Item	Description	
Class name	MELMxLabel	

Method definition	void setUnsignedWordLabel(
	    String name,
	    Integer[] values,
	    int size
	    )	

Argument	[in]name	Specify a label name.
	[in]values	Specify label data.
	[in]size	Specify the number of pieces of label data. For a simple type label, specify '1'. For an array label, specify a number of elements.
Return value	None	
Function	Set a label name, type, and label data to be managed in a instance of this class. Use this method when using a label of which the data type is Word (Unsigned). (☞ Page 33 Data types of labels)	
Reference	getValues()	
Considerations	None	

■ setDoubleWordLabel (To set the label data of Double Word (Signed))

Item	Description	
Class name	MELMxLabel	

Method definition	void setDoubleWordLabel(
	    String name,
	    Long[] values,
	    int size
	    )	

Argument	[in]name	Specify a label name.
	[in]values	Specify label data.
	[in]size	Specify the number of pieces of label data. For a simple type label, specify '1'. For an array label, specify a number of elements.
Return value	None	
Function	Set a label name, type, and label data to be managed in a instance of this class. Use this method when using a label of which the data type is Double Word (Signed). (☞ Page 33 Data types of labels)	
Reference	getValues()	
Considerations	None	

■ setUnsignedDoubleWordLabel (To set the label data of Double Word (Unsigned))

Item	Description	
Class name	MELMxLabel	

Method definition	void setUnsignedDoubleWordLabel(
	    String name,
	    Long values,
	    int size
	    )	

Argument	[in]name	Specify a label name.
	[in]values	Specify label data.
	[in]size	Specify the number of pieces of label data. For a simple type label, specify '1'. For an array label, specify a number of elements.
Return value	None	
Function	Set a label name, type, and label data to be managed in a instance of this class. Use this method when using a label of which the data type is Double Word (Unsigned). (☞ Page 33 Data types of labels)	
Reference	getValues()	
Considerations	None	

■ setFloatSingleLabel (To set the label data of FLOAT (Single Precision))

Item	Description	
Class name	MELMxLabel	
Method definition	void setFloatSingleLabel(　　　String name, 　　　Float[] values, 　　　int size 　　　)	
Argument	[in]name	Specify a label name.
	[in]values	Specify label data.
	[in]size	Specify the number of pieces of label data. For a simple type label, specify '1'. For an array label, specify a number of elements.
Return value	None	
Function	Set a label name, type, and label data to be managed in a instance of this class. Use this method when using a label of which the data type is FLOAT (Single Precision). (☞ Page 33 Data types of labels)	
Reference	getValues()	
Considerations	None	

■ setFloatDoubleLabel (To set the label data of FLOAT (Double Precision))

Item	Description	
Class name	MELMxLabel	
Method definition	void setFloatDoubleLabel(　　　String name, 　　　Double[] values, 　　　int size 　　　)	
Argument	[in]name	Specify a label name.
	[in]values	Specify label data.
	[in]size	Specify the number of pieces of label data. For a simple type label, specify '1'. For an array label, specify a number of elements.
Return value	None	
Function	Set a label name, type, and label data to be managed in a instance of this class. Use this method when using a label of which the data type is FLOAT (Double Precision). (☞ Page 33 Data types of labels)	
Reference	getValues()	
Considerations	None	

■ setAsciiStringLabel (To set the label data of ASCII character string)

Item	Description	
Class name	MELMxLabel	
Method definition	void setAsciiStringLabel(　　　String name, 　　　Long length, 　　　String[] values, 　　　int size 　　　)	
Argument	[in]name	Specify a label name.
	[in]length	Specify the character string length to be stored in a label.
	[in]values	Specify label data.
	[in]size	Specify the number of pieces of label data. For a simple type label, specify '1'. For an array label, specify a number of elements.
Return value	None	
Function	Set a label name, type, and label data to be managed in a instance of this class. Use this method when using a label of which the data type is String [32]. (☞ Page 33 Data types of labels)	
Reference	getValues()	
Considerations	None	

■ setUnicodeStringLabel (To set the label data of Unicode character string)

Item	Description	
Class name	MELMxLabel	

Method definition	void setUnicodeStringLabel(
	        String name,
	        Long length,
	        String[] values,
	        int size
	        )	

Argument	[in]name	Specify a label name.
	[in]length	Specify the character string length to be stored in a label.
	[in]values	Specify label data.
	[in]size	Specify the number of pieces of label data. For a simple type label, specify '1'. For an array label, specify a number of elements.
Return value	None	
Function	Set a label name, type, and label data to be managed in a instance of this class. Use this method when using a label of which the data type is String [Unicode] (32). (☞ Page 33 Data types of labels)	
Reference	getValues()	
Considerations	When 4-byte characters of Unicode (Example: 𠂉(U+20089)) are included in a label name and label data, they are treated as 2 bytes and 2 characters.	

■ setTimeLabel (To set the label data of Time)

Item	Description	
Class name	MELMxLabel	

Method definition	void setTimeLabel(
	        String name,
	        Long[] values,
	        int size
	        )	

Argument	[in]name	Specify a label name.
	[in]values	Specify label data.
	[in]size	Specify the number of pieces of label data. For a simple type label, specify '1'. For an array label, specify a number of elements.
Return value	None	
Function	Set a label name, type, and label data to be managed in a instance of this class. Use this method when using a label of which the data type is Time. (☞ Page 33 Data types of labels)	
Reference	getValues()	
Considerations	None	

MELMxErrDefine class

It is the class to retain an error which occurs in this library.

An error code is retained in 'public static const' field.

For details on error codes, refer to the following section.

☞ Page 68 Error codes returned by MX Component library

5.3 Considerations for Using Methods

When a remote password has been set, unlock the password (open method) before communication.

The remote password is locked when the communication is completed (close method).

5.4   Sample Program

This section explains the sample programs in the provided CD.

The sample programs differ depending on the development environment and CPU modules.

Use one of them according to the development environment and a CPU module to be accessed.

Folder name				Description
SampleProject	AndroidStudio or Eclipse	iQ-F	PLC	Sample for MELSEC iQ-F series FX5CPUs
		iQ-R	PLC_Motion	Sample for MELSEC iQ-R series programmable controller CPUs and motion CPUs
			C_Controller	Sample for MELSEC iQ-R series C Controller modules
		Q	PLC_Motion	Sample for MELSEC-Q/L series programmable controller CPUs and motion CPUs
			C_Controller	Sample for MELSEC-Q series C Controller modules

Created application

The following shows an example of the screen when an application, created by using a sample program, is run with a tablet. Methods are executed by tapping the buttons on the left side of the screen. After executing them, the result is displayed on the right side of the screen.

When the processing succeeded, "SUCCESS" and the value of execution result are displayed. If the processing failed, the error code appears.

The following table shows the processing corresponding to each button on the screen above.

Button name	Processing
Open	Page 58 execOpen
Close	Page 59 execClose
ReadDeviceBlock	Page 60 execReadDeviceBlock
WriteDeviceBlock	Page 61 execWriteDeviceBlock
ReadDeviceRandom	Page 62 execReadDeviceRandom
WriteDeviceRandom	Page 63 execWriteDeviceRandom
ReadArrayLabel	Page 64 execReadArrayLabel
WriteArrayLabel	Page 65 execWriteArrayLabel
ReadLabelRandom	Page 66 execReadLabelRandom
WriteLabelRandom	Page 67 execWriteLabelRandom

Operation for application

The following shows the basic operation for communication.

Operating procedure

1. Start the application.

2. Tap the [Open] button.

3. Tap the button for reading and writing of devices and labels.

4. Tap the [Close] button.

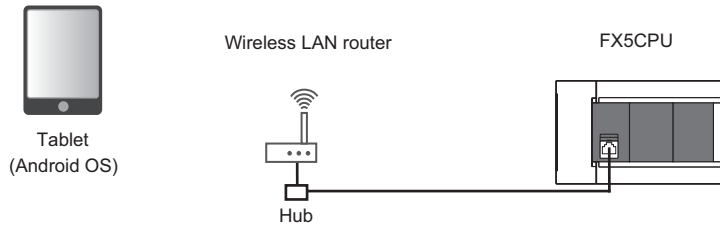If tap the [ClearRecord] button, the execution result is cleared.

Operating environment

Each sample program is created to operate in the following environment.

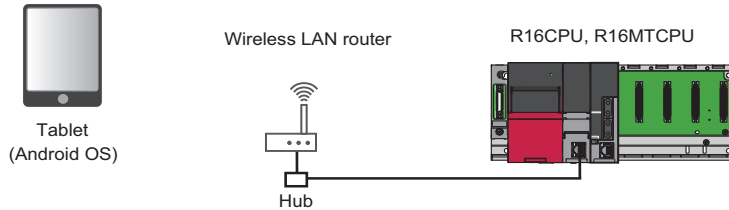| Folder name	Item		Description	System configuration
iQ-F	PLC	CPU	FX5UCPU (Host CPU)	☞ Page 57 iQ-F PLC
		hostAddress	192.168.2.100	
		destinationPortNumber	5014	
iQ-R	PLC_Motion	CPU	R16CPU (Host CPU), R16MTCPU (CPU No.2)	☞ Page 57 iQ-R PLC_Motion
		hostAddress	192.168.2.100	
		destinationPortNumber	5007	
	C_Controller	CPU	R16CPU (Host CPU), R12CCPU-V (CPU No.2)	☞ Page 57 iQ-R C_Controller
		hostAddress	192.168.2.100	
		destinationPortNumber	5007	
Q	PLC_Motion	CPU	Q20UDEHCPU (Host CPU), Q172DSCPU (CPU No.2)), QJ71E71-100	☞ Page 57 Q PLC_Motion
		hostAddress	192.168.2.100	
		destinationPortNumber	5002	
	C_Controller	CPU	Q12DCCPU-V (Host CPU)	☞ Page 57 Q C_Controller
		hostAddress	192.168.2.100	
		destinationPortNumber	5010	

System configuration

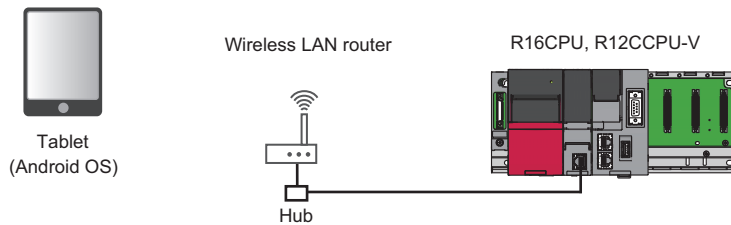Each sample program is created to operate in the following system configuration.
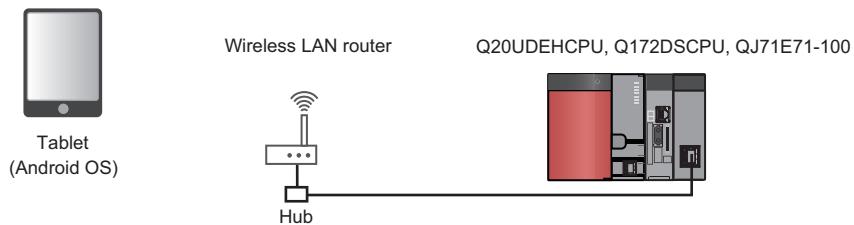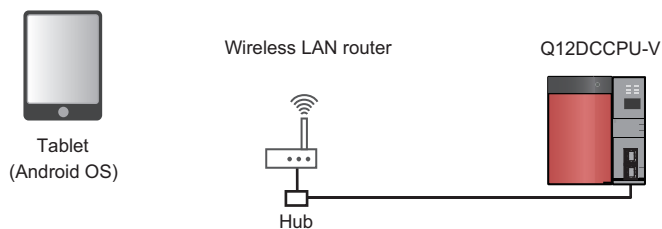
■ iQ-F PLC

Tablet (Android OS) — Wireless LAN router — Hub — FX5CPU

■ iQ-R PLC_Motion

Tablet (Android OS) — Wireless LAN router — Hub — R16CPU, R16MTCPU

■ iQ-R C_Controller

Tablet (Android OS) — Wireless LAN router — Hub — R16CPU, R12CCPU-V

■ Q PLC_Motion

Tablet (Android OS) — Wireless LAN router — Hub — Q20UDEHCPU, Q172DSCPU, QJ71E71-100

■ Q C_Controller

Tablet (Android OS) — Wireless LAN router — Hub — Q12DCCPU-V

For the communication method (open method), refer to the following section.

Sample method

The methods explained in this section are included in a "MXComponentManager.java" file.

Development environment	Storage folder
Android Studio	SampleProject ⇨ AndroidStudio ⇨ (folder for each series) ⇨ (folder for each module) ⇨ MXComponentSample ⇨ src ⇨ main ⇨ java ⇨ jp ⇨ co ⇨ melco ⇨ mxcomponentsample
Eclipse	SampleProject ⇨ Eclipse ⇨ (folder for each series) ⇨ (folder for each module) ⇨ src ⇨ jp ⇨ co ⇨ melco ⇨ mxcomponentsample

Processing corresponding to each button

The following shows the processing corresponding to each button.

■ execOpen

The following processing are executed by tapping the [Open] button.
- The open method of a MELMxCommunication class is executed in the background.
- The elapsed time is counted. When the specified timeout time is over, a timeout error is returned.

```java
public void execOpen(int seqno) {

    //[check command exetcute state ]
    if(commandexecute)return;

    //[set command exetcute state]
    commandexecute=true;

    final int seq=seqno;

    //[Call MX "Open" by Background Thread]
    new AsyncTask<Void, Void, Integer>() {

        //[Set Process Start Time]
        long start = System.currentTimeMillis();

        @Override
        protected Integer doInBackground(Void... arg0) {

            //[call "Open" API]
            return  mxcomm.open(mxopen, password);
        }
        @Override
        protected void onPostExecute(Integer result){

            //[Check Pointer(not nul)]
            if (null != callback) {

                //[set DetailResultString Host/port String]
                detailstring=getOpenString();

                //[Set Process Duration]
                long interval = System.currentTimeMillis()-start;

                //[Call result callback]
                callback.resultMXComponent(seq,(int)result,interval,detailstring);

            }
            //[set command exetcute state]
            commandexecute=false;

            return ;

        }
    }.executeOnExecutor(AsyncTask.THREAD_POOL_EXECUTOR);

}
```

■ execClose

The following processing are executed by tapping the [Close] button.

・ The close method of a MELMxCommunication class is executed in the background.

・ The elapsed time is counted. When the specified timeout time is over, a timeout error is returned.

```java
public void execClose(int seqno) {

    //[check command exetcute state ]
    if(commandexecute)return;

    //[set command exetcute state]
    commandexecute=true;

    final int seq=seqno;

    //[Call MX "Close" by Background Thread]
    new AsyncTask<Void, Void, Integer>() {

        //[Set Process Start Time]
        long start = System.currentTimeMillis();

        @Override
        protected Integer doInBackground(Void... arg0) {

            //[call "Close" API]
            return  mxcomm.close();
        }
        @Override
        protected void onPostExecute(Integer result){

            //[Check Pointer(not nul)]
            if (null != callback) {

                //[set DetailResultString blank]
                detailstring="";

                //[Set Process Duration]
                long interval = System.currentTimeMillis()-start;

                //[Call result callback]
                callback.resultMXComponent(seq,(int)result,interval,detailstring);

            }
            //[set command exetcute state]
            commandexecute=false;

            return ;

        }
    }.executeOnExecutor(AsyncTask.THREAD_POOL_EXECUTOR);

}
```

■ execReadDeviceBlock

The following processing are executed by tapping the [ReadDeviceBlock] button.

- A value is read from the device 'D100' of the opened connection target and displayed.
- The elapsed time is counted. When the specified timeout time is over, a timeout error is returned.

```java
public void execReadDeviceBlock(int seqno) {

    //[check command exetcute state ]
    if(commandexecute)return;

    //[set command exetcute state]
    commandexecute=true;

    final int seq=seqno;

    //[Call MX "Read Device Block" by Background Thread]
    new AsyncTask<Void, Void, Integer>() {

        //[Set Process Start Time]
        long start = System.currentTimeMillis();

        @Override
        protected Integer doInBackground(Void... arg0) {

            //[call "Read Device Block" API]
            //[ex. Device:"D100" size:"1"]
            int readdata[]=new int[1];
            int result = mxcomm.readDeviceBlock("D100",1,readdata);

            //[if API Success Set DetailResultString Readed data("fault" set blank)]
            if(result==0)
                detailstring="D100= [" + String.valueOf(readdata[0]) +"] ";
            else
                detailstring="";

            return result;
        }
        @Override
        protected void onPostExecute(Integer result){

            //[Check Pointer(not nul)]
            if (null != callback) {

                //[Set Process Duration]
                long interval = System.currentTimeMillis()-start;

                //[Call result callback]
                callback.resultMXComponent(seq,(int)result,interval,detailstring);
            }
            //[set command exetcute state]
            commandexecute=false;

            return ;
        }
    }.executeOnExecutor(AsyncTask.THREAD_POOL_EXECUTOR);
}
```

■ execWriteDeviceBlock

The following processing are executed by tapping the [WriteDeviceBlock] button.

・A random value from 0 to 99 is written to the device 'D100' of the opened connection target.

・The elapsed time is counted. When the specified timeout time is over, a timeout error is returned.

```java
public void execWriteDeviceBlock(int seqno) {

    //[check command exetcute state ]
    if(commandexecute)return;

    //[set command exetcute state]
    commandexecute=true;

    final int seq=seqno;

    //[Call MX "Write Device Block" by Background Thread]
    new AsyncTask<Void, Void, Integer>() {

        //[Set Process Start Time]
        long start = System.currentTimeMillis();

        @Override
        protected Integer doInBackground(Void... arg0) {


            int writedata[]=new int[1];

            Random rnd = new Random();
            writedata[0]= (short)rnd.nextInt(100);


            //[call "Write Device Block" API]
            //[ex. Device:"D100" size:"1"]
            int result = mxcomm.writeDeviceBlock("D100",1,writedata);

            //[Set DetailResultString Write data]
            detailstring="D100= [" + String.valueOf(writedata[0]) +"] ";

            return result;

        }
        @Override
        protected void onPostExecute(Integer result){

            //[Check Pointer(not nul)]
            if (null != callback) {


                //[Set Process Duration]
                long interval = System.currentTimeMillis()-start;

                //[Call result callback]
                callback.resultMXComponent(seq,(int)result,interval,detailstring);


            }
            //[set command exetcute state]
            commandexecute=false;

            return ;


        }
    }.executeOnExecutor(AsyncTask.THREAD_POOL_EXECUTOR);

}
```

■ execReadDeviceRandom

The following processing are executed by tapping the [ReadDeviceRandom] button.

- Values are read from the devices 'D100', 'M0', and 'LZ0' of the opened connection target and displayed.
- For a bit device, the execution result is displayed in ON or OFF.
- The elapsed time is counted. When the specified timeout time is over, a timeout error is returned.

```java
public void execReadDeviceRandom(int seqno) {

    //[check command exetcute state ]
    if(commandexecute)return;

    //[set command exetcute state]
    commandexecute=true;

    final int seq=seqno;

    //[Call MX "Read Device Random" by Background Thread]
    new AsyncTask<Void, Void, Integer>() {

        //[Set Process Start Time]
        long start = System.currentTimeMillis();

        @Override
        protected Integer doInBackground(Void... arg0) {

            //[call "Read Device Random" API]
            //[ex. Device:"D100"/"M0"/"LZ0"]
            String devicenames[] = { "D100","M0","LZ0" };
            int readdata[]=new int[3];

            int result = mxcomm.readDeviceRandom(devicenames,readdata);

            //[if API Success Set DetailResultString Readed data("fault" set blank)]
            if(result==0)
                detailstring="D100= [" + String.valueOf(readdata[0]) +"]  " +
                        "M0= ["+ String.valueOf(readdata[1]==0?"OFF":"ON") +"] " +
                        " LZ0= ["+ String.valueOf(readdata[2]) +"] ";
            else
                detailstring="";

            return result;

        }
        @Override
        protected void onPostExecute(Integer result){

            //[Check Pointer(not nul)]
            if (null != callback) {

                //[Set Process Duration]
                long interval = System.currentTimeMillis()-start;

                //[Call result callback]
                callback.resultMXComponent(seq,(int)result,interval,detailstring);

            }
            //[set command exetcute state]
            commandexecute=false;

            return ;

        }
    }.executeOnExecutor(AsyncTask.THREAD_POOL_EXECUTOR);

}
```

■ execWriteDeviceRandom

The following processing are executed by tapping the [WriteDeviceRandom] button.

• Random values are written to the devices 'D100', 'M0', and 'LZ0' of the opened connection target.

• For a bit device, the execution result is displayed in ON or OFF.

• The elapsed time is counted. When the specified timeout time is over, a timeout error is returned.

```java
public void execWriteDeviceRandom(int seqno) {

    //[check command exetcute state ]
    if(commandexecute)return;

    //[set command exetcute state]
    commandexecute=true;

    final int seq=seqno;

    //[Call MX "Write Device Random" by Background Thread]
    new AsyncTask<Void, Void, Integer>() {

        //[Set Process Start Time]
        long start = System.currentTimeMillis();

        @Override
        protected Integer doInBackground(Void... arg0) {

            //[call "Read Device Random" API]
            //[ex. Device:"D100"/"M0"/"LZ0"]
            String devicenames[] = { "D100","M0","LZ0" };
            int writedata[]=new int[3];
            //[Make Write Data 0～99 pseudorandom]
            Random rnd = new Random();
            writedata[0]= (short)rnd.nextInt(100);
            //[Make Write Data 0～1 pseudorandom]
            rnd = new Random();
            writedata[1]= (short)rnd.nextInt(2);
            //[Make Write Data 0～9999999 pseudorandom]
            rnd = new Random();
            writedata[2]= (short)rnd.nextInt(10000000);

            int result = mxcomm.writeDeviceRandom(devicenames,writedata);

            //[Set DetailResultString Write data]
            detailstring="D100= [" + String.valueOf(writedata[0]) +"] " +
                " M0= ["+ (writedata[1]==0?"OFF":"ON") +"] " +
                " LZ0= ["+ String.valueOf(writedata[2]) +"] ";

            return result;

        }
        @Override
        protected void onPostExecute(Integer result){

            //[Check Pointer(not nul)]
            if (null != callback) {

                //[Set Process Duration]
                long interval = System.currentTimeMillis()-start;

                //[Call result callback]
                callback.resultMXComponent(seq,(int)result,interval,detailstring);

            }
            //[set command exetcute state]
            commandexecute=false;

            return ;

        }
    }.executeOnExecutor(AsyncTask.THREAD_POOL_EXECUTOR);

}
```

■ execReadArrayLabel

The following processing are executed by tapping the [ReadArrayLabel] button.

・A value is read from 'wLabel[0]'[*1] of the opened connection target and displayed.

・The elapsed time is counted. When the specified timeout time is over, a timeout error is returned.

*1 'wLabel' is an array label of Word type.

```java
public void execReadArrayLabel(int seqno) {

    //[check command exetcute state ]
    if(commandexecute)return;

    //[set command exetcute state]
    commandexecute=true;

    final int seq=seqno;

    //[Call MX "Read Array Label" by Background Thread]
    new AsyncTask<Void, Void, Integer>() {

        //[Set Process Start Time]
        long start = System.currentTimeMillis();

        @Override
        protected Integer doInBackground(Void... arg0) {

            //[call "Read Array Label" API]
            //[ex. Label:"wLabel(WordType) Size:1]
            MELMxLabel label= new MELMxLabel();
            try{
                label.setWordLabel("wLabel[0]",null,1);
            }catch(Exception e){
                detailstring="";
                return 0xFFFFFFFF;
            }
            int result =mxcomm.readArrayLabel(label);


            //[if API Success Set DetailResultString Readed data("fault" set blank)]
            if(result==0)
                detailstring="wLabel= [" + String.valueOf( label.getValues()[0] ) +"] ";
            else
                detailstring="";

            return result;

        }
        @Override
        protected void onPostExecute(Integer result){

            //[Check Pointer(not nul)]
            if (null != callback) {

                //[Set Process Duration]
                long interval = System.currentTimeMillis()-start;

                //[Call result callback]
                callback.resultMXComponent(seq,(int)result,interval,detailstring);

            }
            //[set command exetcute state]
            commandexecute=false;

            return ;

        }
    }.executeOnExecutor(AsyncTask.THREAD_POOL_EXECUTOR);

}
```

■ execWriteArrayLabel

The following processing are executed by tapping the [WriteArrayLabel] button.

- A random value from 0 to 99 is written to 'wLabel[0]'[*1] of the opened connection target.
- The elapsed time is counted. When the specified timeout time is over, a timeout error is returned.

*1   'wLabel' is an array label of Word type.

```java
public void execWriteArrayLabel(int seqno) {

    //[check command exetcute state ]
    if(commandexecute)return;

    //[set command exetcute state]
    commandexecute=true;

    final int seq=seqno;

    //[Call MX "Write Array Label" by Background Thread]
    new AsyncTask<Void, Void, Integer>() {

        //[Set Process Start Time]
        long start = System.currentTimeMillis();

        @Override
        protected Integer doInBackground(Void... arg0) {

            //[call "Write Array Label" API]
            //[ex. Label:"wLabel(WordType) Size:1]
            MELMxLabel label= new MELMxLabel();

            //[Make Write Data 0~99 pseudorandom]
            Integer writedata[]=new Integer[1];
            Random rnd = new Random();
            writedata[0]= Integer.valueOf((int)rnd.nextInt(100));
            try{
                label.setWordLabel("wLabel[0]",writedata,1);
            }catch(Exception e){
                detailstring="";
                return 0xFFFFFFFF;
            }
            int result =mxcomm.writeArrayLabel(label);


            //[if API Success Set DetailResultString Readed data("fault" set blank)]
            detailstring="wLabel= [" + String.valueOf( writedata[0] ) +"] ";


            return result;

        }
        @Override
        protected void onPostExecute(Integer result){

            //[Check Pointer(not nul)]
            if (null != callback) {


                //[Set Process Duration]
                long interval = System.currentTimeMillis()-start;

                //[Call result callback]
                callback.resultMXComponent(seq,(int)result,interval,detailstring);

            }
            //[set command exetcute state]
            commandexecute=false;

            return ;

        }
    }.executeOnExecutor(AsyncTask.THREAD_POOL_EXECUTOR);

}
```

■ execReadLabelRandom

The following processing are executed by tapping the [ReadLabelRandom] button.

- Values are read from 'wLabel[0]', 'bLabel[0]', and 'sLabel[0]'[*1] of the opened connection target and displayed.

- The elapsed time is counted. When the specified timeout time is over, a timeout error is returned.

*1 'wLabel' is an array label of Word type. 'bLabel' is an array label of Bit type. 'sLabel' is an array label of String type.

```java
public void execReadLabelRandom(int seqno) {

    //[check command exetcute state ]
    if(commandexecute)return;

    //[set command exetcute state]
    commandexecute=true;

    final int seq=seqno;

    //[Call MX "Read Label Random" by Background Thread]
    new AsyncTask<Void, Void, Integer>() {

        //[Set Process Start Time]
        long start = System.currentTimeMillis();

        @Override
        protected Integer doInBackground(Void... arg0) {

            //[call "Read Label Random" API]
            //[ex. Label:"wLabel(WordType)"/"bLabel(BitType)"/"sLabel(UnicodeString:32bytes)" Size:1]
            MELMxLabel[] labels = new MELMxLabel[3];

            //[Word Label Class Setting]
            MELMxLabel wlabel= new MELMxLabel();
            try{
                wlabel.setWordLabel("wLabel[0]",null,1);
            }catch(Exception e){
                detailstring="";
                return 0xFFFFFFFF;
            }
            labels[0]=wlabel;

            //[Bit Label Class Setting]
            MELMxLabel blabel= new MELMxLabel();
            try{
                blabel.setBitLabel("bLabel[0]",null,1);
            }catch(Exception e){
                detailstring="";
                return 0xFFFFFFFF;
            }
            labels[1]=blabel;

            //[UnicodeString Label Class Setting]
            MELMxLabel slabel= new MELMxLabel();
            try{
                slabel.setUnicodeStringLabel("sLabel[0]",32L,null,1);
            }catch(Exception e){
                detailstring="";
                return 0xFFFFFFFF;
            }
            labels[2]=slabel;

            int result =mxcomm.readLabelRandom(labels);

            //[if API Success Set DetailResultString Readed data("fault" set blank)]
            if(result==0){
                Boolean bdata=(Boolean)labels[1].getValues()[0];

                detailstring="wLabel[0]= [" + String.valueOf( labels[0].getValues()[0] ) +"]  bLabel[0]= ["
                            + (bdata==false?"OFF":"ON")   +"] sLabel[0]= ["
                            + String.valueOf( labels[2].getValues()[0] ) +"] ";
            }else
                detailstring="";

            return result;
        }
        @Override
        protected void onPostExecute(Integer result){

            //[Check Pointer(not nul)]
            if (null != callback) {

                //[Set Process Duration]
                long interval = System.currentTimeMillis()-start;

                //[Call result callback]
                callback.resultMXComponent(seq,(int)result,interval,detailstring);
            }
            //[set command exetcute state]
            commandexecute=false;

            return ;
        }
    }.executeOnExecutor(AsyncTask.THREAD_POOL_EXECUTOR);
}
```

■ execWriteLabelRandom

The following processing are executed by tapping the [WriteLabelRandom] button.

- Random values are written to 'wLabel[0]', 'bLabel[0]', and 'sLabel[0]'[*1] of the opened connection target.
- The elapsed time is counted. When the specified timeout time is over, a timeout error is returned.

*1   'wLabel' is an array label of Word type. 'bLabel' is an array label of Bit type. 'sLabel' is an array label of String type.

```java
public void execWriteLabelRandom(int seqno) {

    //[check command exetcute state ]
    if(commandexecute)return;

    //[set command exetcute state]
    commandexecute=true;

    final int seq=seqno;

    //[Call MX "Write Label Random" by Background Thread]
    new AsyncTask<Void, Void, Integer>() {

        //[Set Process Start Time]
        long start = System.currentTimeMillis();

        @Override
        protected Integer doInBackground(Void... arg0) {

            //[call "Write Label Random" API]
            //[ex. Label:"wLabel(WordType)"/"bLabel(BitType)"/"sLabel(UnicodeString:32bytes)" Size:1]
            MELMxLabel[] labels = new MELMxLabel[3];

            //[Word Label Class Setting]
            MELMxLabel wlabel= new MELMxLabel();
            Integer wdata[]=new Integer[1];
            try{
                Random rnd = new Random();
                wdata[0]= Integer.valueOf(rnd.nextInt(100));
                wlabel.setWordLabel("wLabel[0]",wdata,1);
            }catch(Exception e){
                detailstring="";
                return 0xFFFFFFFF;
            }
            labels[0]=wlabel;

            //[Bit Label Class Setting]

            MELMxLabel blabel= new MELMxLabel();
            Boolean bdata[]=new Boolean[1];
            try{
                Random rnd = new Random();
                bdata[0]= rnd.nextInt(2)%2==0?Boolean.valueOf(false):Boolean.valueOf(true);

                blabel.setBitLabel("bLabel[0]",bdata,1);
            }catch(Exception e){
                detailstring="";
                return 0xFFFFFFFF;
            }
            labels[1]=blabel;

            //[UnicodeString Label Class Setting]
            MELMxLabel slabel= new MELMxLabel();
            String sdata[]=new String[1];
            try{
                sdata[0]=getRandomString();
                slabel.setUnicodeStringLabel("sLabel[0]",32L,sdata,1);
            }catch(Exception e){
                detailstring="";
                return 0xFFFFFFFF;
            }
            labels[2]=slabel;

            int result =mxcomm.writeLabelRandom(labels);

            //[if API Success Set DetailResultString Readed data("fault" set blank)]
            detailstring="wLabel[0]= [" + String.valueOf( wdata[0] ) +"] bLabel[0]= ["
                        + (bdata[0]==false?"OFF":"ON")  +"] sLabel[0]= ["
                        + String.valueOf( sdata[0] ) +"] ";

            return result;
        }
        @Override
        protected void onPostExecute(Integer result){

            //[Check Pointer(not nul)]
            if (null != callback) {

                //[Set Process Duration]
                long interval = System.currentTimeMillis()-start;

                //[Call result callback]
                callback.resultMXComponent(seq,(int)result,interval,detailstring);
            }
            //[set command exetcute state]
            commandexecute=false;

            return ;

        }
    }.executeOnExecutor(AsyncTask.THREAD_POOL_EXECUTOR);
}
```

6 TROUBLESHOOTING

This chapter explains the error code contents and corrective actions.

6.1 Abnormal Processing and Corrective Actions

An error code is returned to an abnormal processing.

For error codes and corrective actions, refer to the following section.

☞ Page 68 Error Code List

6.2 Error Code List

This section shows the error code contents and corrective actions.

Error codes returned by a CPU module or a module

The lower 4 digits of an error code indicate whether the error code is for a CPU module or a module.

For details, refer to the manual of each module.

Error code	Connection destination		Reference
	Series	**Module**	
0x010A0000 to 0x010AFFFF	MELSEC iQ-R series	Programmable controller CPU	⌷MELSEC iQ-R Programmable Controller CPU Module User's Manual ⌷MELSEC iQ-R Process CPU Module User's Manual
		C Controller module[*1]	⌷MELSEC iQ-R C Controller Module User's Manual
		Motion CPU[*1]	⌷MELSEC iQ-R Motion Controller Programming Manual (Common)
	MELSEC-Q series	Programmable controller CPU	⌷QCPU User's Manual (Hardware Design, Maintenance and Inspection)
		C Controller module[*1]	⌷MELSEC-Q C Controller Module User's Manual
		Motion CPU[*1]	⌷Q173D(S)CPU/Q172D(S)CPU Motion controller Programming Manual (COMMON)
	MELSEC-L series	Programmable controller CPU	⌷MELSEC-L CPU Module User's Manual (Hardware Design, Maintenance and Inspection)
0x01090000 to 0x0109FFFF	MELSEC iQ-F series	Programmable controller CPU	⌷MELSEC iQ-F FX5 User's Manual (Application)
0x010C0000 to 0x010CFFFF	MELSEC iQ-R series	Programmable controller CPU/Ethernet module	⌷MELSEC iQ-R Ethernet User's Manual (Application)
	MELSEC iQ-F series	Programmable controller CPU	⌷MELSEC iQ-F FX5 User's Manual (Ethernet Communication)
	MELSEC-Q series	Programmable controller CPU	⌷QnUCPU User's Manual (Communication via Built-in Ethernet Port)
		Ethernet module	⌷Q Corresponding Ethernet Interface Module User's Manual (Basic)
	MELSEC-L series	Programmable controller CPU	⌷MELSEC-L CPU Module User's Manual (Built-In Ethernet Function)
		Ethernet module	⌷MELSEC-L Ethernet Interface Module User's Manual (Basic)

*1   In a multiple CPU system, refer to the manuals of each module.

Error codes returned by MX Component library

The following error code values are the values obtained by converting the return values of integer type in hexadecimal.

• 0xF0000000 to 0xF1FFFFFF, 0xFE000001 to 0xFE00FFFF: MX Component for Android library itself

• 0x01800000 to 0x0180FFFF, 0x01900000 to 0x0190FFFF: EasySocket Communication

Error code	Error description	Corrective action
0x00000000	Normal completion	—
0x01800001	No command error The specified CPU module or module does not support the corresponding function.	• Check if the CPU module or module supports the function.

Error code	Error description	Corrective action
0x01800003	Memory securing error	• End other running applications, and then restart the application created by using MX Component for Android. • Restart the device such as a tablet.
0x01800004	Load error	• Check if the specified unitType, cpuType, and protocolType match the system configuration used. • End other running applications, and then restart the application created by using MX Component for Android. • Restart the device such as a tablet. • If the error occurs again even after taking the corrective actions mentioned above, please consult your local Mitsubishi representative.
0x01801002	Multi-line open error	• End other running applications, and then restart the application created by using MX Component for Android. • Restart the device such as a tablet.
0x01801003	Open not yet executed	
0x01801006	Specified module error	• Check if the specified unitType matches the system configuration used. • End other running applications, and then restart the application created by using MX Component for Android. • Restart the device such as a tablet.
0x01801007	Specified CPU error	• Check if the specified cpuType matches the system configuration used. • End other running applications, and then restart the application created by using MX Component for Android. • Restart the device such as a tablet.
0x01801009	Setting file open failed	• End other running applications, and then restart the application created by using MX Component for Android. • Restart the device such as a tablet. • Reinstall MX Component library. • If the error occurs again even after taking the corrective actions mentioned above, please consult your local Mitsubishi representative.
0x0180100B	Protocol type error The specified protocol is incorrect.	• Check if the specified protocolType matches the system configuration used. • End other running applications, and then restart the application created by using MX Component for Android. • Restart the device such as a tablet. • Reinstall MX Component library. • If the error occurs again even after taking the corrective actions mentioned above, please consult your local Mitsubishi representative.
0x01802001	Device error The device character string specified in the method is an incorrect device character string.	• Review the device name.
0x01802002	Device number error The device character string number specified in the method is an incorrect device number.	• Review the device number.
0x01802005	Size error The number of points specified in the method is incorrect.	• Check the number of points specified in the method. • Review the system such as the CPU module settings, Ethernet module settings, and cable condition. • Check the communication status between the device such as a tablet and the wireless LAN router. • End other running applications, and then restart the application created by using MX Component for Android. • Restart the device such as a tablet. • Reinstall MX Component library.
0x01802007	Receive data error The received data is abnormal.	• Review the system such as the CPU module settings, Ethernet module settings, and cable condition. • Check the communication status between the device such as a tablet and the wireless LAN router. • End other running applications, and then restart the application created by using MX Component for Android. • Restart the device such as a tablet.
0x01802079	'null' specification error 'null' is specified to the argument.	• Specify the correct value to the argument.
0x01808001	Multiple open error The open method was executed again after it was executed.	• Execute any methods except for the open method. • End other running applications, and then restart the application created by using MX Component for Android. • Restart the device such as a tablet.

6

Error code	Error description	Corrective action
0x01808007	Socket object generation error The generation of the Socket object failed.	• Check if the application which uses the same port number is running. • Change the value specified to PortNumber and retry. • End other running applications, and then restart the application created by using MX Component for Android. • Restart the device such as a tablet.
0x01808008	Port connection error Establishment of connection failed. The connection target does not respond.	• Review the values specified to PortNumber and hostAddress. • Review the system such as the CPU module settings, Ethernet module settings, and cable condition. • End other running applications, and then restart the application created by using MX Component for Android. • Restart the device such as a tablet.
0x01808101	Multiple close error	• End other running applications, and then restart the application created by using MX Component for Android. • Restart the device such as a tablet.
0x01808201	Send error Data send failed.	• Review the system such as the CPU module settings, Ethernet module settings, and cable condition. • Check the communication status between the device such as a tablet and the wireless LAN router. • Retry the method. • End other running applications, and then restart the application created by using MX Component for Android. • Restart the device such as a tablet.
0x01808301	Receive error Data receive failed.	• Review the system such as the CPU module settings, Ethernet module settings, and cable condition. • Review the value specified to timeOut. • Check the communication status between the device such as a tablet and the wireless LAN router. • Retry the method. • End other running applications, and then restart the application created by using MX Component for Android. • Restart the device such as a tablet.
0x01808404	Open not yet executed	• Execute the open method. • End other running applications, and then restart the application created by using MX Component for Android. • Restart the device such as a tablet.
0x0180840B	Time-out error Data could not be received before the timeout period had elapsed.	• Review the system such as the CPU module settings, Ethernet module settings, and cable condition. • Review the value specified to timeOut. • Check the communication status between the device such as a tablet and the wireless LAN router. • Retry the method. • Execute the close processing at once, then execute the open method again. • End other running applications, and then restart the application created by using MX Component for Android. • Restart the device such as a tablet. • Disable the battery saving function of Wi-Fi setting (depending on a device being used, the display differs such as "Wi-Fi Advanced" → "Wi-Fi optimization" or "Wi-Fi Advanced" → "Wi-Fi timer"). • Disable the auto connect/disconnect function for Wi-Fi network (depending on a device being used, the display differs such as "Wi-Fi Advanced" → "Wi-Fi optimization" or "Wi-Fi Advanced" → "Wi-Fi timer"). • End the auto disconnect applications for Wi-Fi communication.
0x01902001	Extended device mixed error Both a device and extended device are specified.	• When using an extended device, specify only the extended device.
0x01902801	Label data incorrect error The specified label data to the argument is incorrect.	• Check the number of pieces of the label data for the argument.
0x01902802	Label name length error Label name length or total number of label name length exceeds the applicable length.	• Specify the label name length or the total number of label name length within the applicable length.
0x01902803	Label data size error The data size of the label exceeds the applicable size.	• Specify the data size of the label data within the applicable range.
0x01902804	Label name length and label data size error The total size of label name length and label data exceeds the applicable size.	• Specify the total value of label name length and label data size within the applicable range.

6 TROUBLESHOOTING
6.2 Error Code List

Error code	Error description	Corrective action
0x01902805	Label data incorrect error The type of the specified label is unknown.	• Specify the data type supported the label.
0x01903801	Data type mismatch error The data type of the specified label mismatches the one set to the CPU module.	• Specify the correct label data type (that matches the one in the CPU module) and execute the method again.

0x01904001	Remote password length error
	The remote password length is not 4 bytes or exceeds 32 bytes.
		• MELSEC iQ-R series CPU, FX5CPU:
		  Set the remote password length between 6 to 32 bytes.
		• MELSEC-Q series CPU, LCPU:
		  Set the remote password length to 4 bytes.

0x01904002	Port number error The port number is wrong.	• Specify the correct portNumber and destinationPortNumber.
0x01905001	The send error of the remote password lock has occurred in the close processing when setting the remote password.	• The close processing has been completed. Execute the open method again as necessary.
0x0190FFFF	Exception occurrence error in the operating environment An exception error has occurred in the running device.	• End other running applications, and then restart the application created by using MX Component for Android. • Restart the device such as a tablet. • Reinstall MX Component library. • If the error occurs again even after taking the corrective actions mentioned above, please consult your local Mitsubishi representative.
0xF0000001	No-license error	• MX Component for Android is used without the license. Install the license of MX Component for Android duly.
0xF0000003	Already open error The open method was executed again after it was executed.	• Execute the close processing at once, then execute the open method again.
0xF0000004	Not yet open error The open method has not been executed.	• After executing the open method, execute the corresponding method.
0xF1000002	Start I/O No. error The value of specified start I/O number is incorrect. No matching start I/O number exist.	• Check the value of start I/O number to be specified to the method. • Review the system such as the CPU module settings, multiple CPU system settings, and cable condition. • End other running applications, and then restart the application created by using MX Component for Android. • Restart the device such as a tablet.
0xF1000005	Size error The size specified to the read/write method is abnormal. The 'first read/write number + size' exceeds the device or buffer area.	• Check the size specified in the method.
0xFE003001	Label name length error Label name length or total number of label name length exceeds the applicable length.	• Specify the label name length or the total number of label name length within the applicable length.
0xFE003002	Label data size error The data size of the label exceeds the applicable size.	• Specify the data size of the label data within the applicable range.
0xFE003003	Label name length and label data size error The total size of label name length and label data exceeds the applicable size.	• Specify the total value of label name length and label data size within the applicable range.
0xFE003004	Label data type error The data type of the label is unsupported.	• Change the data type of the label to the supported data type. • If the error occurs again even after taking the corrective actions mentioned above, please consult your local Mitsubishi representative.
0xFE004001	'null' is specified. 'null' is specified to the argument.	• Specify the correct value to the argument.

6

MEMO

APPENDIX

Appendix 1 Added and Changed Functions

The following table shows the added or changed functions, and the applicable software version of MX Component for Android.

Added/changed contents	Applicable software version
Supported by the following modules. • C Controller module (R12CCPU-V, Q12DCCPU-V, Q24DHCCPU-V, and Q24DHCCPU-LS) • Motion CPU (R16MTCPU, R32MTCPU, Q172DCPU, Q173DCPU, Q172DSCPU, and Q173DSCPU) • RnENCPU (R04ENCPU, R08ENCPU, R16ENCPU, R32ENCPU, and R120ENCPU)	1.01B or later
The multiple CPU system is supported.	
Windows 10 is supported.	1.02C or later
Android Studio is supported.	
Android OS 5.0.X, 5.1.X, 6.0.X are supported.	
The sample program for Android Studio is added.	
Supported by the RnPCPUs (R08PCPU, R16PCPU, R32PCPU, R120PCPU).	1.03D or later
Android OS 7.0.X is supported.	

A

MEMO

METHOD INDEX

I

MEMO

REVISIONS

Revision date	*Manual number	Description
February 2015	SH(NA)-081498ENG-A	First edition
October 2015	SH(NA)-081498ENG-B	■Added or modified parts TERMS, Section 2.1, Section 2.2, Section 2.4, Chapter 3, Section 4.1, Section 5.2, Section 5.4, Section 6.2, Appendix 1
May 2016	SH(NA)-081498ENG-C	■Added or modified parts TERMS, Section 2.2, Section 2.3, Section 3.1, Section 3.3, Section 3.5, Section 5.2, Section 5.4, Section 6.2, Appendix 1
March 2017	SH(NA)-081498ENG-D	■Added or modified parts TERMS, Section 2.2, Section 2.3, Section 3.3, Section 5.2, Appendix 1
July 2023	SH(NA)-081498ENG-E	■Added or modified parts SAFETY PRECAUTIONS, CONDITIONS OF USE FOR THE PRODUCT, Section 6.2, COPYRIGHTS

Japanese manual number: SH-081496-E

This manual confers no industrial property rights or any rights of any other kind, nor does it confer any patent licenses. Mitsubishi Electric Corporation cannot be held responsible for any problems involving industrial property rights which may occur as a result of using the contents noted in this manual.

TRADEMARKS

Android is either a registered trademark or a trademark of Google LLC.

Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

Eclipse is a trademark of Eclipse Foundation, Inc.

Java is a registered trademark of Oracle and/or its affiliates.

Microsoft and Windows are trademarks of the Microsoft group of companies.

The company names, system names and product names mentioned in this manual are either registered trademarks or trademarks of their respective companies.

In some cases, trademark symbols such as '™' or '®' are not specified in this manual.

COPYRIGHTS

The screens (screenshots) are used in accordance with the Microsoft Corporation guideline.

MITSUBISHI ELECTRIC CORPORATION

When exported from Japan, this manual does not require application to the
Ministry of Economy, Trade and Industry for service transaction permission.

Specifications subject to change without notice.