Renesas Motor Workbench

DLL for communication Function Manual

Abstract

This document explains how to use DLL abstracting the communication function with the microcomputer among the functions of RMW (Renesas Motor Workbench). For proper use, please read this document carefully.

Contents

1. DLL overview

1.1 About Renesas Motor Workbench

The motor control development support tool Renesas Motor Workbench (hereinafter referred to as "RMW") is a tool thatis a GUI application software of PC that communicates with motor control software. RMW can display numeric value of global variables of software and graphical views like wave form. RMW can write any value into global variables also. And RMW has functions which measure motor parameters and adjust control parameters.

The main functions of the RMW are as follows.

- Reading the variable list information of the execution program
- Saving environmental data to RMT files operated and set by RMWs
- Importing RMT files and importing them into various functions
- Connecting to and disconnecting from a USB-connected microcomputer
- Reading values from the MCU
- Writing values to the MCU
- Reading values continuously from the MCU
- Measurement of motor parameters (e.g. resistance).
- Adjusting control parameters of motor control.

1.2 DLL overview

This DLL is designed as one consisted by extracting functions from RMW to use with user created software (C#/ Excel VBA).

This DLL supports below functions.

- Connection and disconnection with motor control software via USB.
- Read value of global variables single and continuously.
- Write values to global variables.
- Read a list of variables.

1.3 System configuration

The system configuration of this DLL is as follows.

By executing the function of this DLL from the software (C#/ Excel VBA) created by the user, it communicates with the microcomputer via the communication board, and reads and writes the value, and controls the motor.

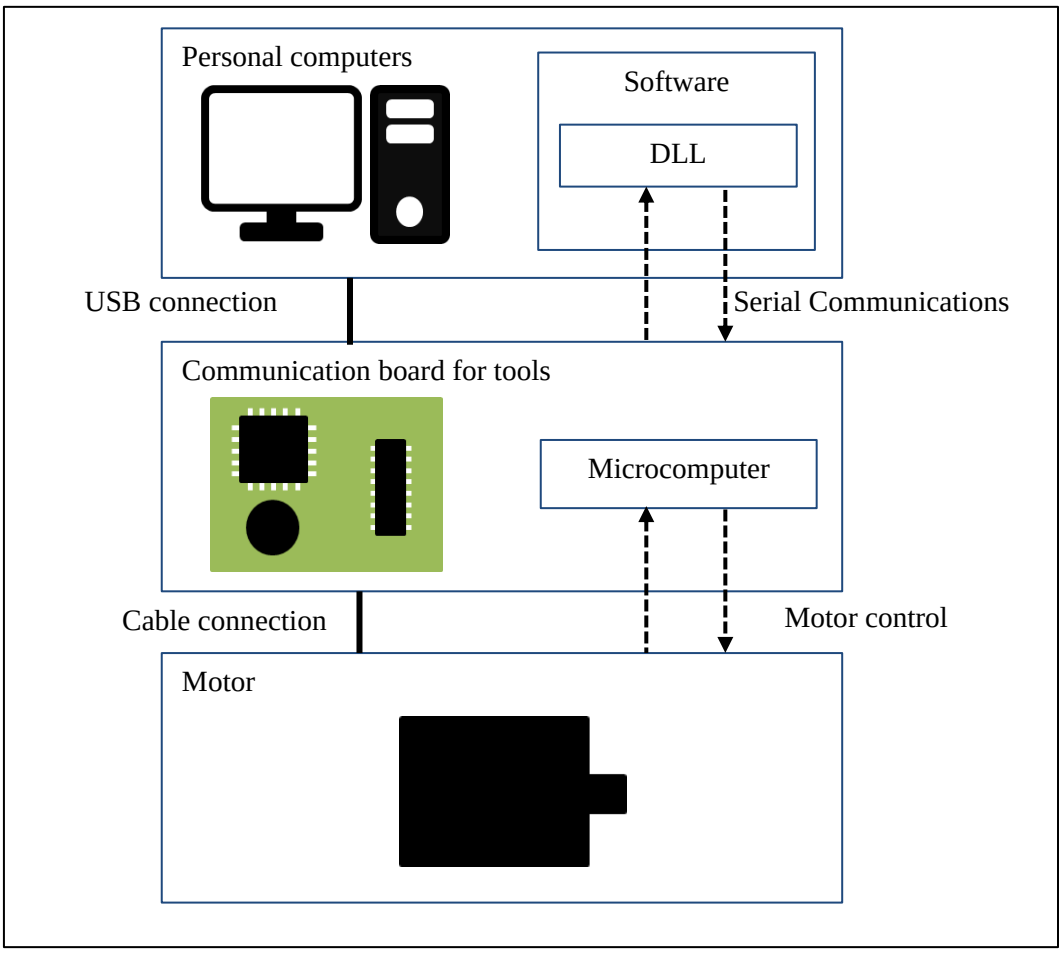


Figure 1-1 System Configuration

1.4 Operating Environment

The environment for guaranteeing the operation of this DLL is described below.

Table 1-1 System Requirements

#	Item Name	Value
1	OS	Windows 10 only
2	.Net Framework	▪ Net Framework 4 6.1 or higher
3	Means of communication	Serial port connection
4	USB driver	Standard-USB drivers for Windows10
5	Development environment	Since Visual Studio 2015, Excel
6	Supported languages	C#、Excel VBA
7	Character code	UTF-16LE

※If Net Framework needs to be updated, obtain the most up-to-date installers on the following webpage and refresh NET Framework.
https://dotnet.microsoft.com/download/dotnet-framework/net48

2. Overview of each function

2.1 Connection function

This function connects to the MCU by specifying the COM port to which the MCU is connected and the baud rate.

2.2 Disconnection function

This function disconnects from the connected MCU using the connection function.

2.3 Read function

Using the address of target variable as start address and specified the size of variables, this function can read continuous variables as a list.

2.4 Write function

Using the address of target variable and specified value, this function write the value into the target variable.

2.5 Scope function

Using the specified channel and trigger information, this function read continuous value of some variables. Therefore these can be used to draw scope graph.

2.6 Map file conversion function

This function converts the specified Map file, expands it in memory, or outputs it to a CSV file.

When a process of expanding to memory or output to a csv file is performed, reading the map file and conversion to CSV file (separated by comma) are performed automatically.

After the conversion to CSV format is completed, the file is expanded to memory or output to a CSV format file according to the executed function.

3. Introduction and deletion of DLLs

3.1 Introduction method

3.1.1 How to introduce into Visual Studio

Describes how to deploy DLLs to Visual Studio.

This DLL is a "Net Framework" compatible DLL. When using this DLL in Visual Studio, be sure to select a project whose project type is "Net Framework" instead of a project whose project type is "Net Core".

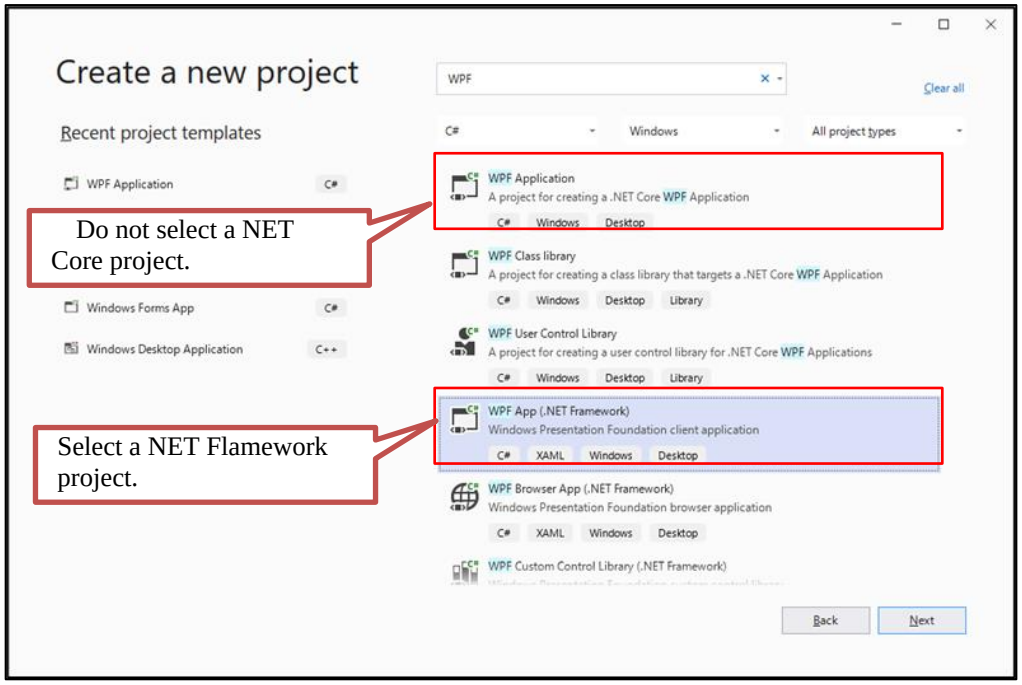


Figure 3-1 Types of Projects

To update the DLL, overwrite the DLL added to the project by "(1) Add DLL to project".(1)Add DLL to project

(1) Add DLL to project

Select “Display” tab and “Solution Explorer” from pull down menu.

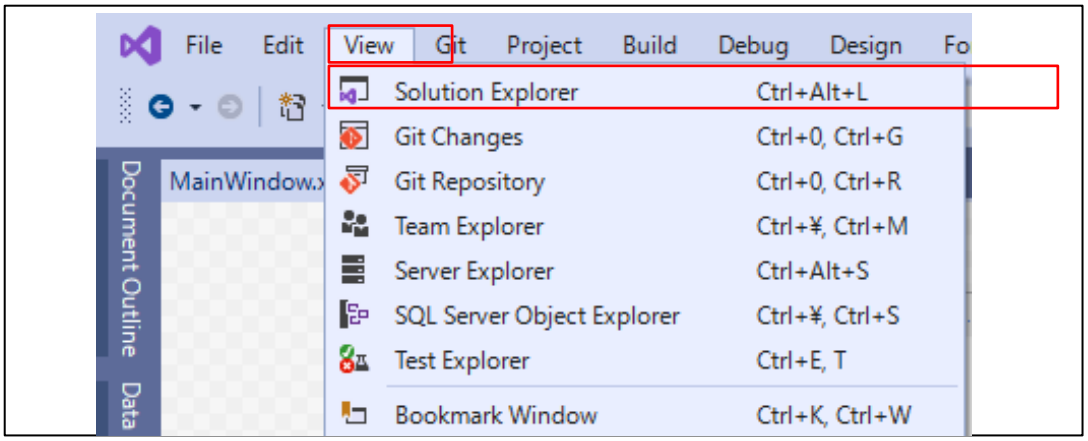


Figure 3-2 Viewing the Solution Explorer

In the Solution Explorer that appears, right-click where you want to add a project or add a new folder for the DLL, and select Add-Existing Item (G).

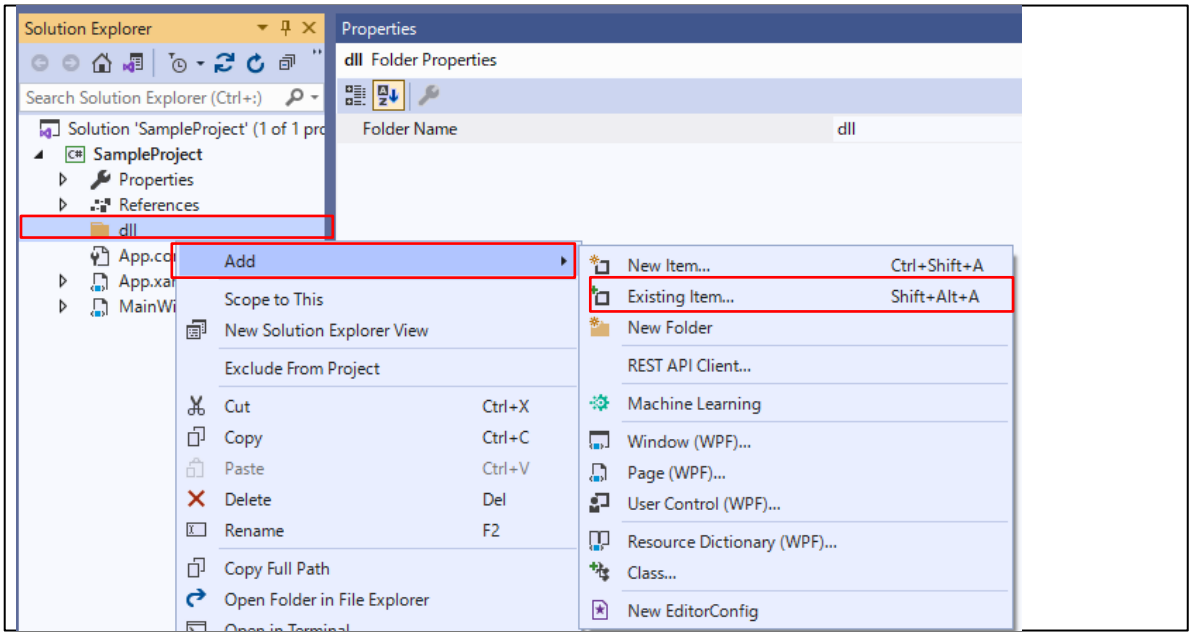


Figure 3-3 Selection in the Solution Explorer

Select "RMWCommunicationLibrary.dll" from the Add Existing Item window and click "Add".

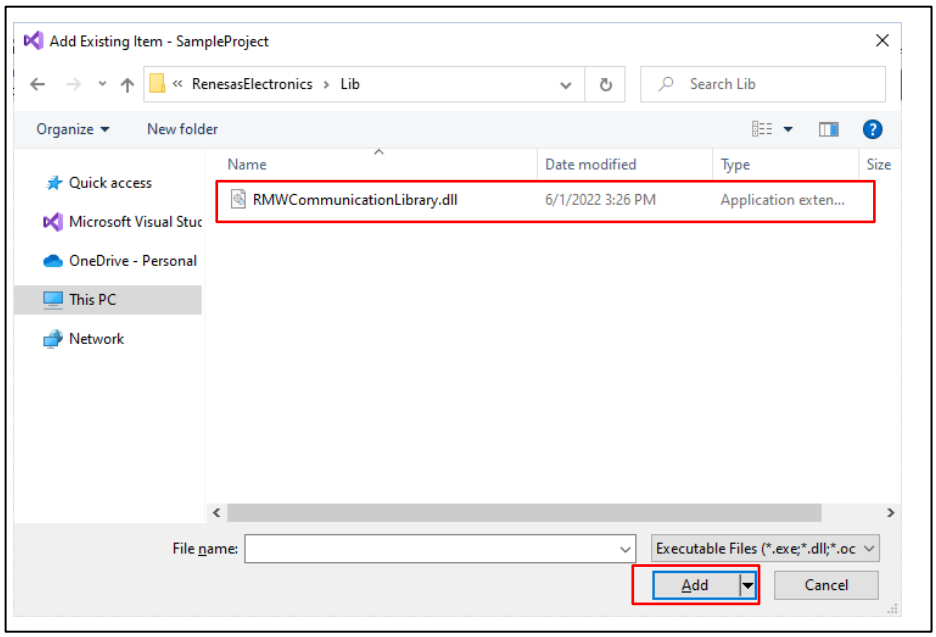


Figure 3-4 Selection of existing items on the addition screen

If the Solution Explorer is registered as follows, the registration of the DLL file to the project is completed.

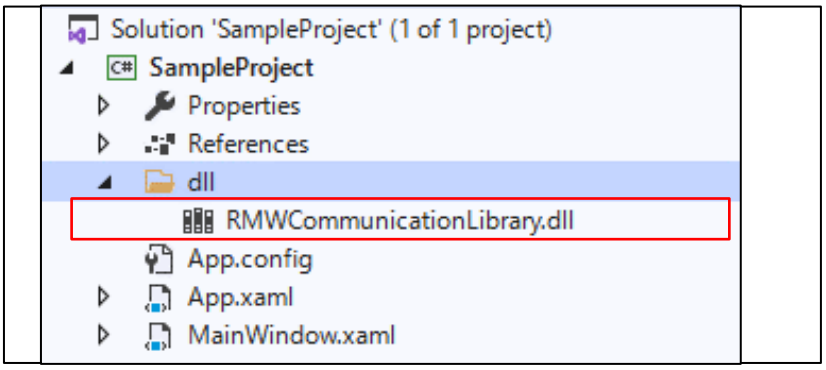


Figure 3-5 Display when registration is completed

(2) Add DLLs to Project Reference Settings

Select Project tab and Add Reference pull down menu.

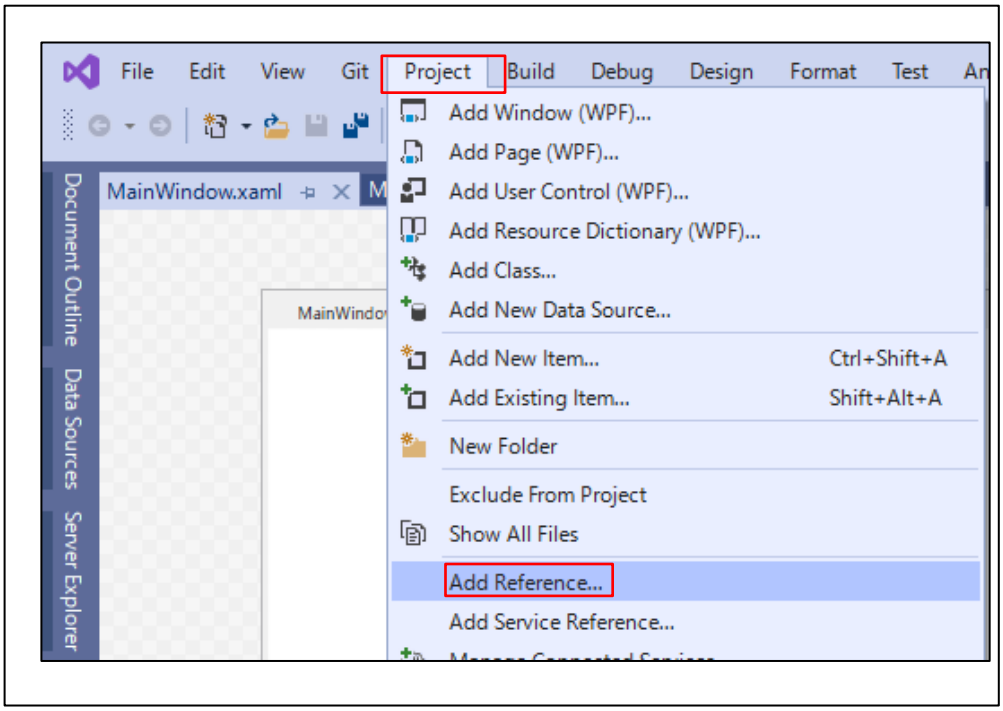


Figure 3-6 Addition to reference menu

Select "Browse..." in the lower right corner of the displayed screen.

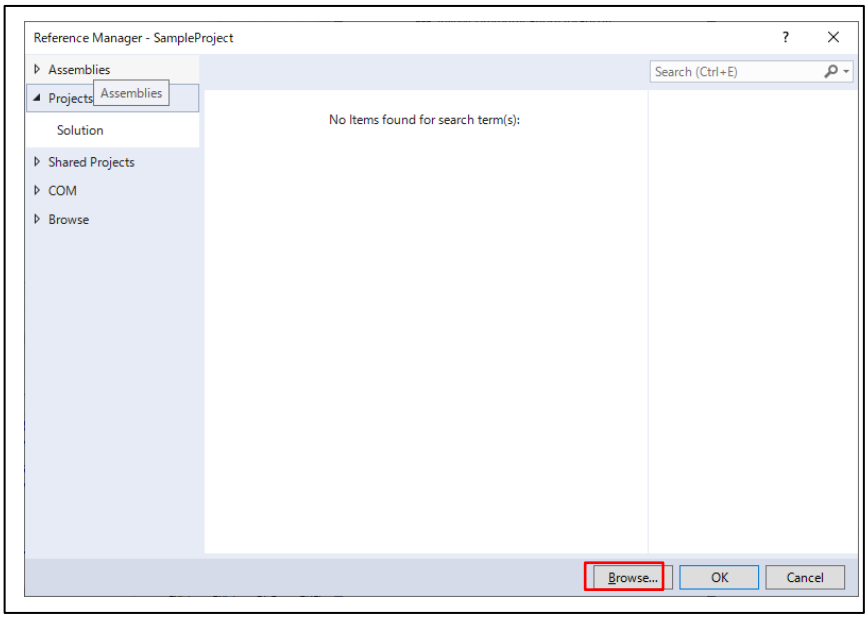


Figure 3-7 Selection at the reference manager

Select the DLL added to the project and click the "Add" button.

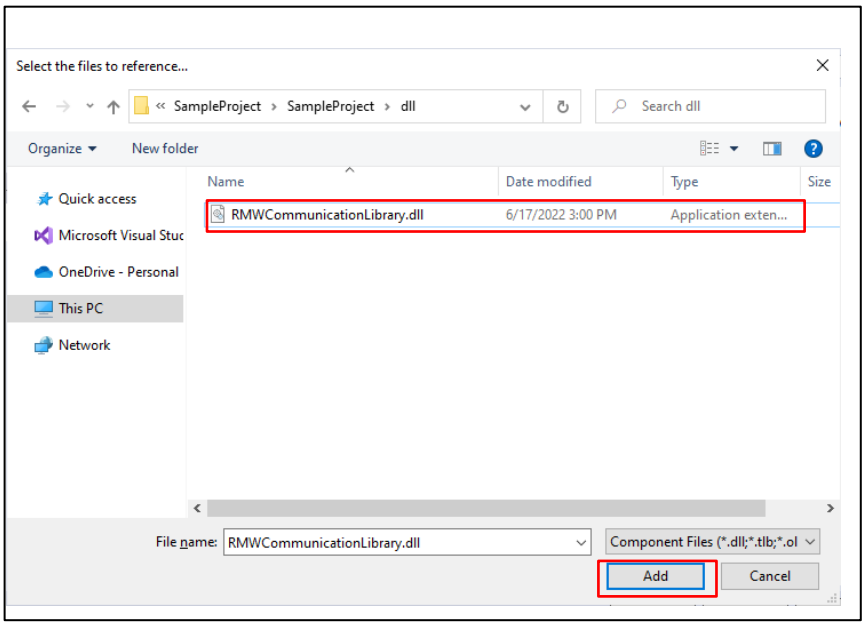


Figure 3-8 Selecting a Referenced File

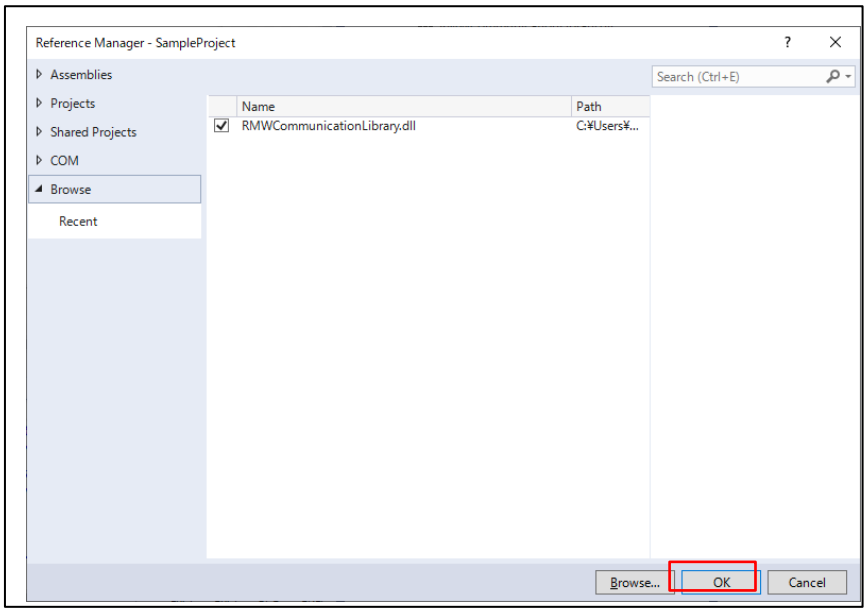


Figure 3-9 Completion of registration of reference

(3) Save the project

Select File tab and Save All pull down menu.

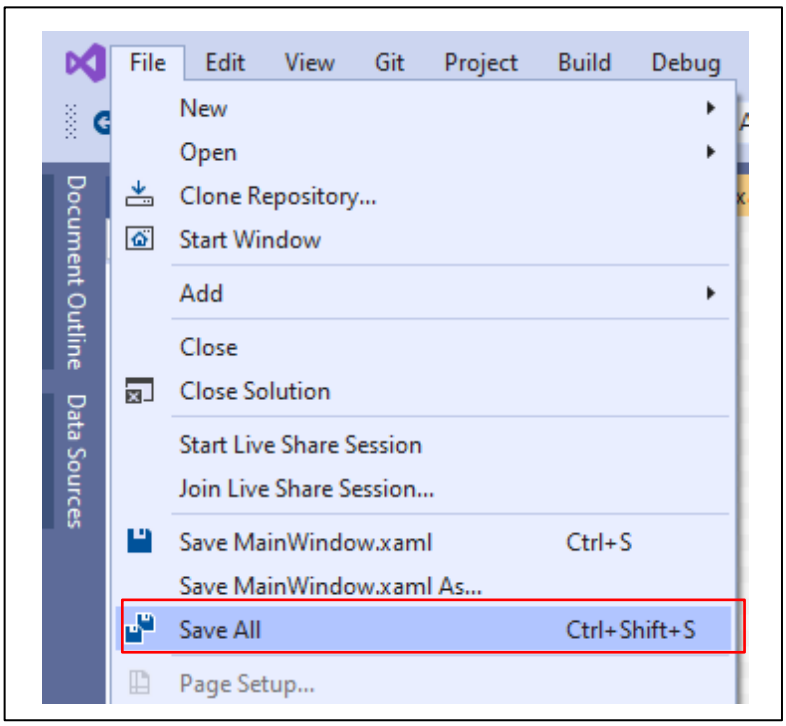


Figure 3-10 Save the project

3.1.2 How to deploy to Excel

Describes how to deploy DLLs to Excel.

To update the DLL, close all open Excel and then execute "(1) Extract the DLL to any directory".(1)Deploy DLLs to any directory

(1) Deploy DLLs to any directory

Extract the DLL to any directory.

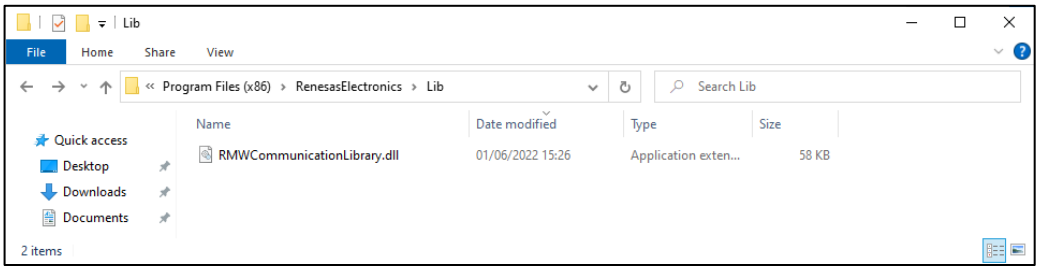


Figure 3-11 DLL Expansion

(2) Registering DLLs in the Registry

From Windows menu, select Windows System Tools.

Right-click Command Prompt and select More-Run as Administrator.

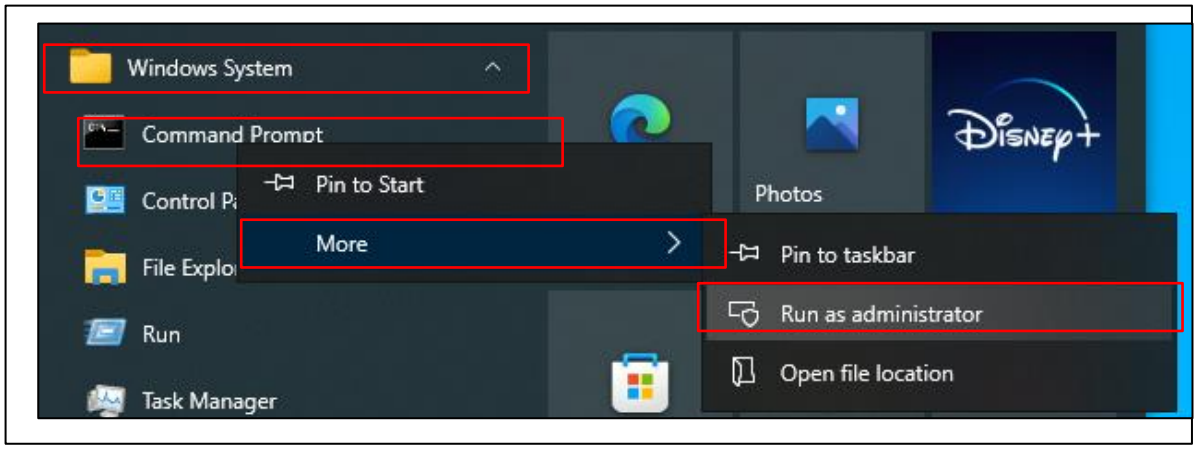


Figure 3-12 Starting the command prompt

In the command prompt, issue the following command:

[Command to be executed]

(RegAsm.exe storage path)¥RegAsm.exe /tlb /codebase "(DLL expansion destination path)¥RMWCommunicationLibrary.dll"

RegAms.exe storage paths are as follows.

64bit Excel:C:¥Windows¥Microsoft.NET¥Framework64¥v4.0.30319

32bit Excel:C:¥Windows¥Microsoft.NET¥Framework¥v4.0.30319

Change RegAsm.exe to run with Excel version, not the OS type.

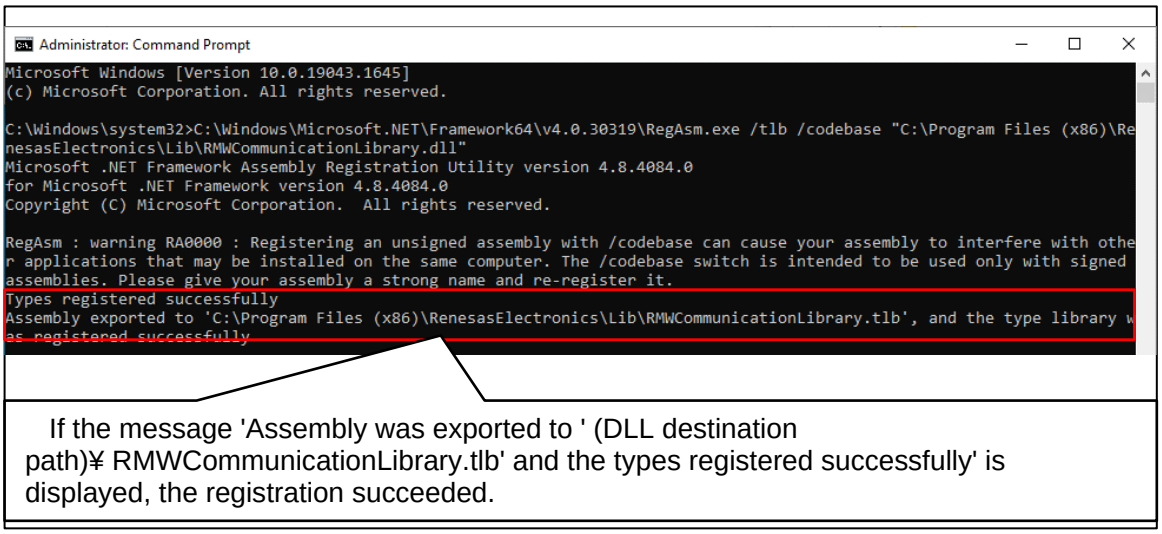


Figure 3-13 Command prompt execution screen

(3) Adding DLLs to Excel lookup settings

Select "Develop"-"Visual Basic" displayed on the ribbon of Excel.

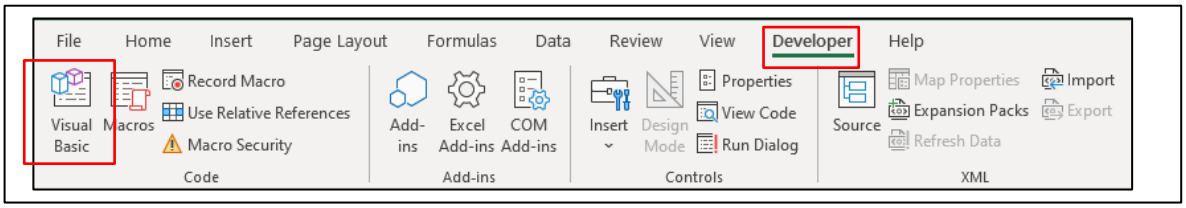


Figure 3-14 How to View Visual Basic Window

Select "Tools"-"Browse Settings" in Visual Basic window.

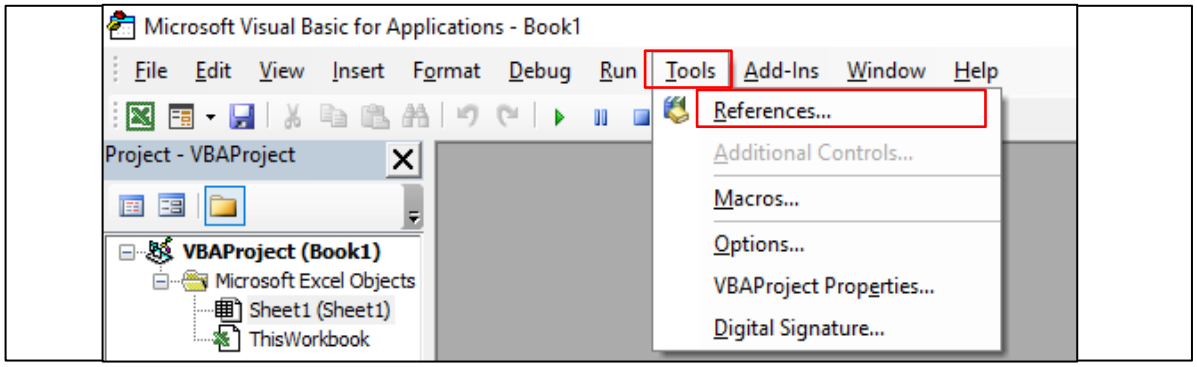


Figure 3-15 How to display the setting screen

In the displayed window, select "RMW Communication Library" and click "OK".

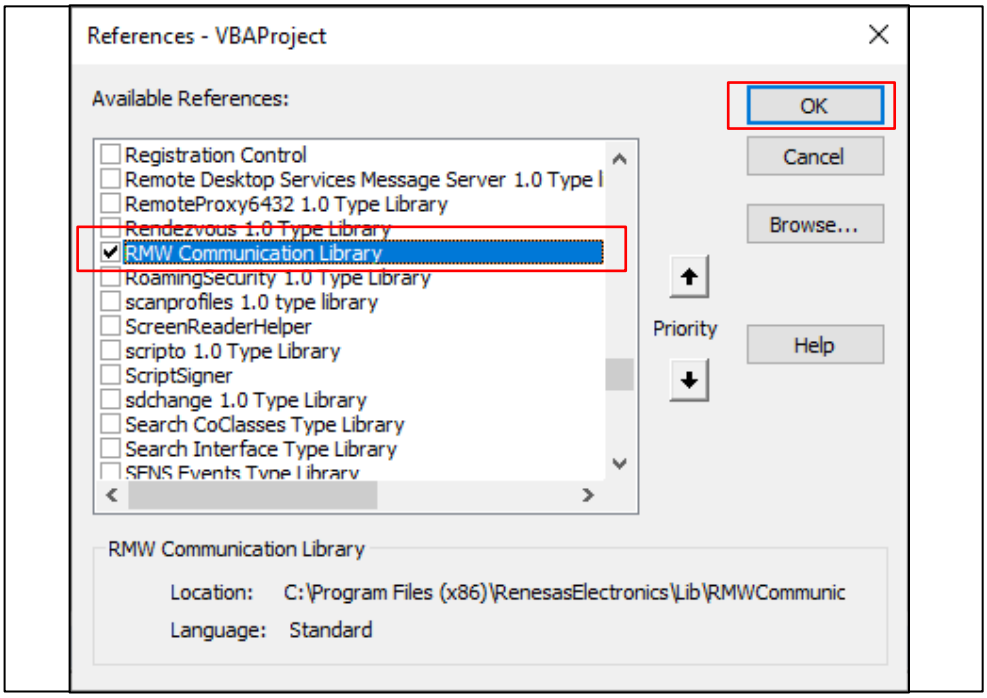


Figure 3-16 Registering RMW Communication Library

※If "RMW Communication Library" is not displayed in the above window, the DLL cannot be registered in the PC. Therefore, execute "(2) Registering the DLL in the registry" again.

3.2 Deletion method

3.2.1 How to remove from a Visual Studio

Describes how to remove DLLs from Visual Studio.

(1) Unregister a DLL from the project's reference settings

At the top of Visual Studio window, click Project-Add Browser.

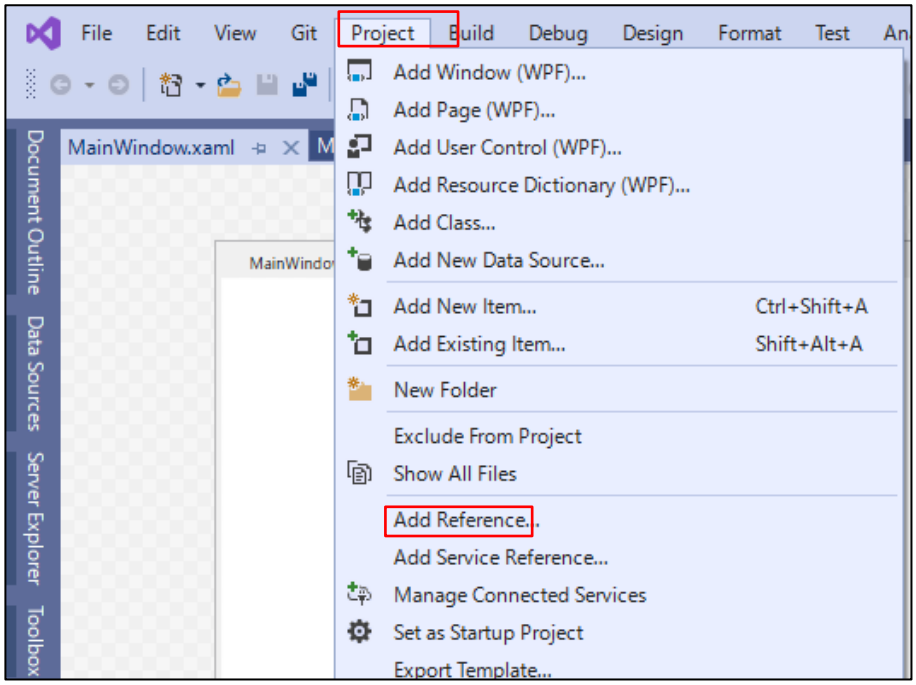


Figure 3-17 Displaying the reference manager screen

In RMWCommunicationLibrary Manager window, uncheck "Browse.dll" and click "OK".

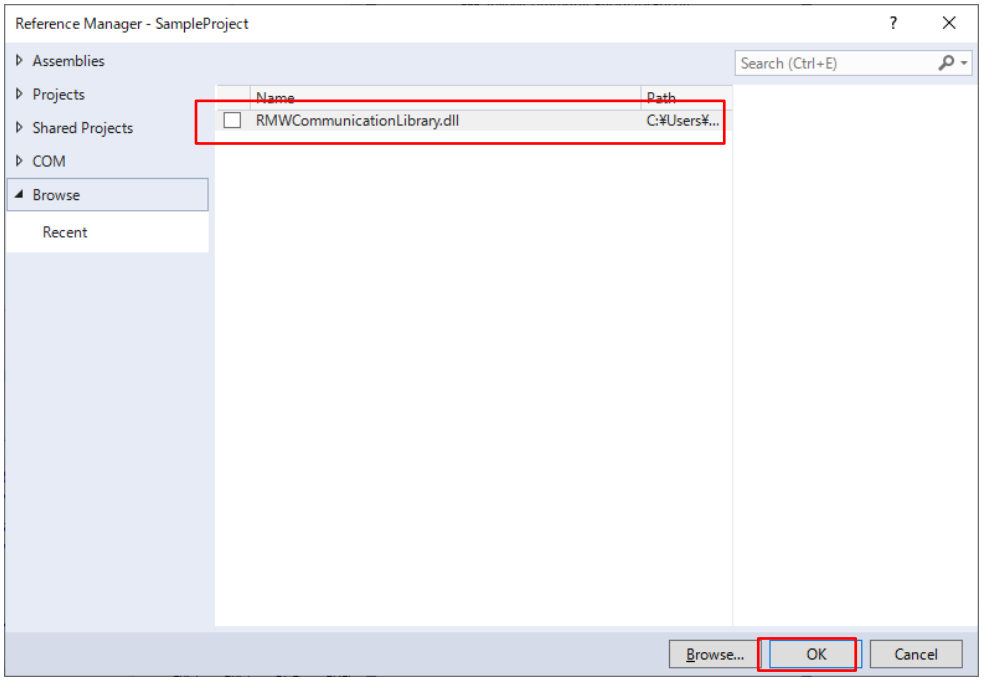


Figure 3-18 Operation of the manager screen

(2) Removing DLL from Project

Select View-Solution Explorer at the top of the screen.

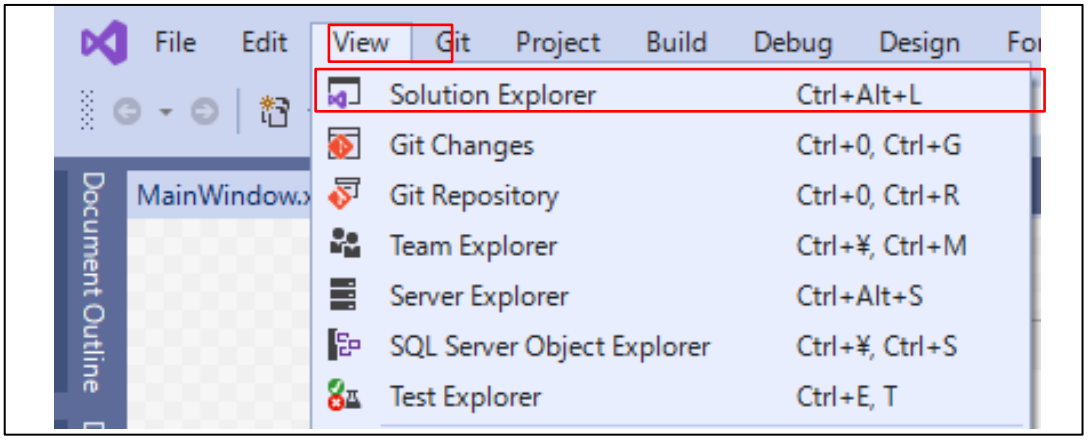


Figure 3-19 Viewing the Solution Explorer

In the Solution Explorer that appears, right-click the DLL file that you want to delete and select Delete.

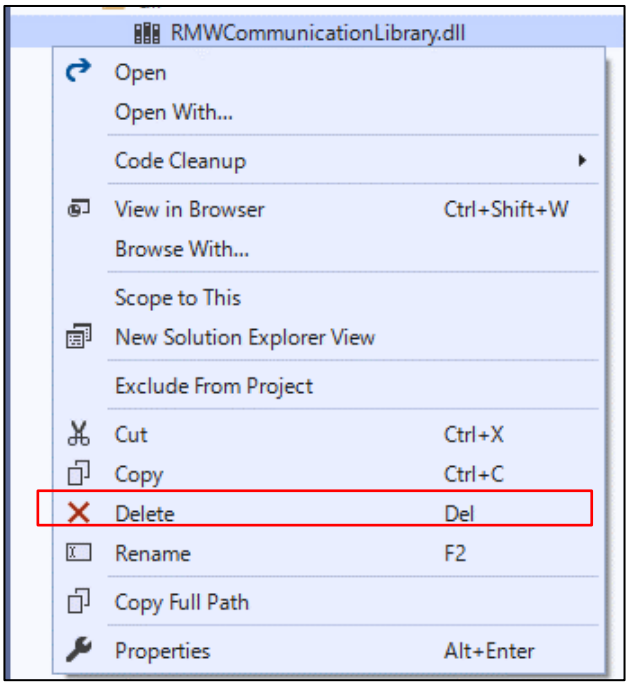


Figure 3-20 Removing a file from a project

When the following message is displayed, press the "OK" button.

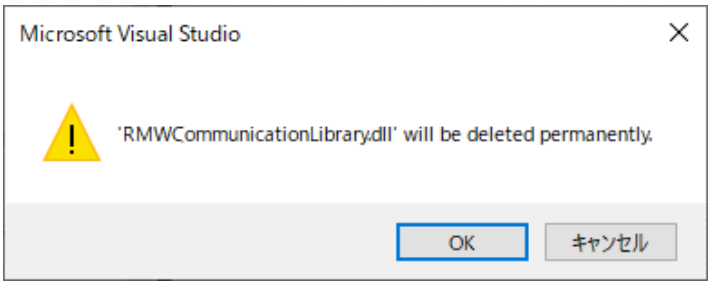


Figure 3-21 Precautions for deleting files

(3) Save the project

Select File tab and Save All pull down menu.

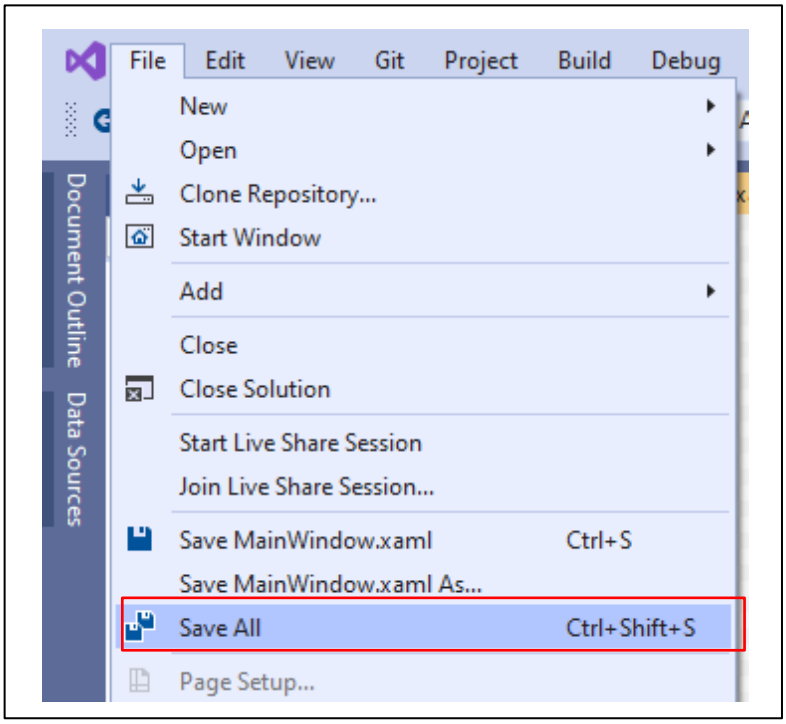


Figure 3-22 Save the project

3.2.2 How to remove from Excel

Describes how to remove DLLs from Excel.

(1) De-register DLLs from Excel reference settings

Select "Develop"-"Visual Basic" displayed on the ribbon of Excel.

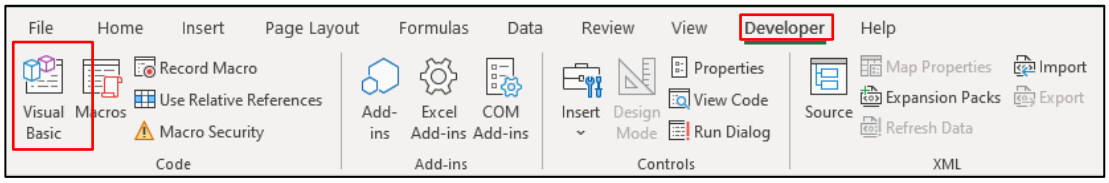


Figure 3-23 How to View Visual Basic Window

Select "Tools"-"Reference Setting" on the displayed screen.

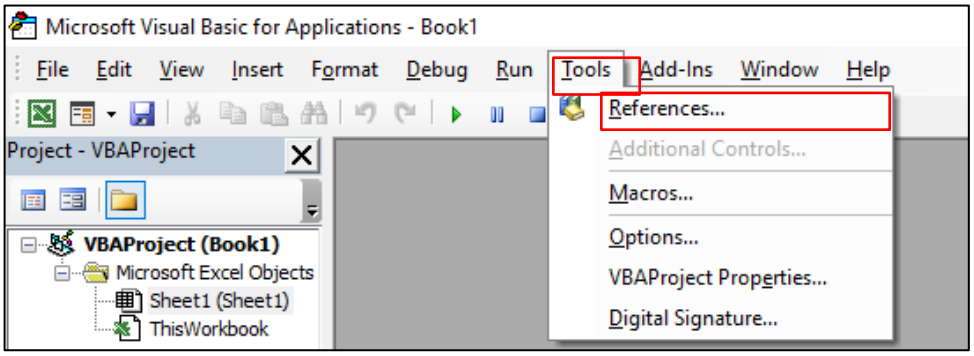


Figure 3-24 How to display the setting screen

In the displayed window, uncheck "RMW Communication Library" and click "OK".

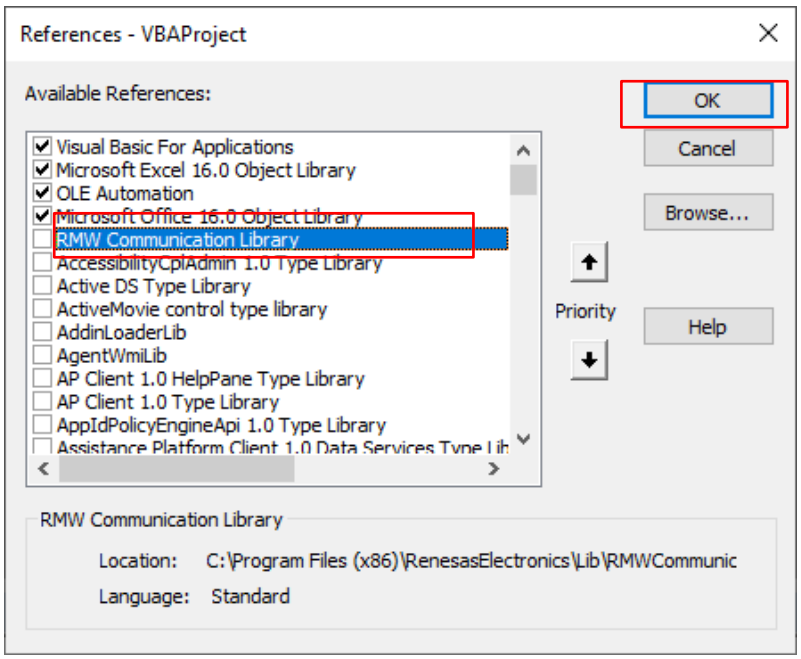


Figure 3-25 Unregistering RMW Communication Library

(2) Unregister a DLL from the Registry

Command prompts are stored in Windows System Tools.

Right-click the command prompt and select More-Run as Administrator.

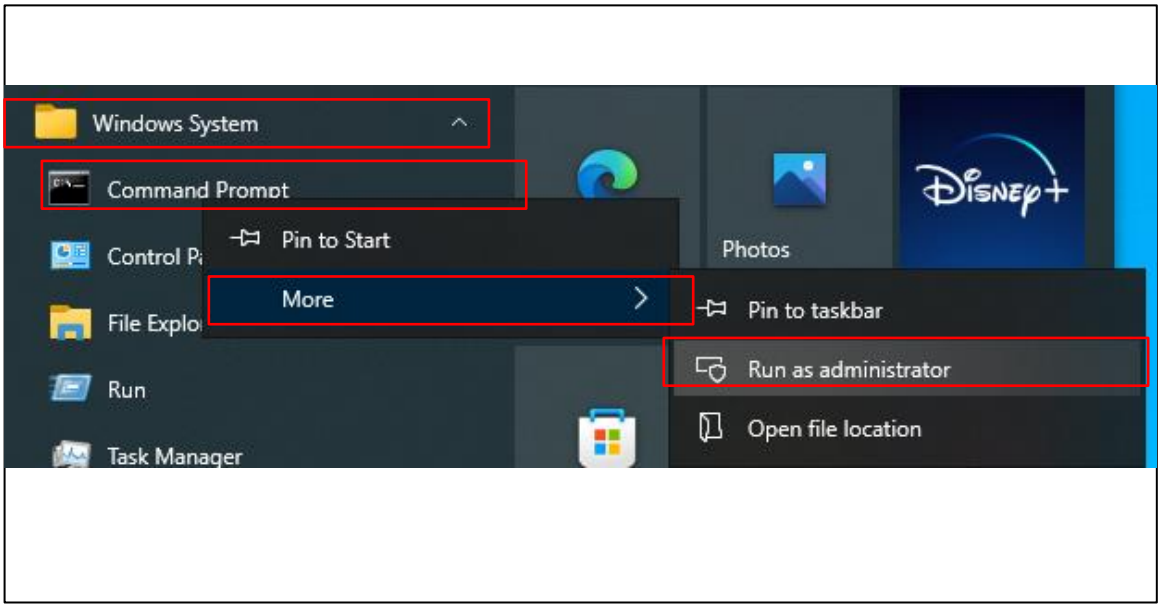


Figure 3-26 Starting the command prompt

From the command prompt, issue the following command:

[Command to be executed]

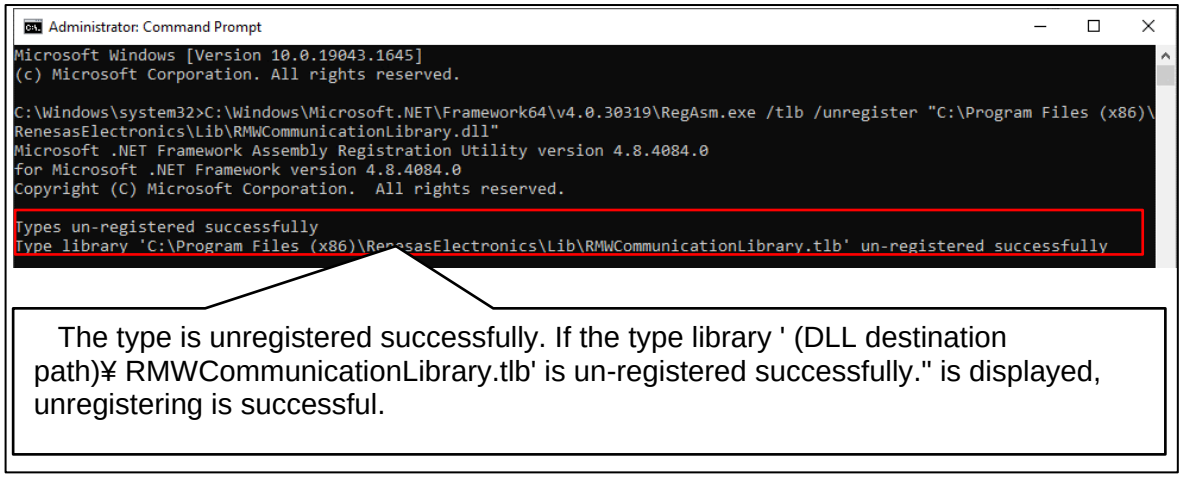
(RegAsm.exe storage path)¥ RegAsm.exe /unregister /codebase "(DLL expansion destination path)¥RMWCommunicationLibrary.dll"

RegAms.exe storage paths are as follows.

64bit Excel:C:¥Windows¥Microsoft.NET¥Framework64¥v4.0.30319

32bit Excel:C:¥Windows¥Microsoft.NET¥Framework¥v4.0.30319

Change RegAsm.exe to run with Excel version, not the OS type.



The type is unregistered successfully. If the type library ' (DLL destination path)¥ RMWCommunicationLibrary.tlb' is un-registered successfully." is displayed, unregistering is successful.

Figure 3-27 Command prompt execution screen

Deletes the DLL file that was extracted during deployment.

※In addition to dll, files have been generated, but because they are generated at the time of registration, delete them accordingly.

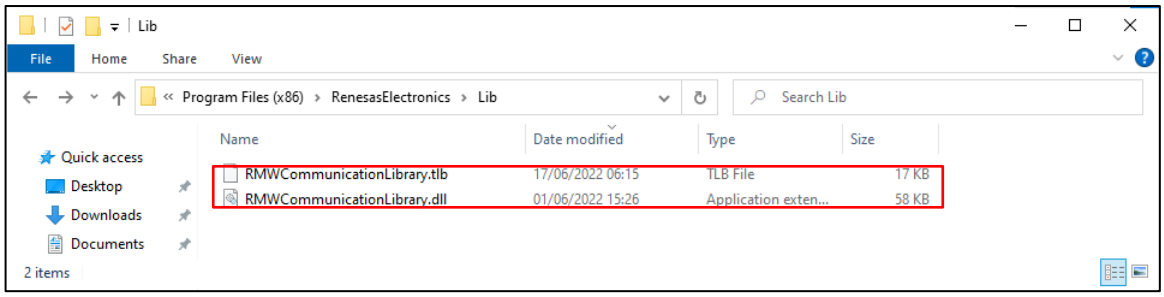


Figure 3-28 Deleting extracted files

4. List of DLL Functions

The following is a list of DLL functions.

Table 4-1 List of functions

Class Name	Function Name	Overview	Argument	Return value
ComCommunication	Connect	Connect to a microcomputer	1st argument: COM port to connect 2nd: Baud rate used for connection	Result of connection processing
	DisConnect	Disconnect from the microcomputer	-	Result of the cutting process
	Read	Reading values from the MCU	1st argument: Read result 2nd argument: Address to read 3rd: Data type to read 4th argument: Endian	Results of the read operation
	Write	Writing values to the MCU	1st argument: Address to write 2nd argument: Data type to write 3rd: Value to write 4th argument: Endian	Result of the write process
	ScopeStart	Begin Scope process	1st argument: Trigger channel 2nd argument: Trigger level 3rd parameter: Scope setting	Consequence of Scope treatment
	ScopeGetCondition	Get the status of Scope process	-	Status of Scope treatment
	ScopeGetData	Get Scope process	1st argument: Read index 2nd: Number of read data 3rd: Variable for storing acquired data 4th argument: Endian	Acquisition status of data
MapConversion	ConvertMapToMemory	Extract Map file conversion results to memory	1st argument: Map file path 2nd argument: Prefix for variable 3rd: Prefix for array 4th: Upper limit of data 5th: Number of data conversions 6th: Number of data to be stored 7th: Data storage variable 8th: Compiler type	Result of the conversion process
	ConvertMapToCSV	Output Map file conversion result to CSV file	1st argument: Map file path 2nd argument: Prefix for variable 3rd: Prefix for array 4th: Output file path 5th: Overwrite flag	Result of the conversion process
ScopeSetting	SetProcessInfo	Set the sampling time, positions, number of acquired data, etc. required for Scope processing.	1st Argument: Sample Time 2nd Argument: Position 3rd: Number of acquired data 4th argument: Trigger edge 5th: Trigger mode	Result of the setting process

	AddChannelInfo	Adding Channels Required for Scope Processing	1st argument: Address to read 2nd argument: Channel number 3rd argument: Data type 4th argument: Scale	Result of additional processing
	RemoveChannelInfo	Deletes the channel information specified by the argument.	1st argument: Channel number to delete	Result of the deletion process
	ClearChannelInfo	Delete all registered channel information	-	-

5. Functional description of each function

5.1 Connectivity (Connect)

This function uses the COM port and baud rate specified by the arguments and connects to the MCU.

If this function is executed again while it is already connected, the communication is disconnected and the connection processing is restarted.

Table 5-1 Arguments for connection functions

#	Description	Type	I/O	Effective value
1	COM port	String	I	COM-port number recognized on Windows
2	Baud rate	Int	I	1～999,999

After connecting, send the CPU information collection command, judge the returned response, and judge the success or failure of the connection.

If the connection process is successful, the success value returns.

If the connection process fails, the return value is returned as the value corresponding to the cause of the failure.

If an error occurs, check the return value and take the following actions.

Table 5-2 Connection function return values and their correspondence

#	Description	Type	Return value	Supported
1	Result of processing	Int	0:Successful processing	-
2			1: Incorrect COM port	Check the COM port setting and reconnect.
3			2:Baud rate incorrect	Check the baud rate setting and reconnect.
4			3:Connection failure	Check that the communication board for the tool is turned on, and then reconnect.
5			4:Command response is invalid	Reconnect after confirming that the microcomputer is compatible.

5.2 Disconnect function (DisConnect)

This function disconnects from the MCU.

If the disconnection process is successful, the success value returns.

If the disconnect operation fails, the return value is returned as the value corresponding to the cause of the failure.

If an error occurs, check the return value and take the following actions.

Table 5-3 Returns and correspondence between cut functions

#	Description	Type	Return value	Supported
1	Result of processing	Int	0:Successful processing	-
2			1:Not connected	Verify that the connection function is running. Check that the communication board for the tool is turned on. Check that the tool communication board is connected.

5.3 Reader (Read)

As shown in [Read image of each data type], this function reads the value from the MCU by the size corresponding to the variable type, using the address specified by the argument as the start address.

Table 5-4 Read function arguments

#	Description	Type	I/O	Effective value
1	Reading result	String	O	-
2	Address to read	Int	I	0x00000000 ～ 0x7FFFFFFF
3	Data Type to Read	Enum	I	ReadUInt8、ReadInt8ReadUInt16、ReadInt16、ReadUInt32、ReadInt32、ReadFloat、ReadBool、ReadLogic
4	Endian	Enum	I	Little、Big

※For the datatype and endian to be read, specify the values that correspond to the following enum definitions:

Table 5-5 Definition of data type to read

#	Enum	Value	Overview
1	ReadUInt8	0	Reading unsigned char Type Data
2	ReadInt8	1	Reading signed char Type Data
3	ReadUInt16	2	Reading unsigned short Type Data
4	ReadInt16	3	Reading signed short Type Data
5	ReadUInt32	4	Reading unsigned long Type Data
6	ReadInt32	5	Reading signed long Type Data
7	ReadFloat	6	Reading Float Type Data
8	ReadBool	7	Reading Bool Type Data
9	ReadLogic	8	Reading Logic Type Data

Table 5-6 Endian Type Definitions

#	Enum	Value	Overview
1	Little	0	Read in Little endianness
2	Big	1	Read in Big endian

Table 5-7 Supported Types and Sizes

#	Type	Size
1	bool	1
2	logic	1
3	signed char	1
4	unsigned char	1
5	signed short	2
6	unsigned short	2
7	signed long	4
8	unsigned long	4
9	float	4

[Read image of each data type]

○Read 8Bit (1byte)

When reading 8 bits (1byte) from address 1008

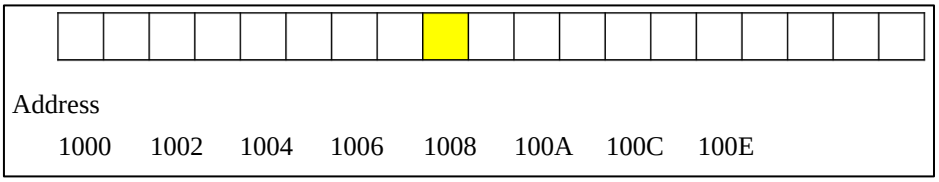


Figure 5-1 8Bit Read Image

○Read 16Bit (2byte)

When 16 bits (2byte) of data are read from address 1008

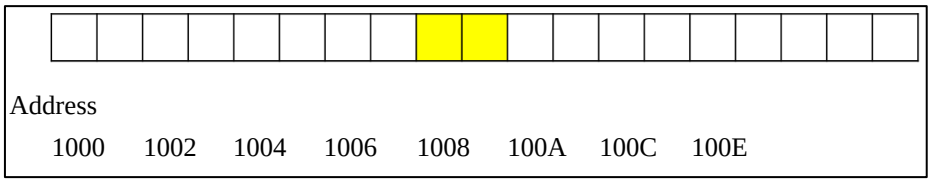


Figure 5-2 16Bit Load Image

○Read 32Bit (4byte)

When reading 32 bits (4byte) from address 1008

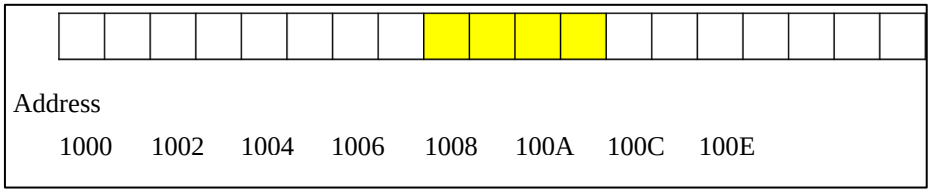


Figure 5-3 32Bit Load Image

The read data is converted with the specified endian as shown in the following figure.

○When Little Endian is specified

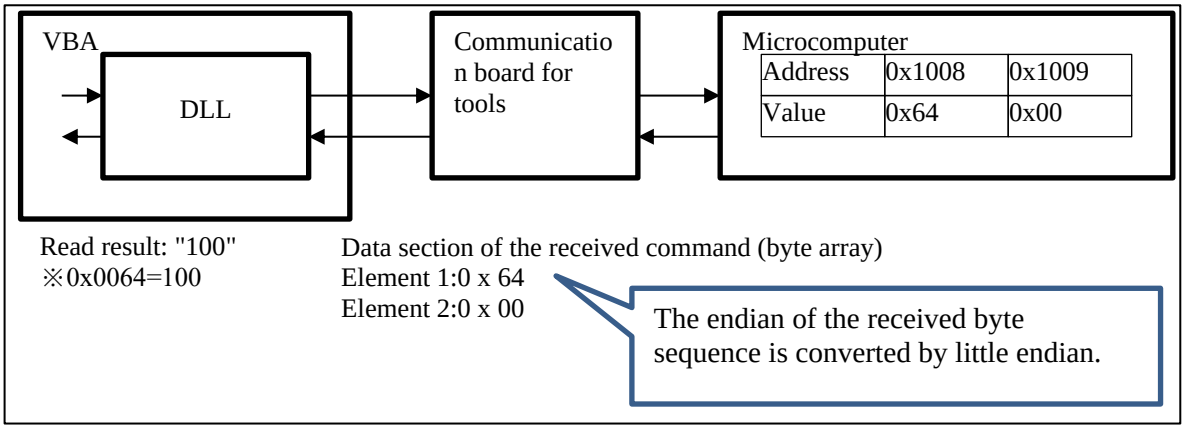


Figure 5-4 Little Endian Load Image

○When Big Endian is specified

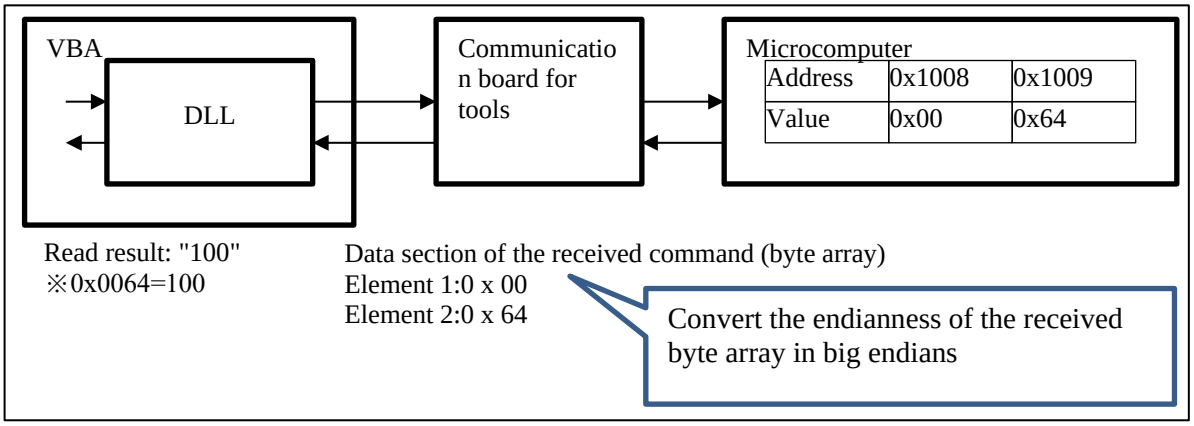


Figure 5-5 Big Endian Load Image

If the read operation is successful, the read result is stored in the argument, and the return value is returned as a successful operation.

If the read operation fails, the return value is returned as the value corresponding to the cause of the failure.

If an error occurs, check the return value and take the following actions.

Table 5-8 Returns and correspondence between read functions

#	Description	Type	Return value	Supported
1	Result of processing	Int	0:Successful processing	-
2			1:Not connected to the MCU	If the connection function is not executed, the read function is executed after the connection function is executed. If the connection function has already been executed, check the connection of the communication board for the tool, execute the connection function, and execute the read function.
3			2:Communication with the MCU failed.	If the connection function is not executed, the read function is executed after the connection function is executed. If the connection function has already been executed, check the connection of the communication board for the tool, execute the connection function, and execute the read function.
4			3:The value specified for the address is invalid.	Execute the read function again after checking the address setting.
5			4:Invalid data type	Execute the read function again after checking the data type setting.
6			5: Incorrect Endian	After confirming Endian setting, execute the read function again.

5.4 Write-function (Write)

This function writes the value specified by the argument to the address of the microcontroller specified by the argument, as in [Write image of each data type].

Table 5-9 Arguments for write functions

#	Description	Type	I/O	Effective value
1	Write address	Int	I	0x00000000 ～ 0x7FFFFFFF
2	Write value	String	I	(varies depending on the data type)
3	Write Data Type	Enum	I	WriteUInt8、WriteInt8、WriteUInt16、WriteInt16、WriteUInt32、WriteInt32、WriteFloat、WriteBool、WriteLogic
4	Endian	Enum	I	Little、Big

※Because the valid range of the value to be written varies depending on the data type, set it with the character string type.
For write datatypes and endianness, specify values that correspond to the following enum definitions:

Table 5-10 Write data type definitions

#	Enum	Value	Overview
1	WriteUInt8	0	Writes a unsigned char type
2	WriteInt8	1	Writes a signed char type
3	WriteUInt16	2	Writes a unsigned short type
4	WriteInt16	3	Writes a signed short type
5	WriteUInt32	4	Writes a unsigned long type
6	WriteInt32	5	Writes a signed long type
7	WriteFloat	6	Writes a Float type
8	WriteBool	7	Writes a Bool type
9	WriteLogic	8	Writes a Logic type

Table 5-11 Definition of endian type

#	Enum	Value	Overview
1	Little	0	Write in Little endian
2	Big	1	Write in Big Endian

Table 5-12 Supported Types and Sizes

#	Type	Size
1	bool	1
2	logic	1
3	signed char	1
4	unsigned char	1
5	signed short	2
6	unsigned short	2
7	signed long	4
8	unsigned long	4
9	float	4

[Write image of each data type]

○Write 8Bit (1byte)

To write 8-bit (1byte) data 0x01 to address 1004

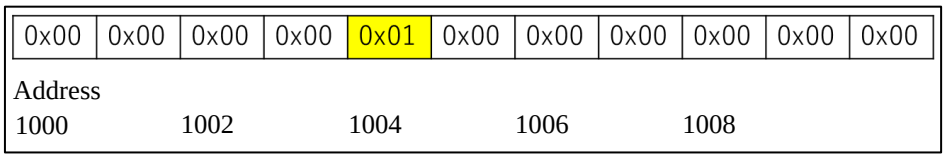


Figure 5-6 8Bit Write Image

○Write 16Bit (2byte)

To write 0x0001 of 16-bit (2byte) at address 1004

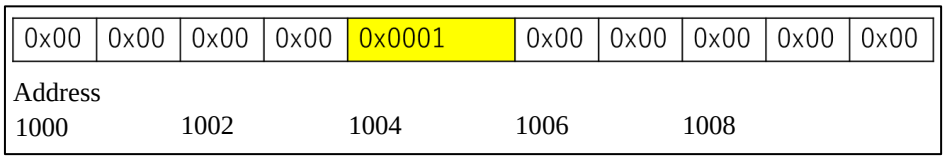


Figure 5-7 16Bit Write Image

○Write 32Bit (4byte)

To write data 0x0000 0001 of 32-bit (4byte) to address 1004

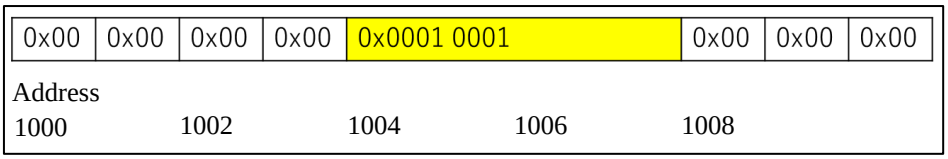


Figure 5-8 32Bit Write Image

The data to be written is converted to a byte array at the specified endian and transmitted.

○When Little Endian is specified

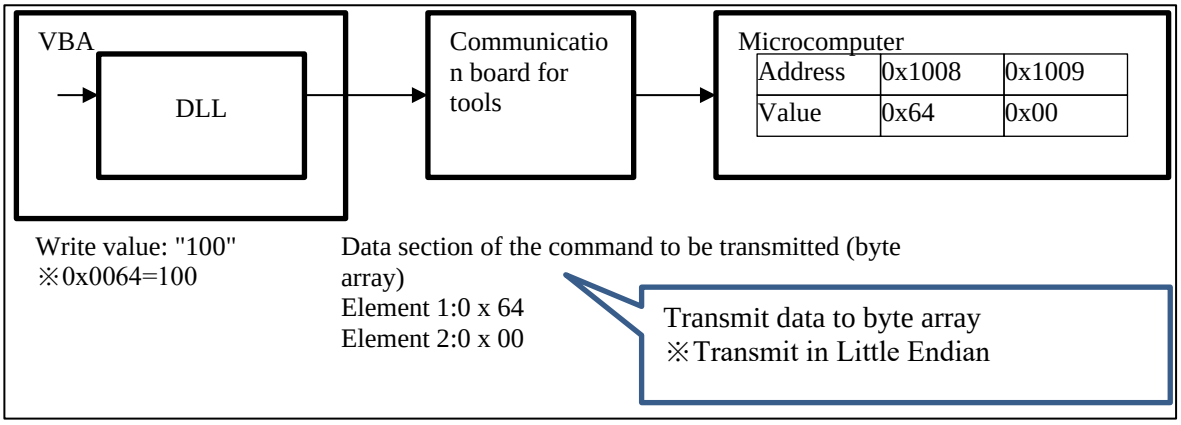


Figure 5-9 Little Endian Write Image

○When Big Endian is specified

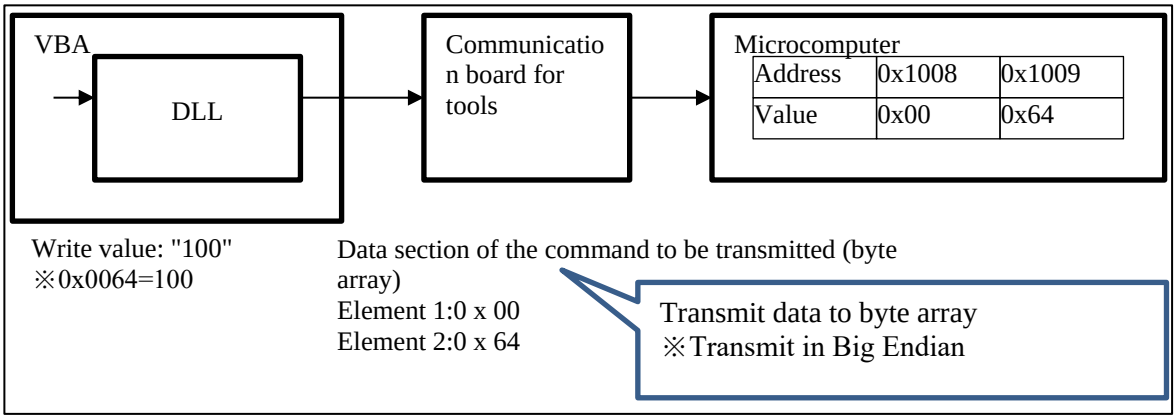


Figure 5-10 Big Endian Write Image

If the write operation is successful, the return value is returned as successful.

If the write operation fails, the return value is returned as the value corresponding to the cause of the failure.

If an error occurs, check the return value and take the following actions.

Table 5-13 Return Values and Correspondence of Write Functions

#	Description	Type	Return value	Supported
1	Result of processing	Int	0:Successful processing	-
2			1:Not connected to the MCU	If the connection function is not executed, the read function is executed after the connection function is executed. If the connection function has already been executed, check the connection of the communication board for the tool, execute the connection function, and execute the read function.
3			2:Address is invalid	Execute the write function again after checking the address setting.
4			3:Write value out of range	Execute the write function again after confirming the write value setting.
5			4:Communication failure with microcontroller	If the connection function is not executed, the read function is executed after the connection function is executed. If the connection function has already been executed, check the connection of the communication board for the tool, execute the connection function, and execute the read function.
6			5:Invalid data type	Execute the write function again after checking the data type setting.
7			6: Incorrect Endian	After confirming Endian setting, execute the write function again.

5.5 Scope Configuration (SetProcessInfo)

This function registers the data required for Scope process specified by the parameter.

If you call it more than once, it overwrites the last recalled setting.

Table 5-14 Parameters of Scope setting function

#	Description	Type	I/O	Value to be set	Effective value
1	Sampling time	Int	I	Sampling time (μs)	20～4000
2	Position	Double	I	Position to determine the trigger	0～1000
3	Number of retrieved records	Int	I	Number of records to retrieve	20 or more
4	Trigger edge	Enum	I	Rise、Down、Both	Rise、Down、Both
5	Trigger mode	Enum	I	Single、Normal、Auto	Single、Normal、Auto

※Refer to [About Trigger Edge] and [About Trigger Mode] for the trigger edge and trigger mode.

[Trigger edge]

When the trigger edge is "Rise (Rising detection)", if the trigger channel value exceeds the trigger level value, the trigger condition is satisfied, and the data of each channel is acquired.

When the trigger edge is "Down (Down detection)" and the value of the trigger channel falls below the trigger level value, the trigger condition is satisfied and the data of each channel is acquired.

When the trigger edge is "Both (Both detections)", if the trigger channel value exceeds or falls below the trigger level value, the trigger condition is satisfied, and the data of each channel is acquired.

Table 5-15 Trigger Edge Definitions

#	Enum	Value	Overview
1	Rise	0	Rising detection
2	Down	1	Down detection
3	Both	2	Both detections

[Trigger Mode]

When the trigger mode is "Single (1 time acquisition)" and the trigger condition is satisfied, data is acquired only once.

When the trigger mode is "Normal (Acquired every time the trigger condition is satisfied)", data is acquired each time the trigger condition is satisfied.

When the trigger mode is "Auto (always acquired)", data is always acquired.

Table 5-16 Trigger mode definitions

#	Enum	Value	Overview
1	Single	0	Acquired once
2	Normal	1	Acquired each time the trigger condition is satisfied
3	Auto	2	Always Acquired

If Scope setting process is successful, the return code is returned as successful.

If Scope setting process fails, the return value is returned as the value corresponding to the cause of the failure.

If an error occurs, check the return value and take the following actions.

Table 5-17 Return values of Scope setting functions and their corresponding values

#	Description	Type	Return value	Supported
1	Result of processing	Int	0:Successful processing	-
2			1:Trigger edge incorrect	After confirming the trigger edge setting, re-execute Scope setting function.
3			2:Trigger mode is invalid	After confirming the trigger mode setting, re-execute Scope setting function.

5.6 Add Scope Channels (AddChannelInfo)

This function adds the channel information specified by the argument to the channel information to be collected.

Table 5-18 Arguments for Scope channel addition function

#	Description	Type	I/O	Value to be set
1	Address information	Int	I	0x00000000 ～ 0x7FFFFFFF
2	Channel number	Byte	I	(varies depending on the microcomputer)
3	Data Type	Enum	I	UINT8、INT8、UINT16、INT16、UINT32、INT32、FLOAT、BOOL、LOGIC
4	Scale information	Byte	I	1～255

※For the types of data to retrieve, specify values that correspond to the following enum definitions:

Table 5-19 Data type definitions

#	enum	Value	Overview
1	UINT8	0	type of unsigned char
2	INT8	1	type of signed char
3	UINT16	2	type of unsigned short
4	INT16	3	type of signed short
5	UINT32	4	type of unsigned long
6	INT32	5	type of signed long
7	FLOAT	6	type of float
8	BOOL	7	type of bool
9	LOGIC	8	type of logic

If the process of adding Scope channels is successful, the return code is returned as successful.

If the process of adding Scope channels fails, the return value is returned as the value corresponding to the cause of the failure.

If an error occurs, check the return value and take the following actions.

Table 5-20 Return values of Scope channel addition functions and their corresponding values

#	Description	Type	Return value	Supported
1	Result of processing	Int	0:Successful processing	-
2			1:Data type is invalid	Execute the channel information addition function again after checking the data type setting.
3			2:Datatypes Not Supported by the Board	Execute the channel information addition function again after checking the data type setting.

5.7 Scope Channel Deletion Function (RemoveChannelInfo)

This function deletes the channel information of the channel number specified in the argument.

Table 5-21 Arguments for Scope channel deletion function

#	Description	Type	I/O	Value to be set
1	Channel number to delete	Byte	I	0 to 31 (already registered channel number)

If Scope channel deletion process is successful, the return code is returned as successful.

If Scope channel deletion process fails, the return value is returned as the value corresponding to the cause of the failure.

If an error occurs, check the return value and take the following actions.

Table 5-22 Returns and correspondence between Scope channel-deletion functions

#	Description	Type	Return value
1	Result of processing	Int	0:Successful processing
2			1:The specified channel does not exist

5.8 Scope channel-information-all-deletion-function (ClearChannelInfo)

This function deletes all channel information registered at the time of execution.

This function is executed without arguments and does not return a return value.

5.9 Understanding Scope Handling Functionality

Scope process uses ScopeStart functions, ScopeGetCondition functions, ScopeGetData functions, and ScopeStop functions.

Table 5-23 Functions used for Scope handling

#	Function Name	Overview
1	ScopeStart	Starting Scope process
2	ScopeGetCondition	Gets the status of Scope process
3	ScopeGetData	Retrieves the data acquired by Scope process.
4	ScopeStop	Scope process stopped

When you perform a Scope operation, you execute the functions in the following order:

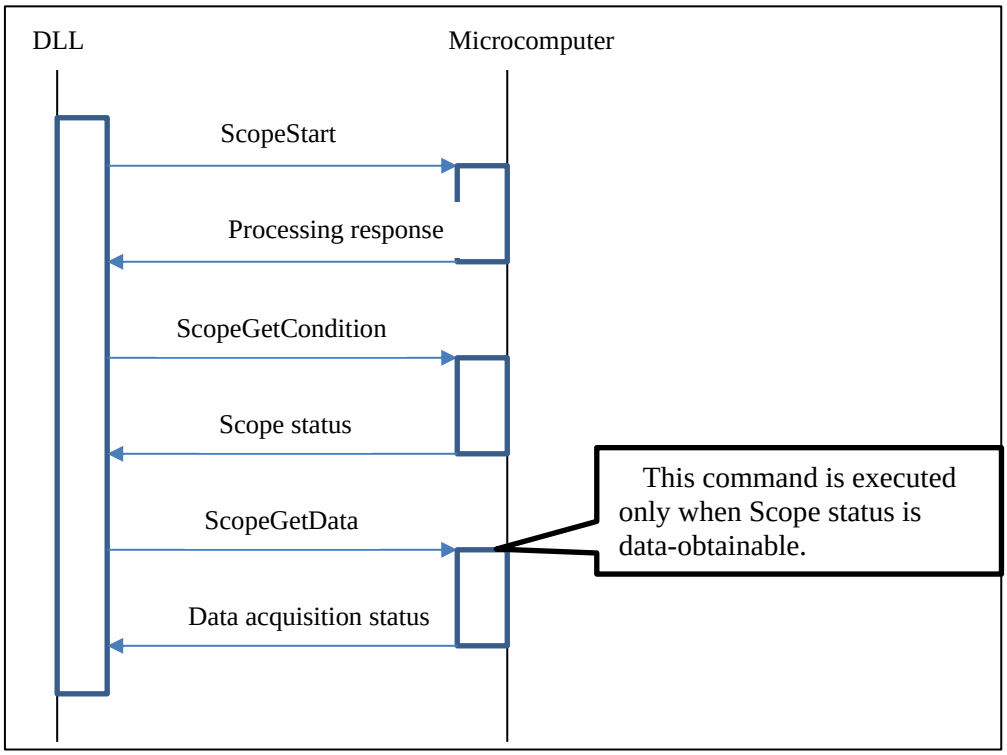


Figure 5-11 Image of Scope process flow

In the above flow, the value of the variable of the specified channel is read continuously from the MCU.

The received data is converted to the specified endian and returned as a read result, as shown in the following figure.

○When Little Endian is specified

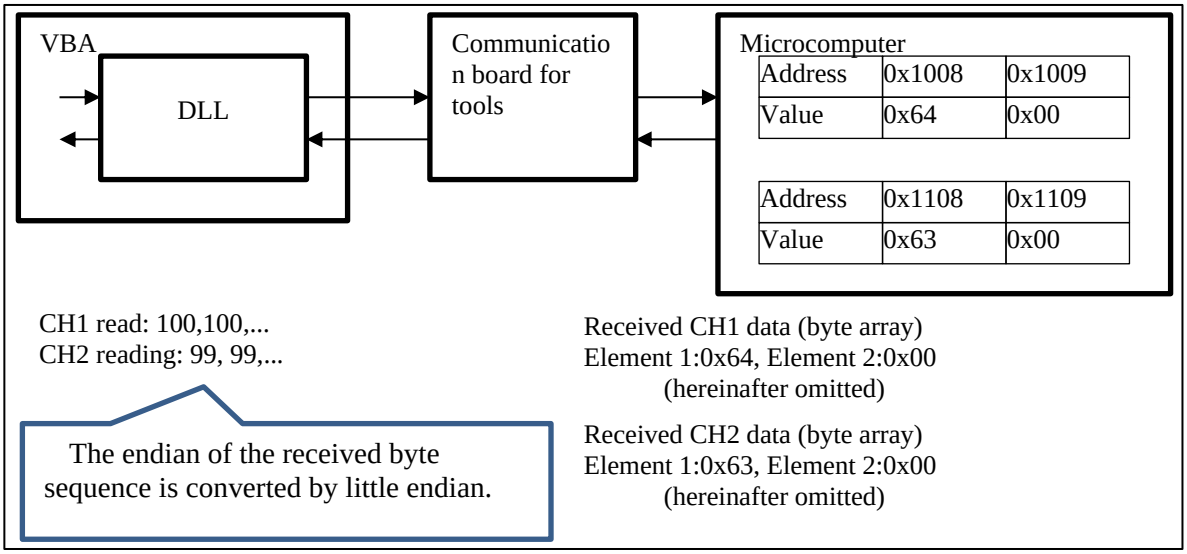


Figure 5-12 Little Endian Continuous Load Image

○When Big Endian is specified

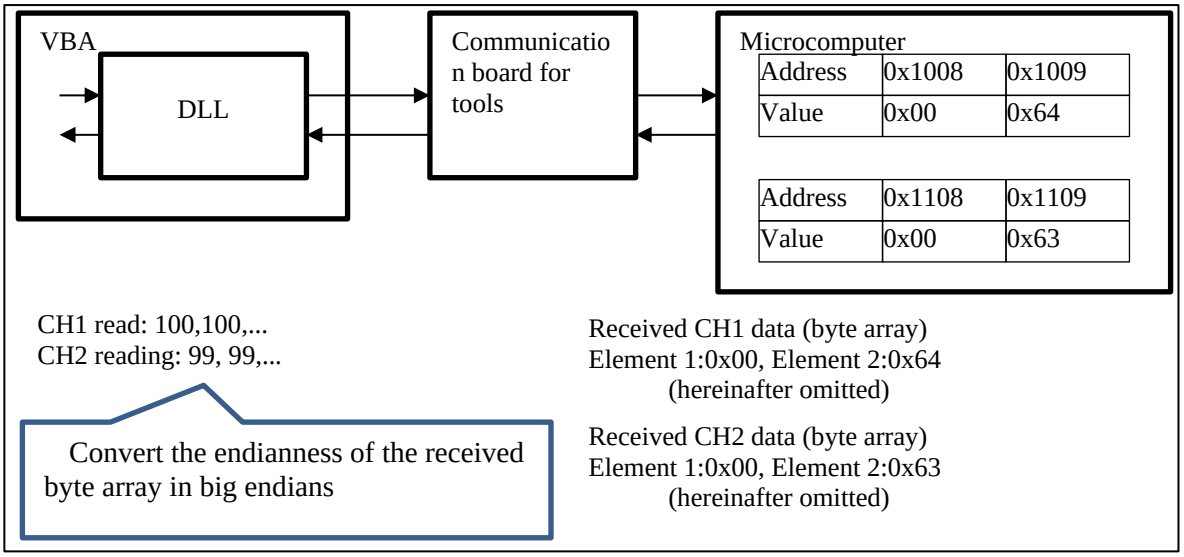


Figure 5-13 Big Endian Continuous Load Image

5.9.1 Scope Initiator (ScopeStart)

This function acquires the value of the variable set in the channel information using the setting of Scope specified in the argument.

Checks the parameter and starts Scope process if it is successful.

Table 5-24 Arguments for Scope starter

#	Description	Type	I/O	Effective value
1	Trigger channel	Byte	I	Maximum number of channels acquired during connection processing (varies depending on the MCU)
2	Trigger level	Double	I	-3.40282346638529E+38～3.40282346638529E+38
3	Configuring Scope Process	ScopeSetting	I	Set the data required for Scope processing such as sampling time, number of acquired data, and positions.

※For ScopeSetting, see [About ScopeSetting Classes].

[About ScopeSetting Classes]

ScopeSetting class registers the information required for Scope process and the settings related to the channels to be acquired.

Prior to executing ScopeStart function, use the following functions to set the information required for Scope process and the channels to be acquired.

Registering the information required for Scope process: See 5.5Scope Configuration (SetProcessInfo)

Registering channel information : See 5.6Add Scope Channels (AddChannelInfo)

Deleting the specified channel information: See 5.7Scope Channel Deletion Function (RemoveChannelInfo)

Deleting all channel information : See 5.8Scope channel-information-all-deletion-function (ClearChannelInfo)

If Scope startup process is successful, the return code is returned as successful.

If Scope startup process fails, the return value is returned as the value corresponding to the cause of the failure.

If an error occurs, check the return value and take the following actions.

Table 5-25 Returns and associations between Scope startup functions

#	Description	Type	Return value	Supported
1	Result of processing	Int	0:Successful processing	-
2			1:Scope setting incorrect	After confirming the scope setting, re-execute Scope start function.
3			2:Trigger level is invalid	After confirming the trigger level setting, re-execute Scope start function.
4			3:Invalid channel setting	After checking the channel setting, re-execute Scope start function.
5			4: Invalid Position setting	After confirming the position setting, execute Scope start function again.
6			5:Invalid record length setting	Check the number of records to be acquired, and then re-execute Scope start function.
7			6:Incorrect sampling period	After checking the sampling period, re-execute Scope start-function.
8			7:Not connected to the MCU	If the connection function is not executed, Scope start function is executed after the connection function is executed. If the connection function has already been executed, check the connection of the communication board for the tool, execute the connection function, and execute Scope start function.
9			8:Command communication failed	If the connection function is not executed, Scope start function is executed after the connection function is executed. If the connection function has already been executed, check the connection of the communication board for the tool, execute the connection function, and execute Scope start function.

5.9.2 Get Scope status (ScopeGetCondition)

This function obtains the status of Scope process started by Scope startfunction.

This function is executed only when Scope startfunction has already been executed.

This function is executed without arguments.

If Scope status acquisition process is successful and data retrieval is possible, the return value is returned as successful.

If Scope status acquisition process was successful, but the data cannot be acquired, the return value is returned while preparing to acquire the data.

If the process of acquiring Scope status fails, the return value is returned as the value corresponding to the cause of the failure.

If an error occurs, check the return value and take the following actions.

Table 5-26 Returns and correspondence between Scope status acquisition functions

#	Description	Type	Return value	Supported
1	Result of processing	Int	0:Successful processing	-
2			1:Preparing to acquire data	Execute the status acquisition function again.
3			2:Not connected to the MCU	If the connection function is not executed, execute the connection function and then re-execute from Scope start function. If the connection function has already been executed, check the connection of the communication board for the tool, execute the connection function, and execute Scope start function.
4			3:Command communication failed	If the connection function is not executed, Scope start function is executed after the connection function is executed. If the connection function has already been executed, check the connection of the communication board for the tool, execute the connection function, and execute Scope start function.

5.9.3 Scope Acquisition (ScopeGetData)

This function acquires the data buffered in the MCU collected during Scope process.

Execute this function only when Scope status acquisition function is processed successfully (when data acquisition is enabled).

This function acquires data from the MCU by specifying the start index of reading the MCU's buffer and the number of read data.

Table 5-27 Arguments for Scope data-retrieval function

#	Description	Type	I/O	Effective value
1	Read Index	Int	I	0 or more
2	Number of read data	Int	I	1 or more
3	Retrieved data storage variable	GetData[]	O	Structure that stores the acquired data
4	Endian	Enum	I	Little、Big

※For GetData, see [About GetData Structures].

For endianness, specify values that correspond to the following enum definitions:

Table 5-28 Endian Type Definitions

#	Enum	Value	Overview
1	Little	0	Write in Little endian
2	Big	1	Write in Big Endian

[About GetData struct]

GetData struct is used to set the acquired data, the number of acquired data, and the maximum number of data to be acquired.

Table 5-29 Variables defined in GetData struct

#	Variable Name	Type	I/O	Overview	Effective value
1	ScopeData	Double[]	I/O	Array containing the retrieved data	(The value is set within the function.)
2	DataCount	Int	O	Number of acquired data	(The value is set within the function.)
3	CountLimit	Int	I	Maximum number of data to retrieve	1 or more

Before executing ScopeGetData function, make the following settings for this structure.

(1)Initialize ScopeData.

(2)Set CountLimit to a value less than or equal to the number of elements of the array assigned in (1).

※ScopeGetData stores in an array the following values specified in CountLimit:

Be sure to set a value less than or equal to the number of elements in the set array.

Perform the above operation for each channel.

If Scope data acquisition process is successful and the maximum number of data items to be acquired has been acquired, the return value is returned as successful.

For data acquisition, the data received in the endian specified by the argument is converted and stored in the variable.

If Scope data acquisition process is successful, but data acquisition is not completed until the maximum number of data items to be acquired, the data acquisition function is executed again.

If the process of acquiring Scope data fails, the return value is returned as the value corresponding to the cause of the failure.

If an error occurs, check the return value and take the following actions.

Table 5-30 Returns of Scope data-acquisition functions and their corresponding values

#	Description	Type	Return value	Supported
1	Result of processing	Int	0:Processing succeeded (data acquisition completed)	-
2			1:Invalid read range setting	Check the values of the read index and the number of read data.
3			2:Invalid setting of the acquired data storage array (the maximum number of data to be acquired does not contain a value)	After confirming the setting of the acquired data storage variable, re-execute the data acquisition function.
4			3:The status is not data-retrievable.	Execute the status acquisition function and wait until data can be acquired.
5			4:Not connected to the MCU	If the connection function is not executed, execute the connection function and then re-execute from Scope start function. If the connection function has already been executed, check the connection of the communication board for the tool, execute the connection function, and execute Scope start function.
6			5:Endian setting is invalid	After checking the endian setting, re-execute the data acquisition function.
7			6: Scope setting is not set	Executed from Scope startfunction.
8			7:Command communication failed	If the connection function is not executed, Scope start function is executed after the connection function is executed. If the connection function has already been executed, check the connection of the communication board for the tool, execute the connection function, and execute Scope start function.

5.9.4 Scope shutdown (ScopeStop)

This function stops Scope process started by Scope startfunction.

Execute this function only at the timing to be stopped after executing Scope startfunction.

To execute Scope process again after executing this function, execute it from Scope startfunction.

This function is executed without arguments.

If Scope shutdown process is successful, the return code is returned as successful.

If Scope stopping process fails, the return value is returned as the value corresponding to the cause of the failure.

Table 5-31 Returns and correspondence between Scope termination functions

#	Description	Type	Return value	Supported
1	Result of processing	Int	0:Successful processing	-
2			1:Not connected to the MCU	If the connection function has not been executed, Scope start function is re-executed after the connection function is executed. If the connection function has already been executed, check the connection of the communication board for the tool, execute the connection function, and execute Scope start function.
3			2:Command communication failed	If the connection function has not been executed, Scope start function is re-executed after the connection function is executed. If the connection function has already been executed, check the connection of the communication board for the tool, execute the connection function, and execute Scope start function.

5.10 About the Map File Conversion Function

This function converts the specified Map file, expands it in memory, or outputs it to a CSV file.

Compilers supported are shown in the table below.

Table 5-32 Supported compilers

#	Compiler
1	CC
2	CA

When a process to expand data into memory or a process to output data to a CSV file, reading the map file and conversion to a CSV file are performed.

After the conversion to CSV format is completed, the file is expanded to memory or output to a CSV format file according to the executed function.

5.10.1 Map file conversion memory expansion function (ConvertMapToMemory)

This function converts the contents of the Map file at the path specified by the argument, and stores the conversion result in the variable specified by the argument.

The number of variables that have been analyzed and converted from the Map file is specified by the argument.

If there are more variables registered in the Map file than the maximum number of variables for storing conversion results, this function stores the values in the variables for storing conversion results as much as possible.

The number of variables stored in the conversion result storage variable is stored in the variable specified by the argument.

For the relationship between the data upper limit, conversion number, and storage number, see the following.

・When the number of conversion is less than the data upper limit

Like below figure case that the number of converted data is lower than the number of store size.

The conversion number and storage number are set to 3.

Figure 5-14 Relationship between data upper limit, conversion number, and storage number (data upper limit > conversion number)

・When the number of conversion is larger than the data upper limit

In the following cases, some data is not set in the conversion result storage variable because the upper limit of the data is less than the converted data.

The conversion number is set to 7 and the storage number is set to 5.

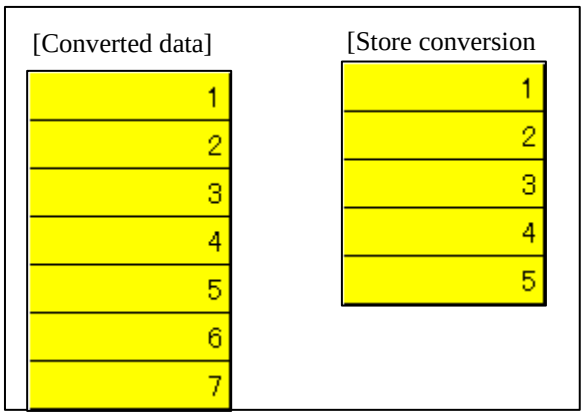


Figure 5-15 Relationship between data upper limit, conversion number, and storage number (data upper limit < conversion number)

Table 5-33 ConvertMapToMemory parameters

#	Variable Name	Type	I/O	Effective value
1	Map file path	String	I/O	Path of the actual existing Map file
2	Prefix for variable	VariablePrefixDefined	O	Class to set prefix information for each type
3	Prefix for array	ArrayPrefixDefined	I	Class to set prefix information for each type
4	Upper limit of data	Int	I	1 or more
5	Number of conversion	Int	O	(The value is set within the function.)
6	Number of storages	Int	O	(The value is set within the function.)
7	Conversion result storage	ConvData	O	(The value is set within the function.)
8	Compiler type	Byte	O	(The value is set within the function.)

※For VariablePrefixDefined, ArrayPrefixDefined, ConvData, see [About VariablePrefixDefined Classes], [About ArrayPrefixDefined Classes], and [About ConvData Classes].

[About VariablePrefixDefined Classes]

VariablePrefixDefined class is a class that sets a prefix for variable typing.

Table 5-34 VariablePrefixDefined Variables

#	Variable Name	Type	I/O	Overview	Effective value	Setting example
1	UINT8	String	I	Define prefix to be determined as UINT8	CSV format string	g_u1_,com_u1
2	INT8	String	I	Define prefix to be determined as INT8		g_s1_
3	UINT16	String	I	Define prefix to be determined as UINT16		g_u2_,com_u2_
4	INT16	String	I	Define prefix to be determined as INT16		g_s2_
5	UINT32	String	I	Define prefix to be determined as UINT32		g_u4_,com_u4
6	INT32	String	I	Define prefix to be determined as INT32		g_s4_
7	Float	String	I	Define prefix to be determined as Float		g_f4_

Prefix information is defined for each type.

To define multiple prefix information for a type, set the value in CSV format (separated by commas).

If no prefix is defined, each data is set to signed and the size depends on the data.

Table 5-35 Types by size

Size	Type
1	INT8
2	INT16
4	FLOAT

[About ArrayPrefixDefined Classes]

ArrayPrefixDefined class is a class that sets a prefix for array typing.

Table 5-36 ArrayPrefixDefined Variables

#	Variable Name	Type	I/O	Overview	Effective value	Setting example
1	UINT8	String	I	Define prefix to be determined as UINT8	CSV format string	g_u1_,com_u1
2	INT8	String	I	Define prefix to be determined as INT8		g_s1_
3	UINT16	String	I	Define prefix to be determined as UINT16		g_u2_,com_u2_
4	INT16	String	I	Define prefix to be determined as INT16		g_s2_
5	UINT32	String	I	Define prefix to be determined as UINT32		g_u4_,com_u4
6	INT32	String	I	Define prefix to be determined as INT32		g_s4_
7	Float	String	I	Define prefix to be determined as Float		g_f4_

Prefix information is defined for each type.

To define multiple prefix information for a type, set the value in CSV format (separated by commas).

If no prefix is defined, each data is set to signed and the size depends on the data.

Table 5-37 Types per size

Size	Type
1	INT8
2	INT16
4	FLOAT

[ConvData Class]

ConvData class is the class that stores the converted data.

Table 5-38 ConvData Variables

#	Variable Name	Type	I/O	Overview	Effective value
1	AddressName	String	O	Stores the address of a variable as an 8-digit hexadecimal number	(The value is set within the function.)
2	VariableName	String	O	Store variable names	(The value is set within the function.)
3	DataType	Byte	O	Stores numeric values of data types 0 to 6	(The value is set within the function.)

Table 5-39 Data Type Values

Value	Description
0	UINT8
1	INT8
2	UINT16
3	INT16
4	UINT32
5	INT32
6	FLOAT

If conversion of the Map file and storage in the variable is successful, the return value is returned as a successful processing.

If conversion of the Map file or storage processing to a variable fails, the return value is returned as the value corresponding to the cause of the failure.

If an error occurs during conversion of the Map file, processing is terminated. Therefore, the conversion result up to the occurrence of the error is not stored in the variable for storing the conversion result specified by the argument.

If an error occurs, check the return value and take the following actions.

Table 5-40 Returns and their corresponding ConvertMapToMemory

#	Description	Type	Return value	Supported
1	Result of processing	Int	0:Successful processing	-
2			1: Map file does not exist	Check the path of the Map file of the argument.
3			2: Map file cannot be read	Check that the Map file specified in the argument is not opened from other devices or read-only.
4			3: The Map file is output from an unsupported compiler.	Check that the Map file specified in the argument is output from the corresponding compiler.
5			4: Invalid variable setting in Map file	Verify that the Map file specified in the argument is correct.
6			5: Addresses in a Map File Cannot Be Converted to Hexadecimal	Verify that the Map file specified in the argument is correct.
7			6: Size in Map file cannot be converted to hexadecimal	Verify that the Map file specified in the argument is correct.
8			7: Map file conversion error	Verify that the Map file specified in the argument is correct.
9			8:Upper limit disabled	Check the setting of the upper limit specified by the argument.
10			9:Invalid storage array definition	Check the definition of the storage array specified by the argument.

5.10.2 Map file conversion CSV output function (ConvertMapToCSV)

This function converts the contents of the Map file specified by the argument, and saves the file in CSV format (separated by commas) converted to the file path specified by the argument.

Table 5-41 Arguments for ConvertMapToCsv

#	Variable Name	Type	I/O	Effective value
1	Map file path	String	I	Path of the actual existing Map file
2	Prefix for variable	VariablePrefixDefined	I	Class to set prefix information for each type
3	Prefix for array	ArrayPrefixDefined	I	Class to set prefix information for each type
4	Output path	String	I	File path that does not contain non-breaking characters
5	Overwrite flag	Bool	I	True: Overwrite enabled False: Overwrite prohibited

※For details about VariablePrefixDefined, ArrayPrefixDefined struct, see 5.10 Map File Conversion Memory Expansion Function (ConvertMapToMemory).5.10About the Map File Conversion Function

This function converts the specified Map file, expands it in memory, or outputs it to a CSV file.

Compilers supported are shown in the table below.

Table 5-32 Supported compilers

#	Compiler
1	CC
2	CA

When a process to expand data into memory or a process to output data to a CSV file, reading the map file and conversion to a CSV file are performed.

After the conversion to CSV format is completed, the file is expanded to memory or output to a CSV format file according to the executed function.

Map file conversion memory expansion function (ConvertMapToMemory)

Table 5-42 Non-breaking characters in file paths

#	Unusable characters	Remarks
1	¥	Not available for file and folder names. Can be entered as a path delimiter.
2	/	-
3	:	Not available for file and folder names. Can be entered as a path delimiter.
4	*	-
5	?	-
6	"	-
7	<>	-
8	|	-

If conversion of the Map file and output processing of the CSV file are successful, the return value is returned as a result of successful processing.

If the file has already been written to the output path, the file is overwritten if the overwrite flag is true, and an error is returned if the overwrite flag is false.

CSV is output in the following format.

Table 5-43 CSV Format

No	Item Name	Output example	Description
1	Compiler-specific string	Compiler:0	Outputs 0 for CC and 3 for CA at the beginning of the file.
2	Variable address	00000401	Output the address of a variable as an 8-digit hexadecimal number
3	Variable Name	g_u1_motor_status	Output variable name
4	Data type	0	Output numeric value representing data type

Table 5-44 Data Type Values

Value	Description
0	UINT8
1	INT8
2	UINT16
3	INT16
4	UINT32
5	INT32
6	FLOAT

An example of CSV output is shown below.

```
Compiler Type：0
Address,Variable Name,Data Type
00000401,g_u1_motor_status,0
00000402,gui_u1_active_gui,1
00000403,com_u1_sw_userif,0
00000404,g_u1_sw_userif,0
00000405,com_u1_mode_system,0
00000406,g_u1_mode_system,0
00000408,com_u1_direction,0
00000409,com_u1_enable_write,0
0000040a,g_u1_enable_write,0
00000432,g_u2_conf_sw_ver,2
00000434,g_u2_max_speed_rpm,2
00000436,com_s2_ref_speed_rpm,3
00000438,com_u2_max_speed_rpm,2
0000043a,com_u2_overspeed_limit_rpm,2
0000043c,com_u2_offset_calc_time,2
0000043e,com_u2_mtr_pp,2
00000440,com_u2_id_up_speed_rpm,2
00000442,com_u2_id_down_speed_rpm,2
00000456,g_u2_conf_hw,2
00000458,g_u2_conf_sw,2
```

If conversion of a Map file or output to a CSV file fails, the return value is returned as the value corresponding to the cause of the failure.

If the conversion of the Map file fails, the file is not output and terminated.

If an error occurs, check the return value and take the following actions.

Table 5-45 Returns and correspondence between ConvertMapToCSV functions

#	Description	Type	Return value	Supported
1	Result of processing	Int	0:Successful processing	-
2			1: Map file does not exist	Check the path of the Map file of the argument.
3			2: Map file cannot be read	Check that the Map file specified in the argument is not opened from other devices or read-only.
4			3: The Map file is output from an unsupported compiler.	Check that the Map file specified in the argument is output from the corresponding compiler.
5			4: Invalid variable setting in Map file	Verify that the Map file specified in the argument is correct.
6			5: Addresses in a Map File Cannot Be Converted to Hexadecimal	Verify that the Map file specified in the argument is correct.
7			6: Size in Map file cannot be converted to hexadecimal	Verify that the Map file specified in the argument is correct.
8			7: Map file conversion error	Verify that the Map file specified in the argument is correct.
9			8:Output path not set	Sets the value to the output path specified by the argument.
10			9:The output path contains invalid characters	Verify that nonbreaking characters are not used in the output path specified by the argument.
11			10:A path that cannot be output by the output path is specified.	Changes the output path specified in the argument to another path.
12			11:There is already a file. ※Occurs when the overwrite flag is false and a file already exists in the destination.	Allow the file to be overwritten or change the output destination.
13			12:File output failure	Check whether the file specified as the output destination is a folder that can be output or is not being used by other functions when overwriting.

6. How to use each function

This section describes how to use each function of this DLL.

6.1 Preparation before using DLL

6.1.1 Connecting Each Device

Connect the PC and the communication board for the tool with the USB cable, and connect the communication board for the tool, the microcomputer, and the motor with the dedicated cable.

For details on how to connect to the communication board for tools, see the manual for the communication board for tools.

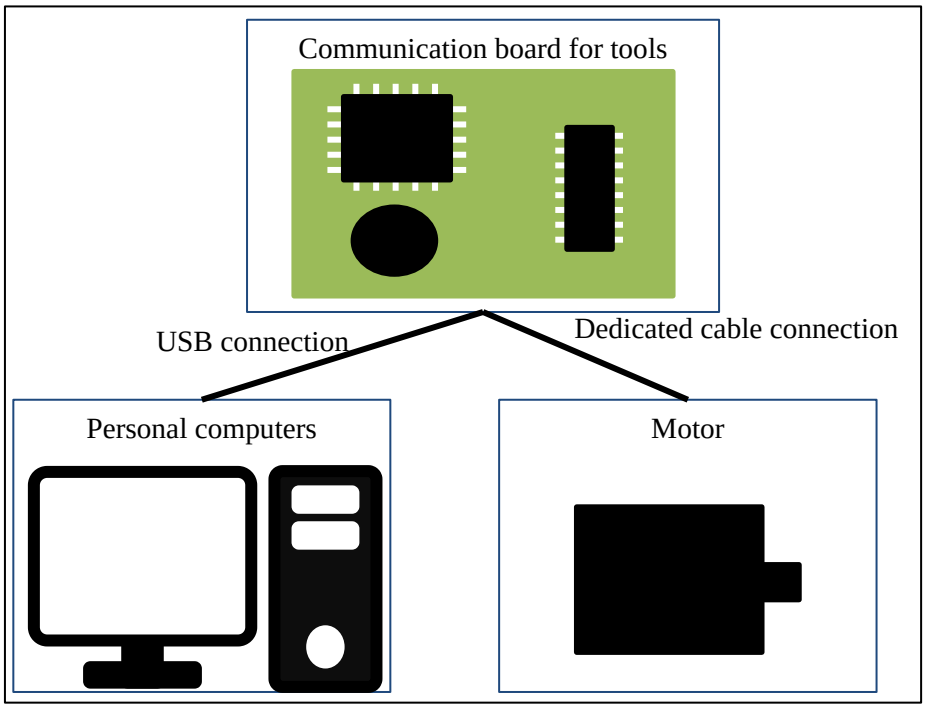


Figure 6-1 Connecting the Communication Board for Tools

6.1.2 Checking the COM Port

To execute the DLL connection function, you need to know which COM port to set as an argument.

Follow the procedure below to check the COM port.

Right-click the Start menu and select Device Manager.

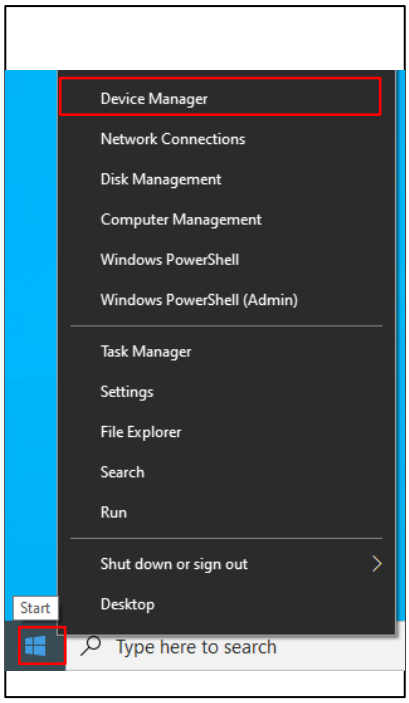


Figure 6-2 Starting the Device Manager

Double-click "Port (COM and LPT)" to display the port list.

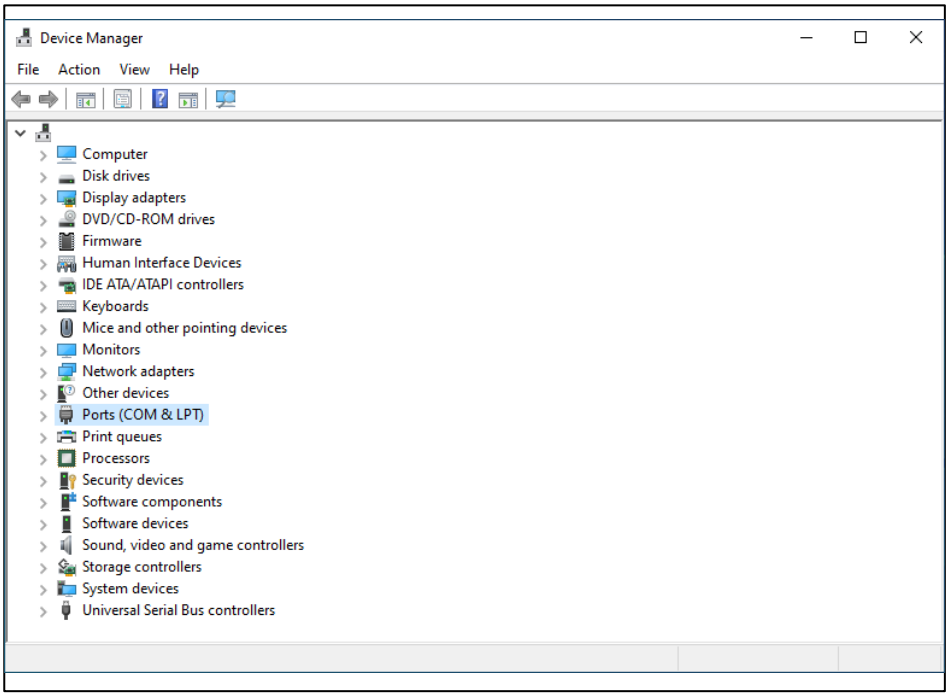


Figure 6-3 Selection in Device Manager Screen

The part of "USB serial device (COM~)" enclosed in parentheses is the COM port number. ("COM3" for picture illustration).

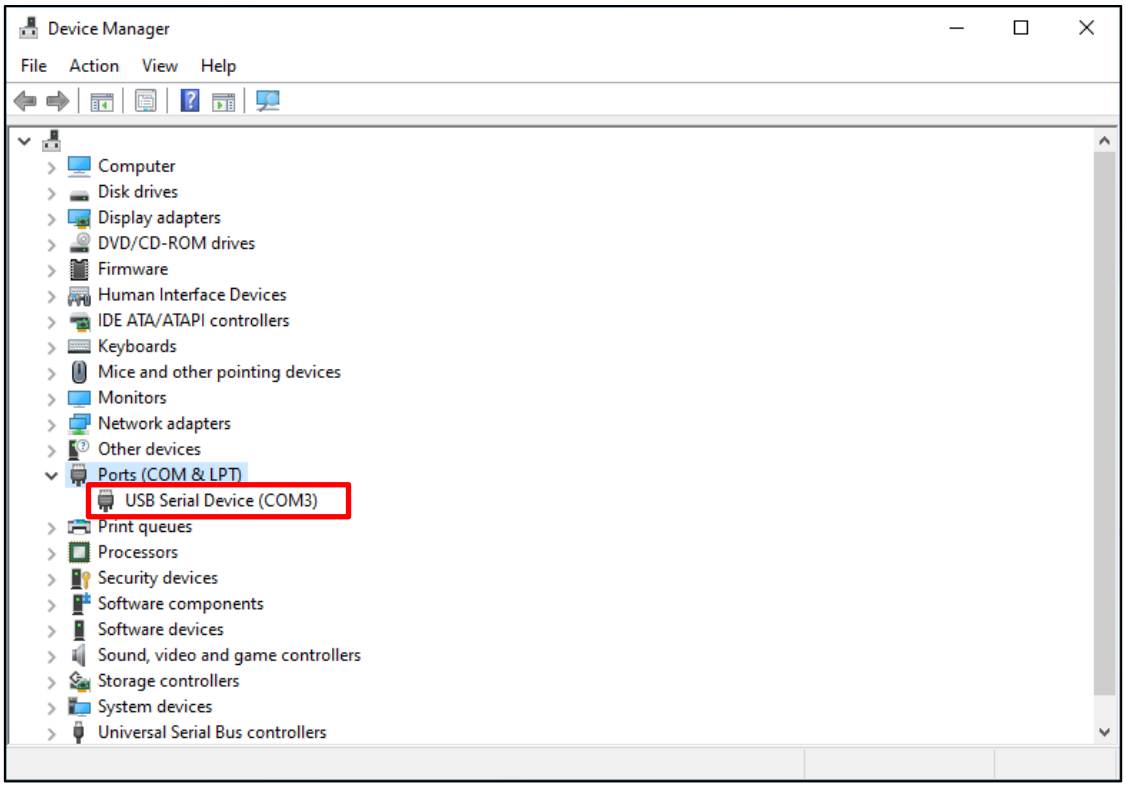


Figure 6-4 Checking the COM port using the Device Manager

6.2 Precautions when using DLL in Excel VBA

(1) Using Integer type

When using a function with an int-type argument from Excel VBA, specify the value to be set as the argument in Long type, not in Integer type.

(2) Using enum type

Enum type defined by the DLL cannot be referenced from Excel VBA.

When using a function that contains arguments of enum type from Excel VBA, specify the value to be set in Long type.

(3) Using DoEvents

When DoEvents is used in Excel VBA, control is returned to the operating system and the pending events are processed (operations on Excel, etc.).

If you want to perform a time-consuming process such as a loop as shown in the following example, use DoEvents function during the process.

(Example)

The process of getting data by looping ScopeGetData until the number of acquired data reaches any number

(HOW TO USE)

After retrieving ScopeGetData function's data, use DoEvents function to handle events from other operations prior to entering the next loop.

```
' Loop until the data acquisition state is completed.
Do
    ' Executes processes stored in the OS every 100 loops.
    ' Prevents DoEvents from running in large numbers.
    If count > 100 Then

        ' Return control to the OS and let it handle pending Excel operations and other events.
        DoEvents
        count = 0
    Else
        count = count + 1
    End If

    ' Execute the ScopeGetCondition function of the RMW communication library.
    condition = comLib.ScopeGetCondition()
Loop While condition = 1

' If the status of Scope is processing completion (data acquisition ready), data acquisition is performed.
If condition = 0 Then

    ' Execute the ScopeGetData function of the RMW communication library.
    getDataResult = comLib.ScopeGetData(100, 100, getDatas, 0)
End If
```

Figure 6-5 DoEvents

7. Using the Sample Program

This section describes the sample program for using this DLL.

7.1 Overview

This sample program executes each function of this DLL from VBA, and displays the processing result of each function on Excel.

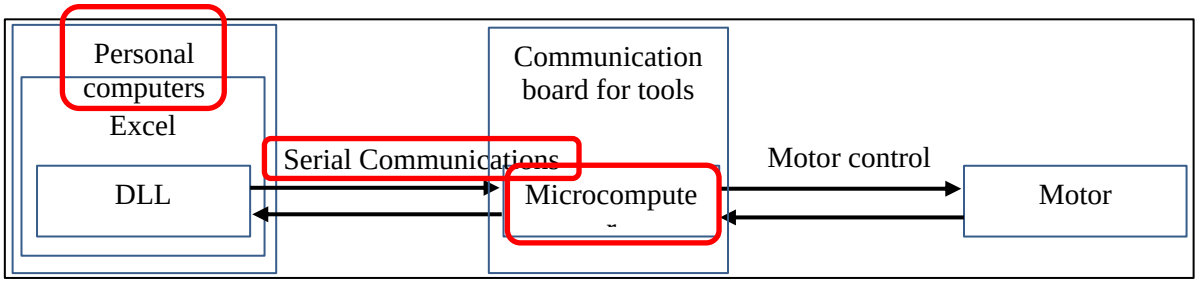


Figure 7-1 Software configuration diagram of the sample program

7.2 Operating Environment

The following environment is required for the operation of this sample program.

Table 7-1 System requirements

#	Item Name	Value
1	OS	Windows 10 only
2	.Net Framework	▪ Net Framework 4 6.1 or higher
3	Excel file format	Excel Macro Valid Book (*. xlsm)

7.3 Quick Start

7.3.1 Preparing Sample Programs

DLLs must be registered when using this DLLs in Excel. For information on registering DLLs, see How to Deploy to Excel

To use this example program, the macro of Excel must be enabled.

7.3.2 Sample Program Operation Procedure

This section describes the procedure from connecting to the microcontroller to reading variable values.

(1) Confirm that the microcontroller to be connected is ready for communication

(2) Enter the COM port number to connect to the communication board(Refer to 6.1.2 Checking the COM Port for COM port number confirmation.)

(a)Enter COM port number

(b)Click the Connect button

(c)Success is displayed on success

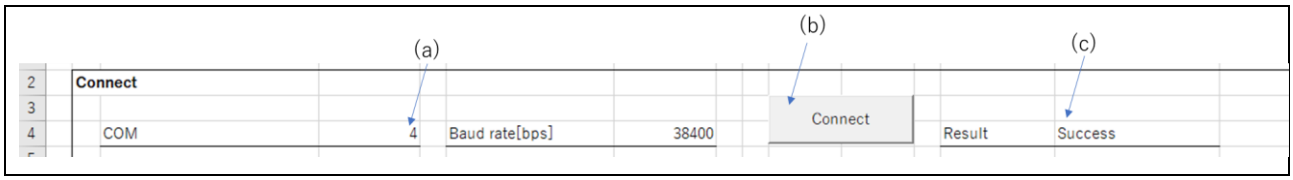


Figure 7-2 connect to the communication board

(3) Read the map file of the program written in the microcontroller

(a)Select the map file

(b)Click the MapToMem button

(c)Success is displayed on success

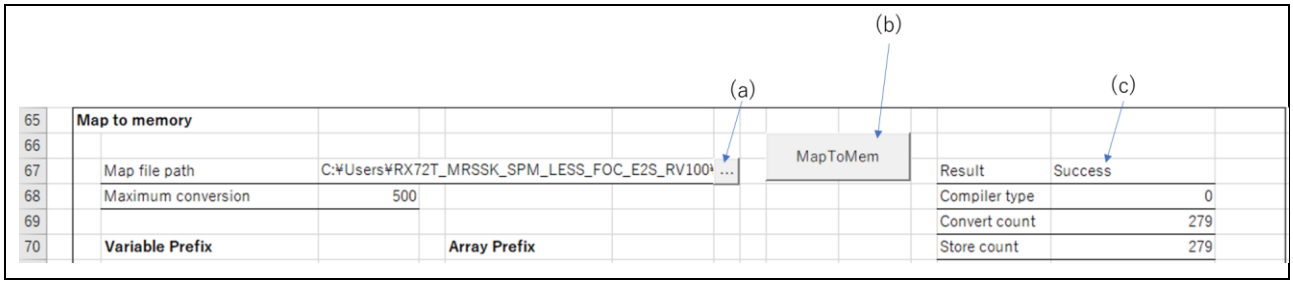


Figure 7-3 Read the map file

(4) Select variables and read the values of the selected variables

(a)Select the variable to read

(b)Click the Read button

(c)Success is displayed on success

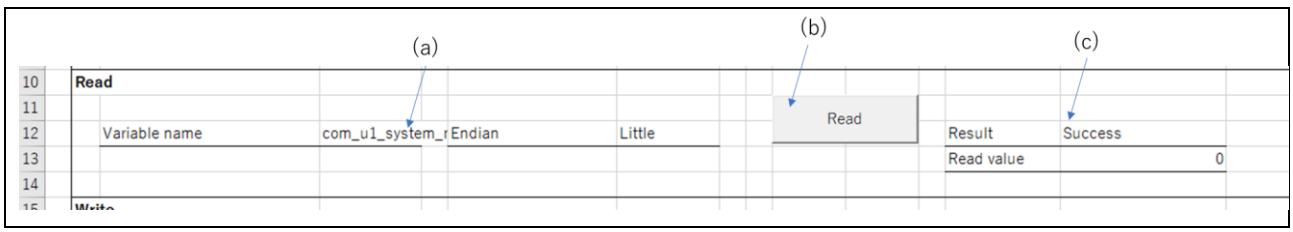


Figure 7-4 Read the values

7.4 OPERATION EXPLANATION

7.4.1 Sample seat

Sample sheet is the main sheet where the buttons for executing the functions of this sample program are arranged.

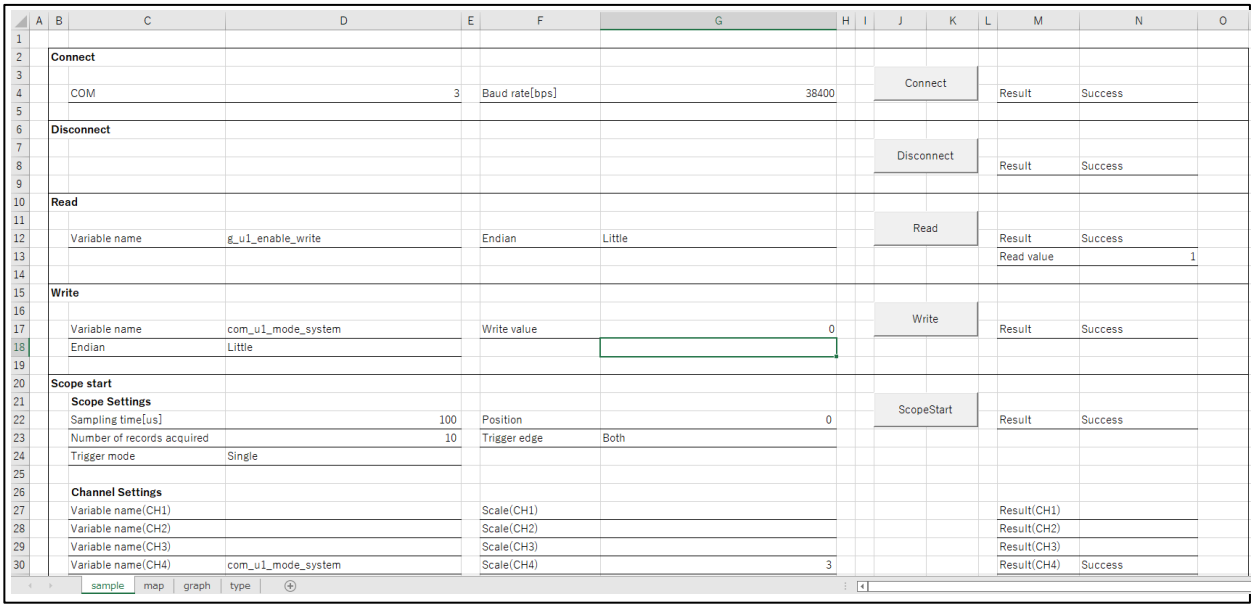


Figure 7-5 sample seat (whole)

Each function is delimited by creases, with input on the left side of each function button and output cells on the right side.

Figure 7-6 sample seat (Raad function)

If the input information is incorrect, the display will look like the following (cells with incorrect information will turn red and an error message will be displayed).

Figure 7-7 Incorrect input information

Functions that read/write microcontroller data, such as the Read and Write functions, require map file information for variable input and address conversion in each cell.

Please use each function after reading the map file information by "(10) Map to memory function".

When Excel is closed, Scope processing is stopped and the microcontroller is disconnected by the Sheet_Finalization function. If you cancel the close operation, please start the connection with the microcontroller again.

(1) Connect function

Enter the required data in the cells shown in the table below and press Connect to execute ConnectButton_Click function. The MCU is connected to the MCU.

Table 7-2 Connect Function Entries

#	Input cell	Input contents
1	COM	COM port number to connect
2	Baud rare[bps]	Baud rate value

Connect of the DLLs are processed in the cells shown in the following tables.

Table 7-3 Connect Function Details

#	Output cell	Output contents
1	Result	Success: Successful process IllegalCOMPort: COM-port incorrect IllegalBaudrate: Baud rate incorrect ConnectFailed: Failed to connect SendCommendFailed: Command response is invalid

(2) DisConnect function

Press Disconnect to execute DisconnectButton_Click and disconnect from the connected MCU.

DisConnect of the DLLs are processed in the cells shown in the following tables.

Table 7-4 DisConnect Function Details

#	Output cell	Output contents
1	Result	Success: Successful process NotConnect: Not connected

(3) Read function

Enter the required information in the cells shown in the table below, and press Read to execute ReadButton_Click function. The data of the connected MCU is read.

Table 7-5 Read Function Entries

#	Input cell	Input contents
1	Variable name	Variable to read
2	Endian	Little: Load in Little Endian Big: Load in Big Endian

Read of the DLLs are processed in the cells shown in the following tables.

Table 7-6 Read Function Details

#	Output cell	Output contents
1	Result	Success: Successful process NotConnect: Not connected to the MCU CommunicationFailed: Communication with the MCU failed. IllegalAddress: The specified address is invalid. IllegalDataType: Invalid datatype Incorrect IllegalEndianType:Endian
2	Read value	Read data

(4) Write function

Enter the required information in the cells shown in the table below, and press Write to execute WriteButton_Click function. Then, write the data to the connected MCU.

Table 7-7 Write Function Entries

#	Input cell	Input contents
1	Variable name	Variable to write
2	Write Value	Write data
3	Endian	Little: Write in Little Endian Big: Write in big endian

Write of the DLLs are processed in the cells shown in the following tables.

Table 7-8 Write Function Outputs

#	Output cell	Output contents
1	Result	Success: Successful process NotConnect: Not connected to the MCU IllegalAddress: Address is invalid IllegalValue: Write data is out of range CommunicationFailed: Communication with the MCU failed IllegalDataType: Invalid datatype Incorrect IllegalEndianType:Endian

(5) Scope start function

Enter the required information in the cells shown in the table below and press ScopeStart to execute SetProccess function, AddChannels function, and StartScope function in order by executing ScopeStartButton_Click function. However, if the function processing fails, ScopeStartButton_Click function processing terminates halfway.

Table 7-9 Scope setting function entry details

#	Input cell	Input contents
1	Sampling time[us]	Sampling time (μs)
2	Position	Position to determine the trigger
3	Number of records acquired	Number of records to retrieve
4	Trigger edge	Rise: Upward detection Fall: Down Detection Both: Both detected
5	Trigger mode	Single: Acquired once Normal: Acquired each time the trigger condition is satisfied. Auto: Always acquired

Table 7-10 Items entered for the function to add Scope channels

#	Input cell	Input contents
1	Variable name(CHx)	Variables read with CHx
2	Scale(CHx)	Scale of CHx ※The value obtained by multiplying the trigger level by the scale is reflected in the microcomputer as the trigger level

Table 7-11 Items entered for Scope startup function

#	Input cell	Input contents
1	Trigger channel	Channel number to trigger on
2	Trigger level	Triggering value

SetProccess function prints the results of Scope configuration function for DLs in the cells shown in the following tables.

Table 7-12 Outputting details of Scope setting function

#	Output cell	Output contents
1	Result	Success: Successful process IllegalTriggerEdge: Trigger edge is incorrect IllegalTriggerMode: Trigger mode is incorrect

AddChannels function prints the result of Scope channel information addition function of the DLL in the cells shown in the following table.

Table 7-13 Contents of the function for adding Scope channels

#	Output cell	Output contents
1	Result(CHx)	Success: Successful process IllegalDataType: Data type is invalid DisableDataType: Datatypes not supported by the board

StartScope function prints the result of the DLL's Scope initiation function in the cells shown in the following table.

Table 7-14 Items displayed by Scope startup function

#	Output cell	Output contents
1	Result	Success: Successful process IllegalScopeSetting: Scope setting is invalid IllegalTriggerLevel: Trigger level is invalid IllegalChannelSetting: Channel setting is invalid Invalid IllegalPositionSetting:Position setting IllegalRecordCount: The record length setting is invalid. IllegalSamplingTime: The sampling period is incorrect. NotConnect: Not connected to the MCU SendCommandFailed: Command communication failed

(6) Scope get condition function

Press ScopeGetCondition to obtain the status of Scope process by executing ScopeGetConditionButton_Click.

Scope status acquisition function for DLLs is processed in the cells shown in the following tables.

Table 7-15 Outputting details of Scope get condition function

#	Output cell	Output contents
1	Result	Success: Successful process PreparingToAcquireData: Preparing to acquire data NotConnect: Not connected to the MCU SendCommandFailed: Command communication failed

If the processing result of Scope status acquisition function is successful, it is judged that data acquisition is ready, and the data is acquired by pressing ScopeGetData button.

(7) Scope get data function

Enter the required information in the cells shown in the table below, and press ScopeGetData to execute Scope function. The data collected during ScopeGetDataButton_Click process is acquired.

Execute this function only when Scope status acquisition function of the DLLs has been successfully processed.

Table 7-16 Items entered for Scope get data function

#	Input cell	Input contents
1	Number of data acquired	Number of data items to be acquired
2	Endian	Little: Load in Little Endian Big: Load in Big Endian

Scope data retrieval function for DLLs is processed in the cells shown in the following tables.

Table 7-17 Items displayed by Scope get data function

#	Output cell	Output contents
1	Result	Success: Processing succeeded (data acquisition completed) IncorrectReadRange: The read area setting is invalid. IllegalGetData: Invalid retrieved data storage array setting DataAcquisitionNotPossible: Status is not data-obtainable NotConnect: Not connected to the MCU IllegalEndianType: Endian setting is invalid ScopeSettingNothing:Scope setting is not set SendCommandFailed: Command communication failed

(8) Scope stop function

By pressing ScopeStop, Scope process started by executing ScopeStopButton_Click function is stopped.

The DLL's Scope shutdown function is processed for each cell shown in the following table.

Table 7-18 Outputting details of Scope stop function

#	Output cell	Output contents
1	Result	Success: Successful process NotConnect: Not connected to the MCU SendCommandFailed: Command communication failed

(9) Map to csv function

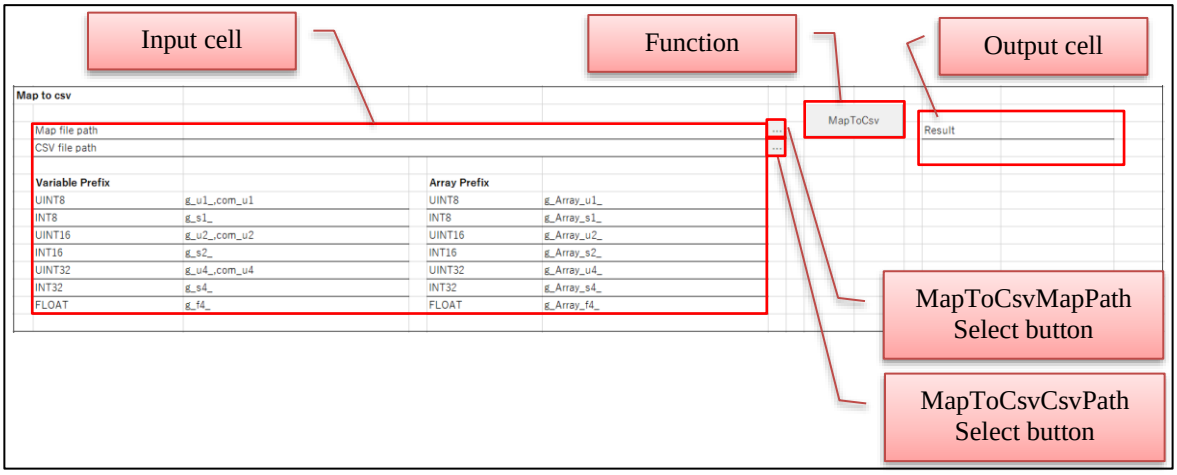


Figure 7-8 Map to csv function

Enter the required information in the cells shown in the table below, and press MapToCsv to execute MapToCsvButton_Click function. The contents of the map file are converted to CSV format (separated by commas), and the file is saved to the specified path.

Table 7-19 Items entered for Map to csv function

#	Input cell	Input contents
1	Map file path	Path of the actual existing Map file
2	CSV file path	File path that does not contain non-breaking characters ※Overwrite the specified csv file if it already exists in the output path
3	Variable Prefix	Variable Prefix data for each data type
4	Array Prefix	Array Prefix data for each data type

The processing results of the DLL Map file conversion CSV output function are output to each cell shown in the following table.

Table 7-20 Items displayed by Map to csv function

#	Output cell	Output contents
1	Result	Success: Successful process MapFileNotFound: Map-file does not exist MapFileCanNotRead: Map-file cannot be read MapFileIsOutputFromAnUnsupportedCompiler: The Map file is output from a compiler that is not supported. IncorrectVariableSettingsInTheMapFile: Invalid variable-setting in Mapfile AddressesCannotBeConvertedToHexadecimal: Addresses in a Mapfile cannot be converted to hexadecimal SizeCannotBeConvertedToHexadecimal: Size in a Mapfile cannot be converted to hexadecimal MapFileConversionError: Map-file conversion error OutputPathIsNotSet: Outputpath is not set OutputPathContainsInvalidCharacters: The Egress Path Contains Invalid Characters OutputPathIsAPathThatCannotBeOutput: A path to which the output path cannot be output is specified. FileAlreadyExists: The file already exists FileOutputFailure: File output failed

(a) Map-file selection function for Map to Csv function

Clicking MapToCsvMapPathSelect executes MapToCsvMapPathSelectButton_Click function. This opens the file selection dialog. Select the map file to convert to a CSV file by selecting the map file from the dialog that opens.

Output the path of the selected map file to each cell shown in the following table.

Table 7-21 Contents of the map file select function for Map to Csv function

#	Output cell	Output contents
1	Map file path	Map file to convert to a CSV file

(b) Csv-file output destination specification function for Map to csv function

Clicking MapToCsvCsvPathSelect executes MapToCsvCsvPathSelectButton_Click_Click and opens the Specify Path dialog. In the dialog that opens, specify the path to output the converted CSV file.

Outputs the specified path to each cell shown in the following table.

Table 7-22 Items output by the Csv-file output destination specification facility for Map to csv facility

#	Output cell	Output contents
1	CSV file path	Path to output the converted CSV file

(10) Map to memory function

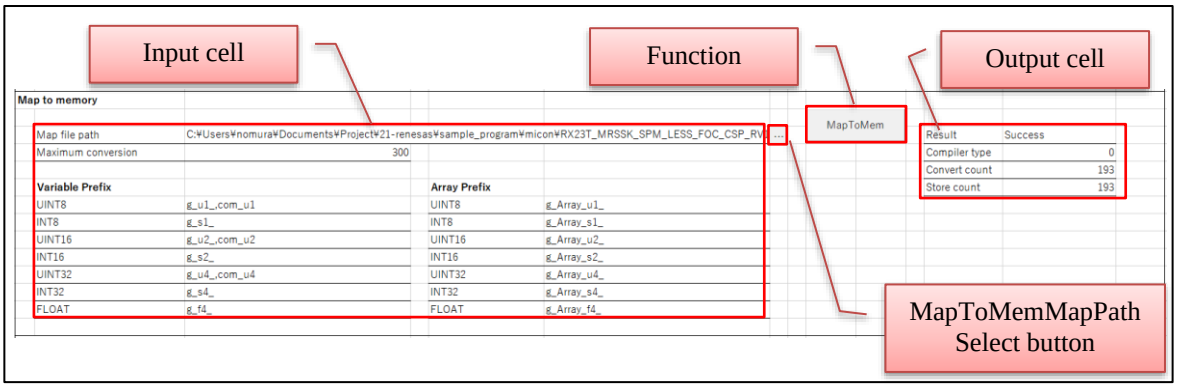


Figure 7-9 Map to memory function

Enter the required information in the cells shown in the table below and press MapToMem to execute MapToMemButton_Click function. The contents of the map file are expanded to the map sheet.

Table 7-23 Items entered for Map to memory function

#	Input cell	Input contents
1	Map file path	Path of the actual existing Map file
2	Maximum conversion	Conversion upper limit
3	Variable Prefix	Variable Prefix data for each data type
4	Array Prefix	Array Prefix data for each data type

The processing result of the DLL Map file conversion memory expansion function is output to each cell shown in the following table.

Table 7-24 Items displayed by Map to memory function

#	Output cell	Output contents
1	Result	Success: Successful process MapFileNotFound: Map-file does not exist MapFileCanNotRead: Map-file cannot be read MapFileIsOutputFromAnUnsupportedCompiler: The Map file is output from a compiler that is not supported. IncorrectVariableSettingsInTheMapFile: Invalid variable-setting in Mapfile AddressesCannotBeConvertedToHexadecimal: Addresses in a Mapfile cannot be converted to hexadecimal SizeCannotBeConvertedToHexadecimal: Size in a Mapfile cannot be converted to hexadecimal MapFileConversionError: Map-file conversion error InvaldDataLimitNum: The upper limit is invalid. InvalidStorageArrayForConversionResult: Invalid storage array definition
2	Compiler type	Compiler type
3	Convert count	Number of conversion
4	Store count	Number of storages
5	Map sheet	Map file information loaded into memory Column A: Variable Column B: Address Column C: Data type

(a) Map-file select function for Map to memory function

Clicking MapToMemMapPathSelect executes MapToMemMapPathSelectButton_Click function. This opens the file selection dialog. Select the map file to convert to a CSV file by selecting the map file from the dialog that opens.

Output the path of the selected map file to each cell shown in the following table.

Table 7-25 Contents of the map file select function for Map to memory function

#	Output cell	Output contents
1	Map file path	Map file to be expanded into the map sheet

(11) Auto read function

Enter the required information in the cells shown in the table below, and press AutoReadStart to execute AutoReadButton_Click function. The data of the connected MCU is read periodically.

Table 7-26 Items entered for Auto read function

#	Input cell	Input contents
1	Variable name	Variable to read
2	Endian	Little: Load in Little Endian Big: Load in Big Endian
3	Sampling time[ms]	Reading cycle

The table below shows the results of Read process in each cell.

The output is overwritten each time Read is executed.

Table 7-27 Outputting Contents of Auto read Function

#	Output cell	Output contents
1	Result	Processing result
2	Read value	Read data

7.4.2 Map sheet

The map sheet displays the map file information expanded to the memory by Map to memory function.

Variables expanded in this sheet are registered in the drop-down list of the cell in which the variable is entered in sample sheet.

This sheet is also used as a reference for addresses and data types when reading and writing variables.

	A	B	C
1	401	g_u1_motor_status	0
2	402	gui_u1_active_gui	1
3	403	com_u1_sw_userif	0
4	404	g_u1_sw_userif	0
5	405	com_u1_mode_system	0
6	406	g_u1_mode_system	0
7	408	com_u1_direction	0
8	409	com_u1_enable_write	0
9	0000040a	g_u1_enable_write	0
10	432	g_u2_conf_sw_ver	2
11	434	g_u2_max_speed_rpm	2
12	436	com_s2_ref_speed_rpm	3
13	438	com_u2_max_speed_rpm	2
14	0000043a	com_u2_overspeed_limit_rpm	2
15	0000043c	com_u2_offset_calc_time	2
16	0000043e	com_u2_mtr_pp	2
17	440	com_u2_id_up_speed_rpm	2
18	442	com_u2_id_down_speed_rpm	2
19	456	g_u2_conf_hw	2
20	458	g_u2_conf_sw	2
21	0000045a	g_u2_conf_tool	2
22	000008f0	com_f4_mtr_r	6
23	000008f4	com_f4_mtr_ld	6
24	000008f8	com_f4_mtr_lq	6
25	000008fc	com_f4_mtr_m	6
26	900	com_f4_mtr_j	6
27	904	com_f4_current_omega	6
28	908	com_f4_current_zeta	6
29	0000090c	com_f4_speed_omega	6
30	910	com_f4_speed_zeta	6

sample | map | graph | type

Figure 7-10 map seat

7.4.3 Graph seat

Graph sheet displays the data obtained using scope function.

In addition, the graph created from the displayed data is displayed.

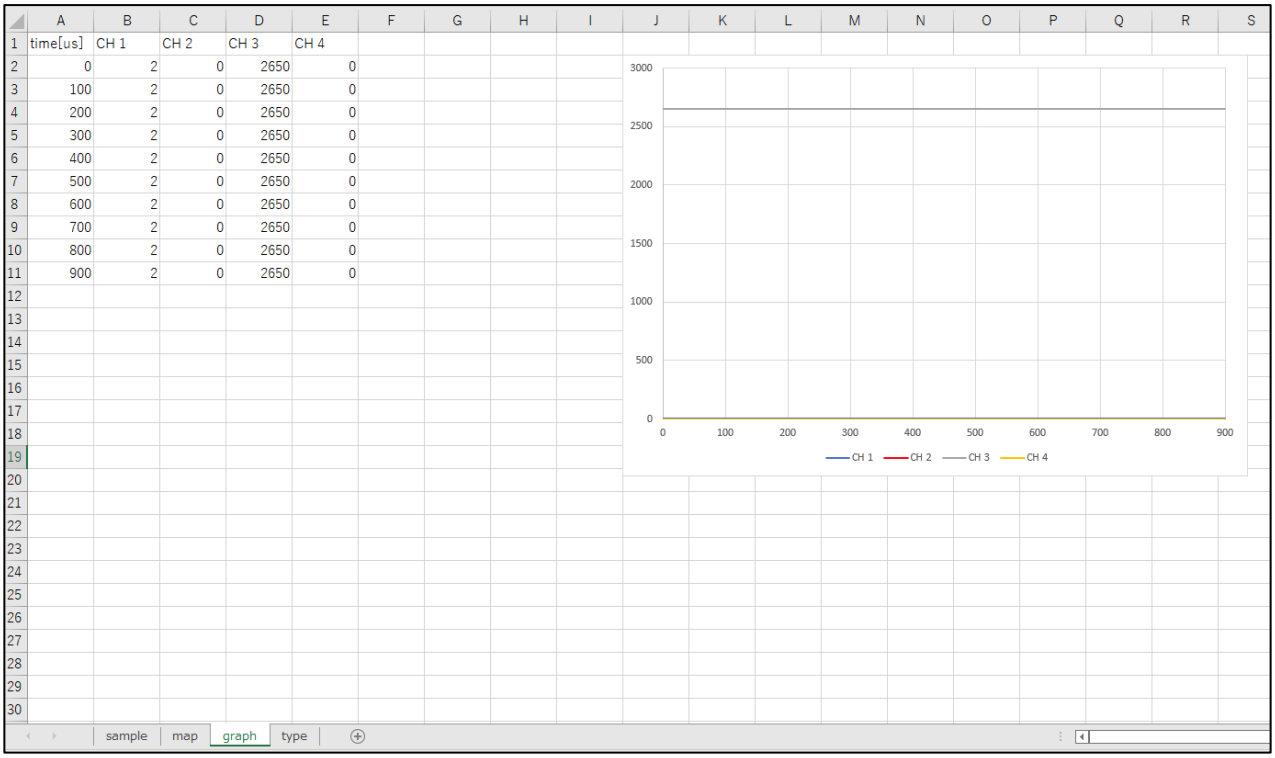


Figure 7-11 graph seat

7.4.4 Type seat

Type sheet is a sheet that summarizes the correspondence between strings (enumerators) and values of enum types defined in the library.

VBAs cannot refer to enum defined in the library as input/output values for individual functions.

For this reason, the input/output values of each function must be treated as numeric values. Therefore, the information in this sheet is used to convert them to input values and to convert them to the meaning of the processing results.

	A	B
1	ConnectResponse	
2	Value	Meaning
3	0	Success
4	1	IllegalCOMPort
5	2	IllegalBaudrate
6	3	ConnectFailed
7	4	SendCommendFailed
8		
9	DisconnectResponse	
10	Value	Meaning
11	0	Success
12	1	NotConnect
13		
14	ReadResponse	
15	Value	Meaning
16	0	Success
17	1	NotConnect
18	2	CommunicationFailed
19	3	IllegalAddress
20	4	IllegalDataType
21	5	IllegalEndianType
22		
23	WriteResponse	
24	Value	Meaning
25	0	Success
26	1	NotConnect
27	2	IllegalAddress
28	3	IllegalValue
29	4	CommunicationFailed
30	5	IllegalDataType
31	6	IllegalEndianType
32		
33	ScopeStartResponse	
34	Value	Meaning
35	0	Success
36	1	IllegalScopeSetting

sample | map | graph | type

Figure 7-12 type seat

7.5 Customization example

This section explains how to duplicate Read function as an example of customizing this sample program.

7.5.1 How to duplicate Read function

This section describes how to duplicate Read function.

Table 7-28 Steps for duplicating Read Function

#	Procedure description
1	Launch Design Mode
2	Copying a Screen
3	Define cell name
4	Define button names
5	Create an event for a new button
6	Copy Processing in Event
7	Rename an I/O cell to a defined name
8	End of Design Mode

(1) Launch Design Mode

Press the "Design Mode" button on the development tab at the top of Excel to switch to Design mode.

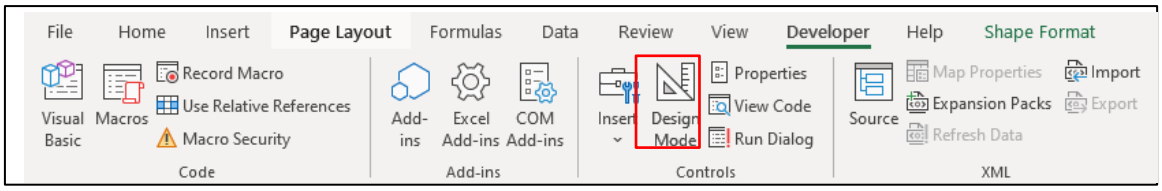


Figure 7-13 Design Mode Activation Method

(2) Copying a Screen

Copy the parts related to Read function of Sample seat and attach them to the desired positions.

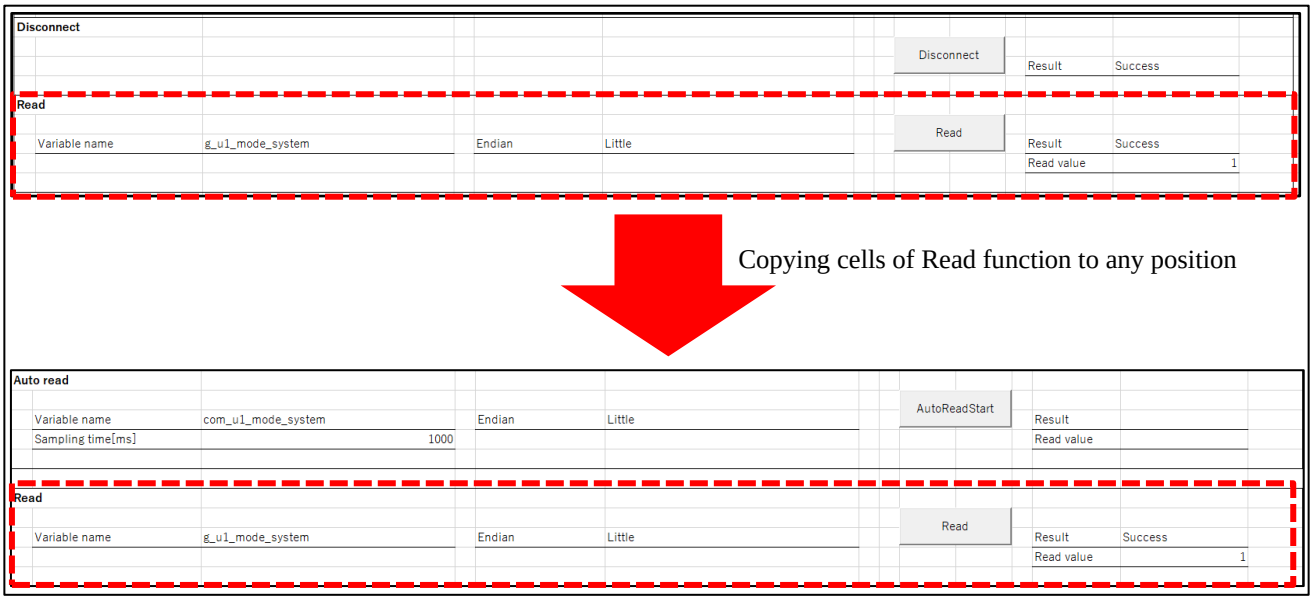


Figure 7-14 Example of Screen Copy

(3) Define cell name

In the processing of each function, the cell to acquire the input and the cell to output the processing result are specified by the name of the defined cell. If you copy a cell, the cell name is not copied. Therefore, define a name for the destination I/O cell.

You define the name of the cell from the New Name screen that opens by right-clicking the cell and choosing Define Name from the menu.

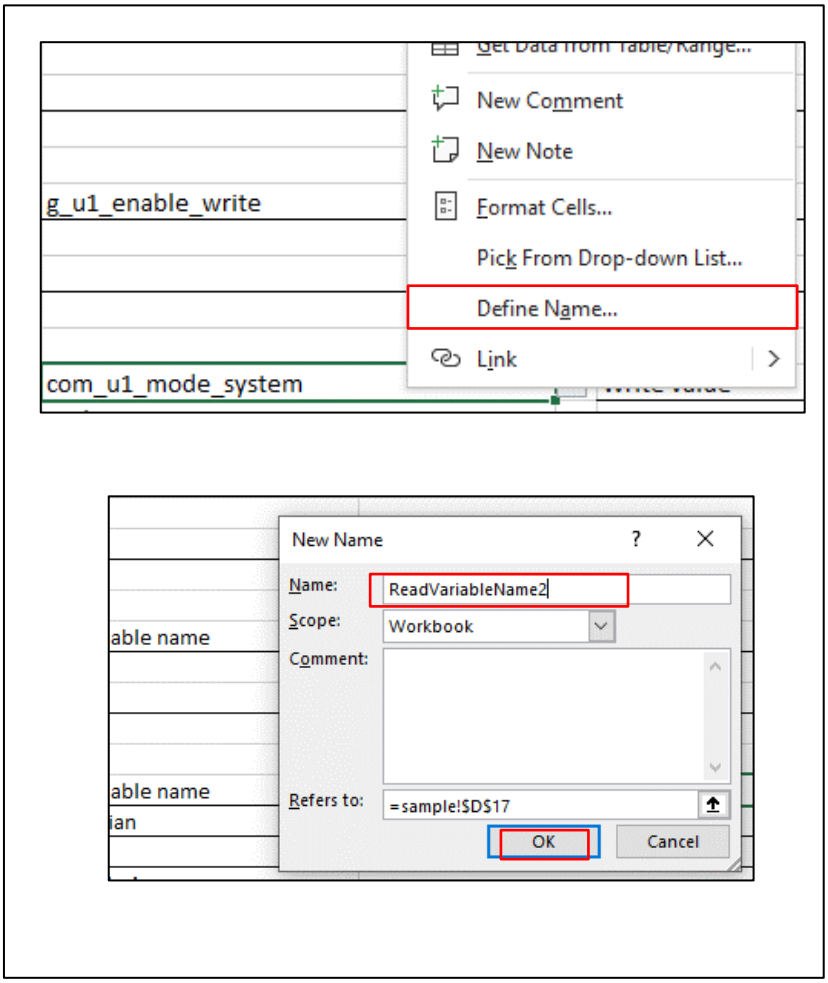


Figure 7-15 How to define cell names

(4) Define button names

Define the name of Read key pasted in step 3.

Right-click on Read button. Select Properties from the menu that appears.

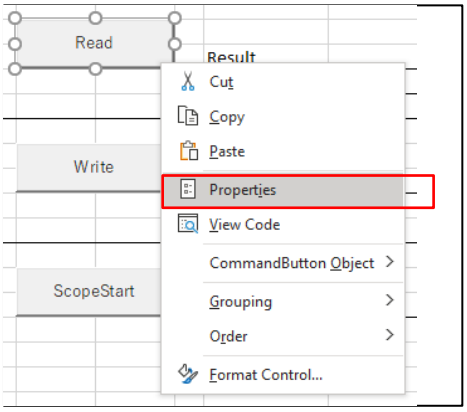


Figure 7-16 Displaying the object properties screen

Change the object name on the properties screen to the desired name.

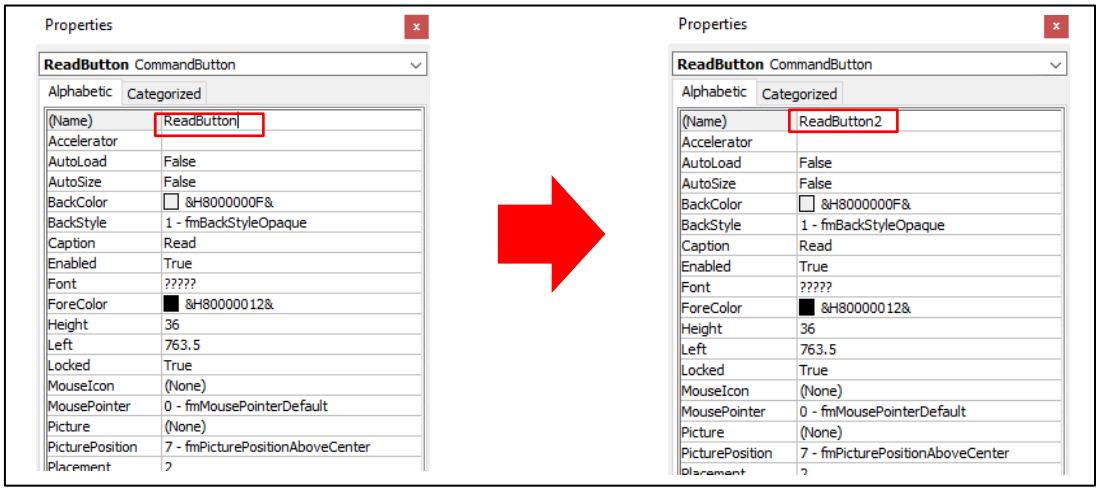


Figure 7-17 Changing the object name

(5) Create an event for a new button

Double-clicking on a newly created button adds a click event for the new button to the process and registers the added event with the button.

The event name is registered with "(Object name)_Click".

(6) Copy Processing in Event

Copies the operation of ReadButton_Click event to the event of the new button.

(7) List information copy for input information check

Copy the list information for input information check in the Common module and change the name of the list information and the cell names defined in "(3) Define cell name".

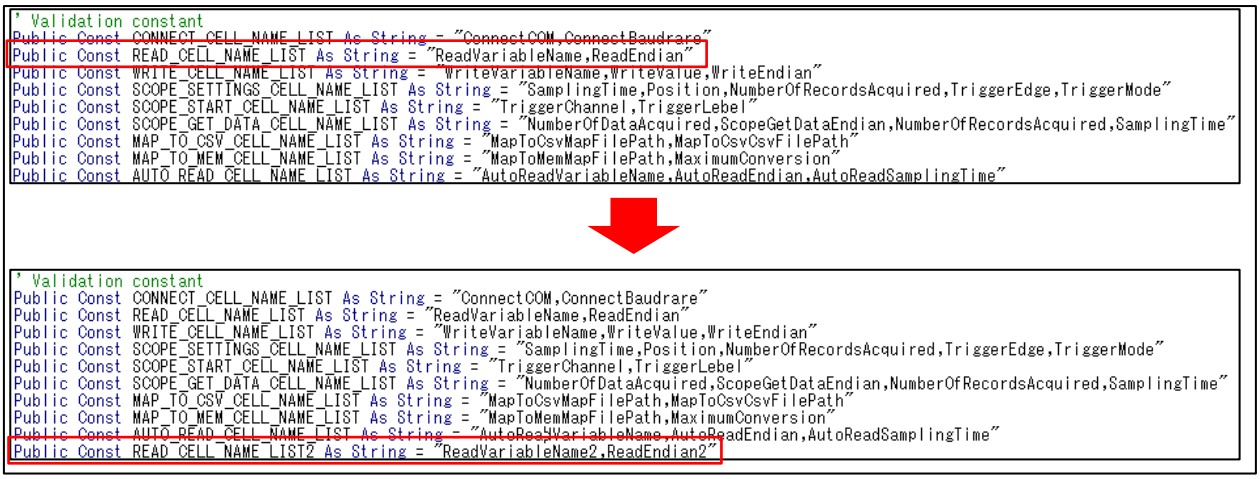


Figure 7-18 List information copy for input information check

(8) Rename an I/O cell to a defined name

Change the names of cells in the process copied in "(6) Copying Processes in Event" to the names defined in "(3) Define Cell Name".

Also, change the names of the sections that refer to list information for input information check to the names added in "List information copy for input information check ".

The red framed areas in the following figure refer to cells, and the blue framed areas refer to list information for input information checking.

```
Private Sub ReadButton2_Click()
    Dim result As Long
    Dim variableName As String
    Dim readData As String
    Dim endianType As Byte
    Dim address As Long
    Dim dataType As Long

    ' Initialize the result display cell.
    Range("ReadResult2") = ""
    Range("ReadValue2") = ""

    ' Input validation
    If ValidationValues(READ_CELL_NAME_LIST2, "Input error") <> VALIDATION_VALUES_SUCCESS_VALUE Then
        Exit Sub
    End If

    ' Initialize variables.
    readData = ""

    ' Retrieve the input contents.
    variableName = Range("ReadVariableName2")
    endianType = ConvertTypeToValue(Range("ReadEndian2"), "EEndianType")

    ' Obtain the address and data type from the variable name.
    ' If the acquisition is successful, read processing is performed.
    If GetAddress(variableName, address, dataType) = GET_ADDRESS_SUCCESS_VALUE Then
        ' Execute the Read function of the RMW communication library.
        result = comLib.Read(readData, address, dataType, endianType)
        ' Convert and display processing results.
        Range("ReadResult2") _
            = ConvertResultValueToMean(result, "ReadResponse")

        ' If the read process is successful, the read value is displayed.
        If result = LIBRARY_PROCESS_SUCCESS_VALUE Then
            Range("ReadValue2") = readData
        End If
    End If
End Sub
```

Figure 7-19 Changes in processing within events

(9) End of Design Mode

Press the "Design Mode" button on the development tab at the top of Excel to exit the design mode.

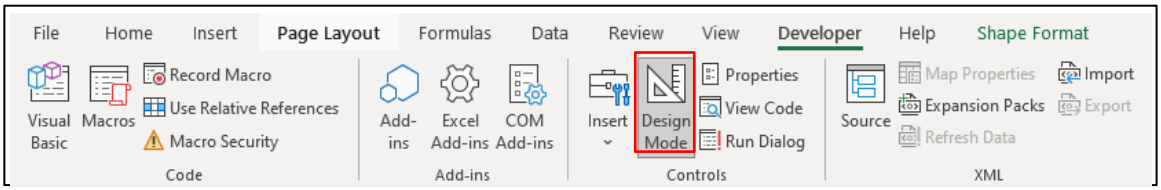


Figure 7-20 How to Exit Design Mode

7.6 Sample Program Module Configuration

The following table lists the functions implemented in each module of the sample program.

Module name	Function Name	Overview	Argument	Return value
ThisWorkbook	Workbook_Open	Event that occurs when a book is opened. Execute Sheet_Initialization of SampleSheet.	None	None
	Workbook_BeforeClose	An event that occurs before the book is closed. Execute the Sheet_Finalization function of SampleSheet.	1st argument: Variable that stops the close operation by setting True	None
SampleSheet	Sheet_Initialization	Initializes SampleSheet module. You initialize AutoRead key.	None	None
	Sheet_Finalization	Terminate the SampleSheet module. Stop Scope processing and disconnect from microcontroller.	None	None
	ConnectButton_Click	Event when Connect button is pressed. Fetches the input from the cell, executes Connect of the DLLs, and prints the result onto the cell.	None	None
	DisconnectButton_Click	Event when DisConnect button is pressed. Execute Disconnect function of the DLL and print the result on the cell.	None	None
	ReadButton_Click	Event when Read button is pressed. Fetches the input from the cell, executes Read of the DLLs, and prints the result onto the cell.	None	None
	WriteButton_Click	Event when Write button is pressed. Fetches the input from the cell, executes Write of the DLLs, and prints the result onto the cell.	None	None
	ScopeStartButton_Click	Event when ScopeStart button is pressed. Executes SetProcess function, AddChannels function, and StartScope function.	None	None
	SetProccess	Fetches the input from the cell, executes SetProcessInfo of the DLLs, and prints the result onto the cell.	1st argument: Scope setting	Processing result

	AddChannels	This function obtains the input data of each channel from the cell, executes AddChannelInfo function of the DLL by the number of channels, and outputs the result to the cell.	1st argument: Scope setting	Processing result
	StartScope	Fetches the input from the cell, executes StartScope of the DLLs, and prints the result onto the cell.	1st argument: Scope setting	None
	ScopeGetConditionButton_Click	Event when ScopeGetCondition button is pressed. Fetches the input from the cell, executes ScopeGetCondition of the DLLs, and prints the result onto the cell.	None	None
	ScopeGetDataButton_Click	Event when ScopeGetData button is pressed. Fetches the input from the cell, executes ScopeGetData of the DLLs, and prints the result onto the cell.	None	None
	ClearGraphData	The cell displaying the data acquired by Scope function is cleared.	None	None
	ScopeStopButton_Click	Event when ScopeStop button is pressed. Fetches the input from the cell, executes ScopeStop of the DLLs, and prints the result onto the cell.	None	None
	MapToCsvMapPathSelectButton_Click	Event when MapToCsvMapPathSelect button is pressed. Select the map file in the file selection dialog and output the result to the cell.	None	None
	MapToCsvCsvPathSelectButton_Click	Event when MapToCsvCsvPathSelect button is pressed. Specify the path to output the CSV file in the file specification dialog box, and output the result on the cell.	None	None
	MapToCsvButton_Click	Event when MapToCsv button is pressed. Fetches the input from the cell, executes ConvertMapToCSV of the DLLs, and prints the result onto the cell.	None	None
	MapToMemMapPathSelectButton_Click	Event when MapToMemMapPathSelect button is pressed. Select the map file in the file selection dialog and output the result to the cell.	None	None
	MapToMemButton_Click	Event when MapToMem button is pressed. Fetches the input from the cell, executes ConvertMapToMemory of the DLLs, and prints the result onto the cell.	None	None
	AutoReadButton_Click	Event when AutoRead button is pressed. When AutoReadStartButton is pressed: Input data is acquired from	None	None

		the cell, and TimerStart of TimerModule is executed. When AutoReadStopButton is pressed: AutoReadStop is executed.		
	AutoRead	Fetches the input from the cell, executes Read of the DLLs, and prints the result onto the cell.	None	None
	AutoReadStop	Execute TimerStop of TimerModule.	None	None
Common	GetAddress	Search the Map sheet to get the address and data type from the variable name.	1st argument: Variable name 2nd: Address storage variable 3rd: Variables for storing data types	Processing result
	VaridationValues	Performs a cell input value check.	1st argument: Cell name 2nd argument: Error message"	Processing result
	SelectFilePath	Map file is selected.	None	Selected path
	GetSaveFilePath	Specify the path of the CSV file.	None	Specified path
	ConvertTypeToValue	Search the tables in Type and convert them from type names to numbers.	1st argument: String of type 2nd argument: Range name of the table to be searched"	Value of the processing result
	ConvertResultValueToMean	Search the tables in Type sheet. Convert the tables to type names.	1st argument: Value of type 2nd argument: Range name of the table to be searched"	Processing result string
TimerModule	TimerProc	Execute AutoRead of SampleSheet.	None	None
	TimerStart	Invokes a timer that executes TimerProc periodically.	1st argument: Period	None
	TimerStop	Stop the timer.	None	None

Revised Records

Rev.	Issue Date	Details of revision	
		Page	Point
1.00	Jun, 30, 2022	-	New issue

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins

   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
6. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/.