InfoSOSA
Modbus Device Connection

DMC Co., Ltd
https://www.dush.co.jp/english/

Copyright and Trademarks

- Copyright of this manual is owned by DMC Corporation.
- Reproduction and/or duplication of this product and/or this manual, in any form, in whole or in part, without permission are strictly prohibited.
- This product and/or the contents in this document are subject to change without prior notice.
- DMC shall not be held liable for any damages or losses, nor be held responsible for any claims by a third party as a result of using this product.
- Microsoft®, Windows® are trademarks or registered trademarks of Microsoft Corporation USA and other countries.
- Other company and product names mentioned herein are trademarks or registered trademarks of their respective companies.

About Notations

Item for IS series.

Item for IS-APP(EM series).

Item for both IS series and IS-APP.

Table of Contents

Copyright and Trademarks 2

About Notations 2

Table of Contents 3

1. Introduction 4
   - 1.1 Overview 5
   - 1.2 Compatible Modbus Devices 7
   - 1.3 Reference Documents 8

2. Preparation 9
   - 2.1 Common (InfoSOSA Builder) 10
   - 2.2 EM series 11
   - 2.3 IS731 series 11

3. Usage 12
   - 3.1 Project Settings 13
     - 3.1.1 Modbus memory creation 13
     - 3.1.2 Number of simultaneous read data settings 17
     - 3.1.3 Priority section setting 19
     - 3.1.4 Export Modbus Settings 22
     - 3.1.5 Importing Modbus settings 23
   - 3.2 IS-APP startup settings 24
   - 3.3 IS731 Communication Settings 26
   - 3.4 How to Read and Write Modbus Device Values 28
     - 3.4.1 Display Device Values of Modbus Devices 28
     - 3.4.2 Write Value to Modbus Device 31

4. Reference 34
   - 4.1 Modbus communication memory 35
     - 4.1.1 Property 36
     - 4.1.2 On Change Value Event 39
     - 4.1.3 Host Communication (InfoSOSA Protocol) 41
   - 4.2 Events for Modbus communication 42
     - 4.2.1 Link Error 43
     - 4.2.2 Link Recover 45
     - 4.2.3 First Sync Complete 46
   - 4.3 IS-APP command line arguments 47
   - 4.4 Modbus setting file 49
     - 4.4.1 csv format (individual setting) 49
     - 4.4.2 ini format (collective setting) 54

Inquiries 57

1. Introduction

1.1 Overview

This document describes the communication function using the Modbus protocol of InfoSOSA.

Device values of Modus devices can be displayed on the InfoSOSA, and InfoSOSA input values can be reflected to the Modbus device.

* In this document, devices that can be connected using the Modbus protocol are referred to as Modbus devices.

For information on InfoSOSA functions other than the Modbus communication function, please refer to the separate “InfoSOSA Reference Manual”.

Version of InfoSOSA Builder

The InfoSOSA Builder needs to have the following version.

Target InfoSOSA Builder version	2.7.1 ~

* The InfoSOSA Builder simulator does not support the Modbus feature.

Version of IS-APP

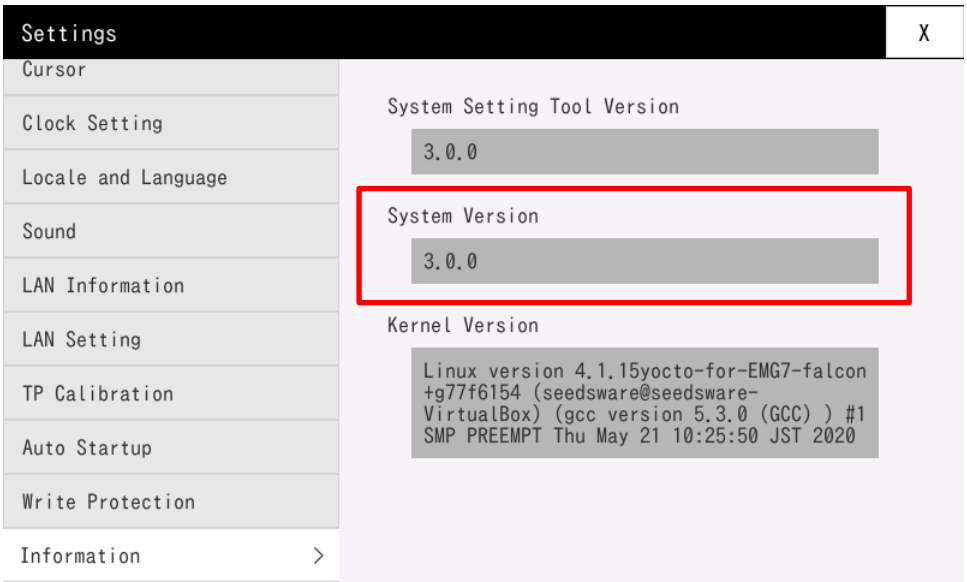
IS-APP related applications must be the following versions.

Target IS-APP version	2.4.1 ~
Target IS-APP SettingTool version	3.1.1 ~

System version of EM series

The EM series must have the following system version.
Check the system version of the EM series from the system setting tool.
For the system setting tool, refer to the attached "EM Series Tool Manual".

Target system version	3.0.0 ~



1.2 Compatible Modbus Devices

InfoSOSA can connect to the following devices that support the Modbus protocol.

Item	Content
Protocol	Modbus/TCP Slave mode Modbus/RTU Slave mode * The InfoSOSA side becomes the master and connects to Modbus devices in slave mode.
Function code	0x01 (Coil read) 0x03 (Register read) 0x0F (Coil write) 0x10 (Register write)
Data type	Bit (1bit) Byte (1byte) signed/unsigned integ Word (2byte）signed/unsigned integ DoubleWord (4byte) signed integ
Number of connected units	1

* The Modbus device must return a response command within 100ms after receiving the command from InfoSOSA.
* Connection is not guaranteed with all Modbus devices that meet the above.

1.3 Reference Documents

The documents related to this document are as follows. Please refer to it according to your purpose.

InfoSOSA Reference Manual

This manual describes InfoSOSA functions and specifications.

InfoSOSA Builder Operation Manual

This manual describes how to operate the InfoSOSA Builder.

IS731 Series Startup Guide

This is a manual for IS731 Series users.
Provides explanations on the features of the InfoSOSA IS731 Series, functions unique to the InfoSOSA IS731 Series, as well as a tutorial.

IS-APP Startup Guide

This is a manual for IS-APP users.
This manual describes the features of the IS-APP, provides a tutorial, as well as explains functions and specifications unique to the IS-APP.

EM Series Software Development Manual

This manual describes how to develop software that operates on the EM series.

EM Series Tool Manual

This manual describes how to use the tools installed in the EM series.

2. Preparation

2.1 Common (InfoSOSA Builder)

Install a version of InfoSOSA Builder that supports Modbus functionality.

Target InfoSOSA Builder version	2.7.1 ~

* The InfoSOSA Builder simulator does not support the Modbus feature.

Please refer to the attached "InfoSOSA Builder Operation Manual" for installation instructions.

2.2 EM series

Transfer the Modbus communication compatible version "IS-APP" and "IS-APP Setting Tool" to the EM series.

Transfer file	Forwarding destination	Permission
is_app	/usr/bin/	755
libisapi.so	/usr/lib/	644
isapp_setting	/usr/bin/	755

To transfer, write protection must be released.
For details, refer to "EM Series Software Development Manual".

The transfer file is included in the download data.
The file path varies depending on the model.

Model	Transfer file Path
EMG7-***A8-****-**7	*[download data]*\software\Modbus\IS-APP-A8
EM8-***A7-****-**7	*[download data]*\software\Modbus\IS-APP-A7
EMG8-***A7-****-**7	

2.3 IS731 series

There is no preparation work for the IS731 series. Simply download from InfoSOSA Builder and it will be installed automatically.

3. Usage

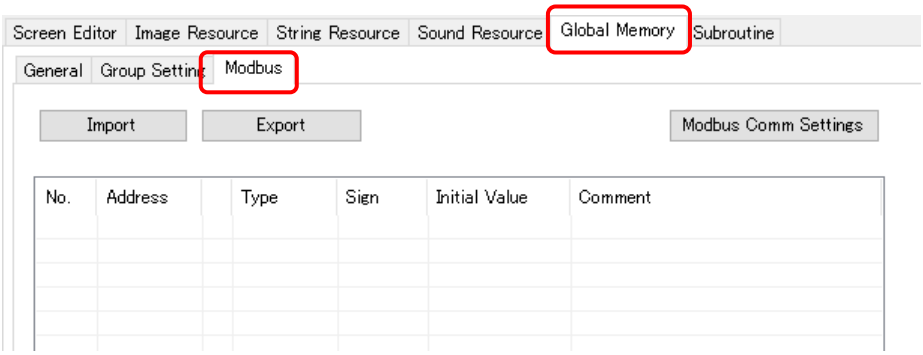
3.1 Project Settings

This section describes how to create an InfoSOSA Builder project (screen data) to connect to the Modbus device.

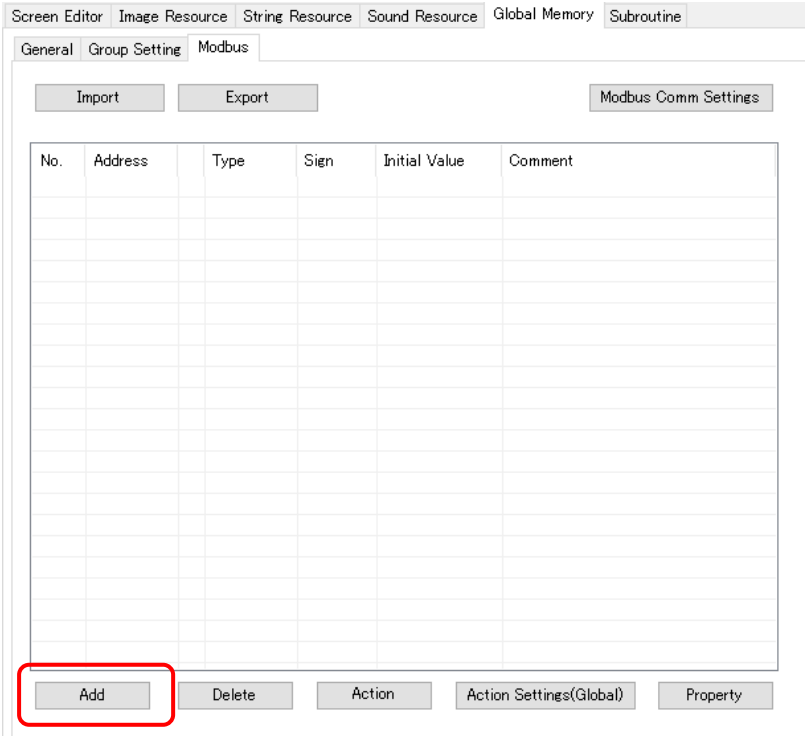
3.1.1 Modbus memory creation

Create a memory for communication with Modbus devices.

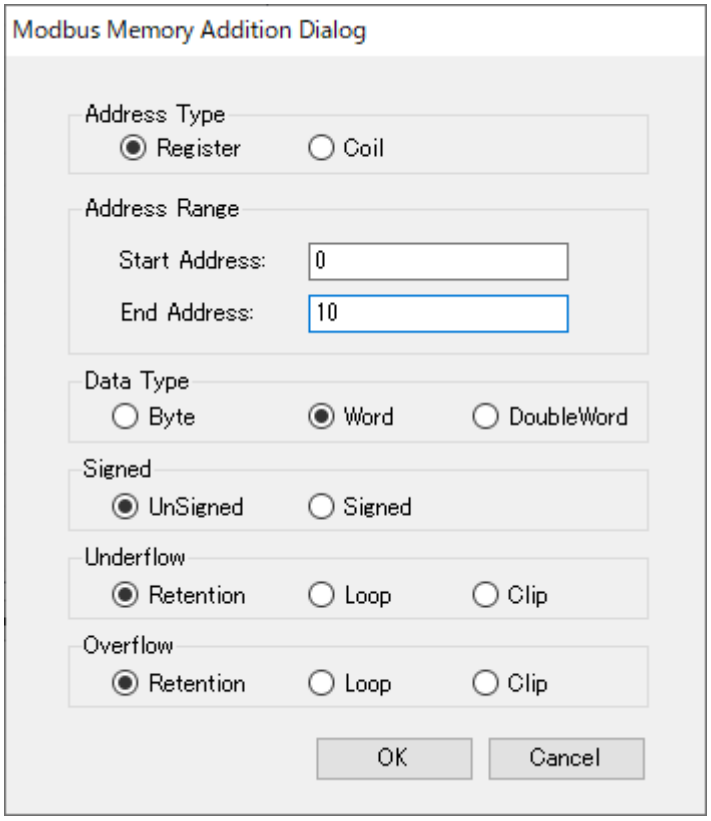
1. Display the “Modbus” tab in the “Global Memory” tab.



2. Click the "Add" button.



3. Set the address range of the memory to be registered and the initial value of the parameter.



Parameter	Description
Address Type	Select the type of Modbus memory to register from "Register" and "Coil". The default is "Register". In case of "Register", communication with Modbus device is done by the following function code. Read:0x03 Write:0x10 In the case of "Coil", communication to Modbus device is performed with the following function code. Read:0x01 Write:0x0F
Address Range	Specifies the Modbus address range to synchronize. Enter the start address and end address in decimal. For the correspondence between the Modbus address and the register of the Modbus device, please check the specifications of your Modbus device. Up to 2000 memory (registers and coils total) for communication with Modbus devices can be registered, including global memory.

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>Data Type</td><td>In the case of "Register", you can select from "Byte", "Word", and "DoubleWord". The default is "Word".

<table>
<tr><th>Data Type</th><th>Size</th></tr>
<tr><td>Byte*1</td><td>1byte</td></tr>
<tr><td>Word</td><td>2byte</td></tr>
<tr><td>DoubleWord*2</td><td>4byte</td></tr>
</table>

*1
If set to “Byte”, the upper data (1 byte) of the Modbus device will be ignored and acquired.

*2
When “DoubleWord” is set, the next address is used as upper data (2 bytes). Combined with the following address.

In the case of “Coil”, it is fixed to "Bit".

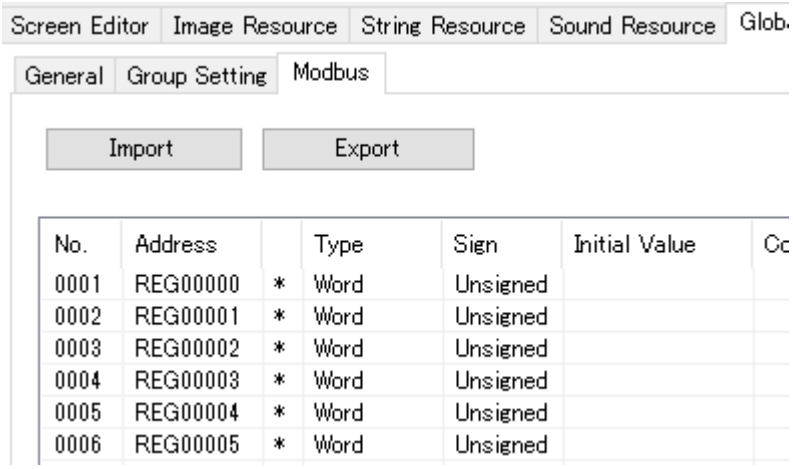
<table>
<tr><th>Data Type</th><th>Size</th></tr>
<tr><td>Bit</td><td>1bit</td></tr>
</table></td></tr>
<tr><td>Signed</td><td>Select whether to display the acquired data as "Signed" or "Snsigned". The default is "Unsigned".
In case of double word, it is fixed as "Signed". In case of “Coil”, it is invalid.

<table>
<tr><th>Data Type</th><th>Signed</th></tr>
<tr><td>Byte</td><td>Signed/Unsigned</td></tr>
<tr><td>Word</td><td>Signed/Unsigned</td></tr>
<tr><td>DoubleWord</td><td>Unsigned</td></tr>
</table></td></tr>
<tr><td>Underflow</td><td>Select the operation when a value smaller than the minimum value of the target Modbus memory is set. The default is "Retention". In case of "Coil", it is invalid.

<table>
<tr><th>Setting</th><th>Action</th></tr>
<tr><td>Retention</td><td>Saves value right before underflow occurs.</td></tr>
<tr><td>Loop</td><td>Underflow memory subtracted from maximum value will be the value of target memory.</td></tr>
<tr><td>Clip</td><td>Set minimum value to target memory.</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>Overflow</td><td>Defines the operation when a value larger than the maximum value of the target Modbus memory is set. The default is " Retention ". In case of "Coil", it is invalid.
<table>
<tr><th>Setting</th><th>Action</th></tr>
<tr><td>Retention</td><td>Saves value right before overflow occurs.</td></tr>
<tr><td>Loop</td><td>Overflow memory added to minimum value will be the value of target memory.</td></tr>
<tr><td>Clip</td><td>Set maximum value to target memory.</td></tr>
</table></td></tr>
</table>

The Modbus device communication memory is registered as shown below.



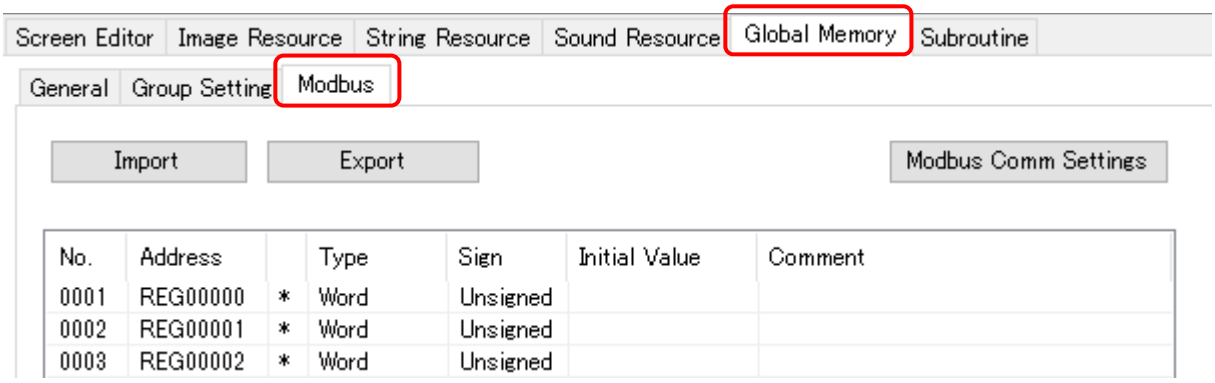
3.1.2 Number of simultaneous read data settings

For the connected Modbus device, set the maximum number of addresses to read collectively in one communication. Set for each function code.

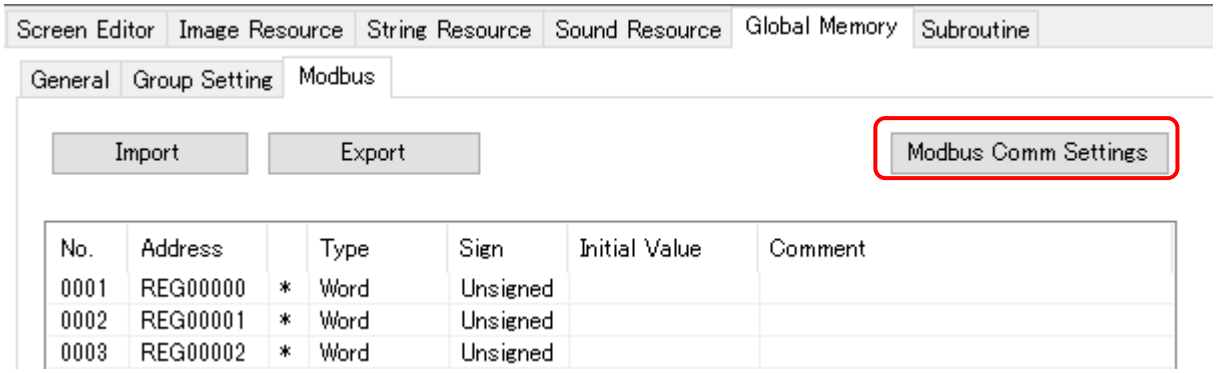
The higher the value, the better the performance, but please check the specifications of the connected Modbus device for the maximum value.

* If a value larger than the maximum value on the Modbus device side is set, a communication error will occur.

1. Display the “Modbus” tab in the “Global Memory” tab.



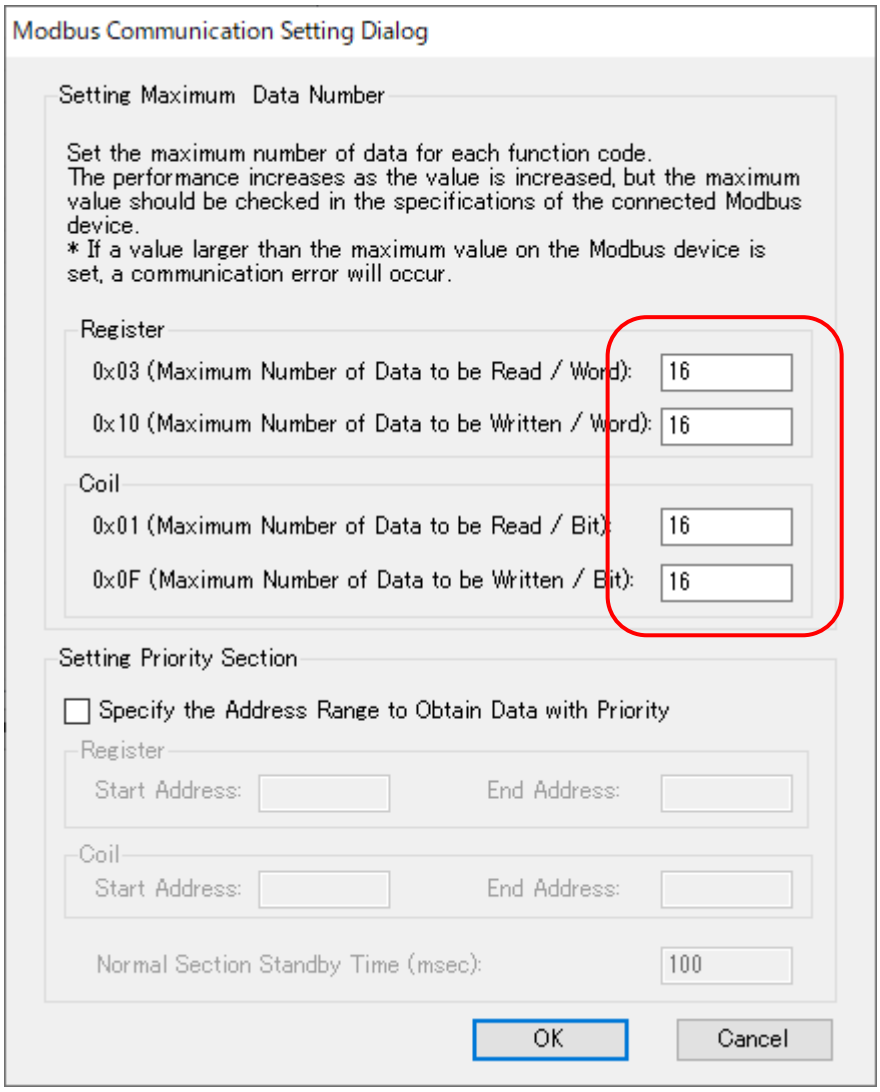
2. Click the "Modbus Comm Settings" button.



3. Set the maximum number of data for each function code.

The higher the value, the better the performance, but please check the specifications of the connected Modbus device for the maximum value.

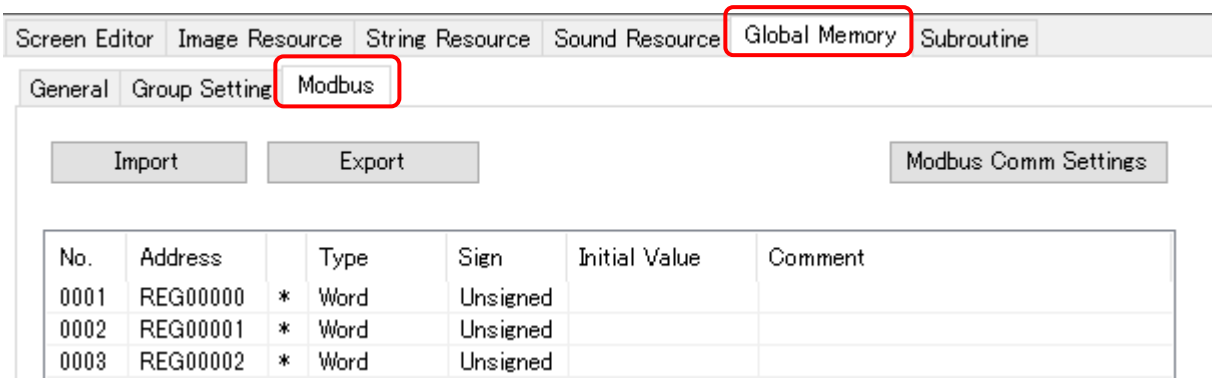
* If a value larger than the maximum value on the Modbus device side is set, a communication error will occur.



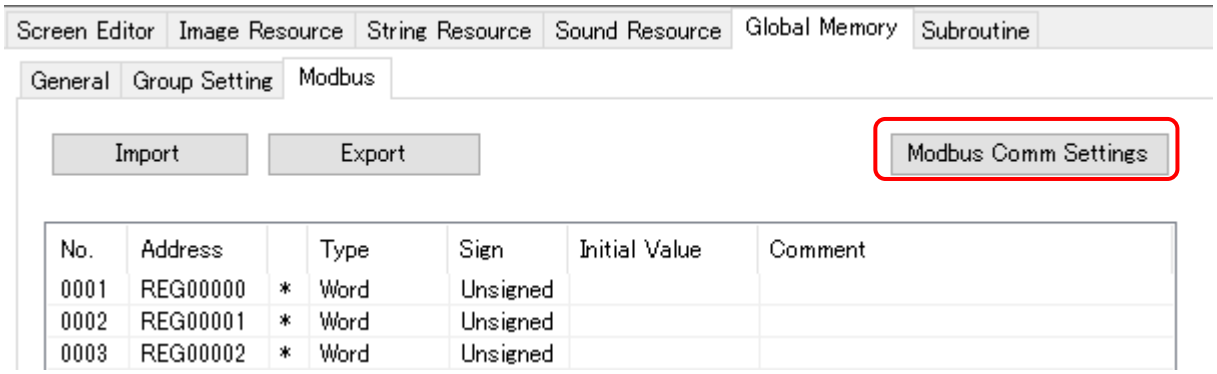
3.1.3 Priority section setting

The time required to read the values of all devices increases in proportion to the number of Modbus memories to register, so the time from the change of the value on the Modbus device side to the reflection on the screen display of InfoSOSA increases.
In that case, it is possible to set the address to read the value with priority.

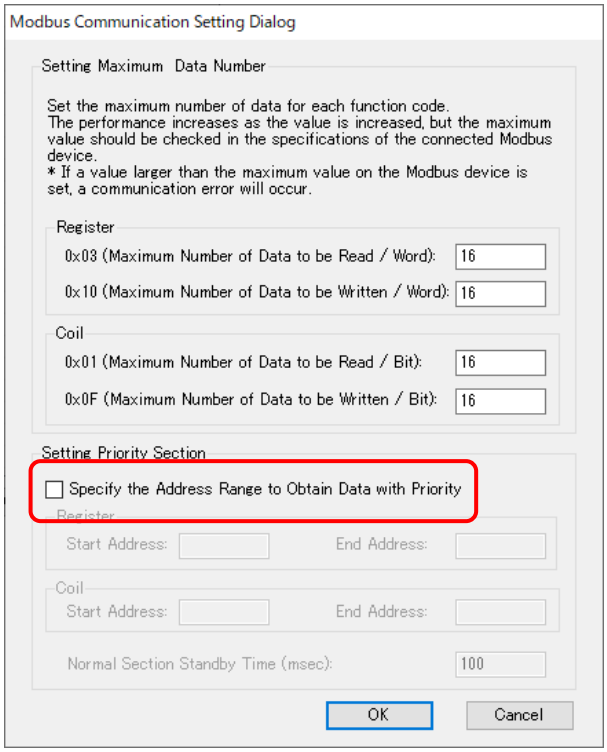
1. Display the “Modbus” tab in the “Global Memory” tab.



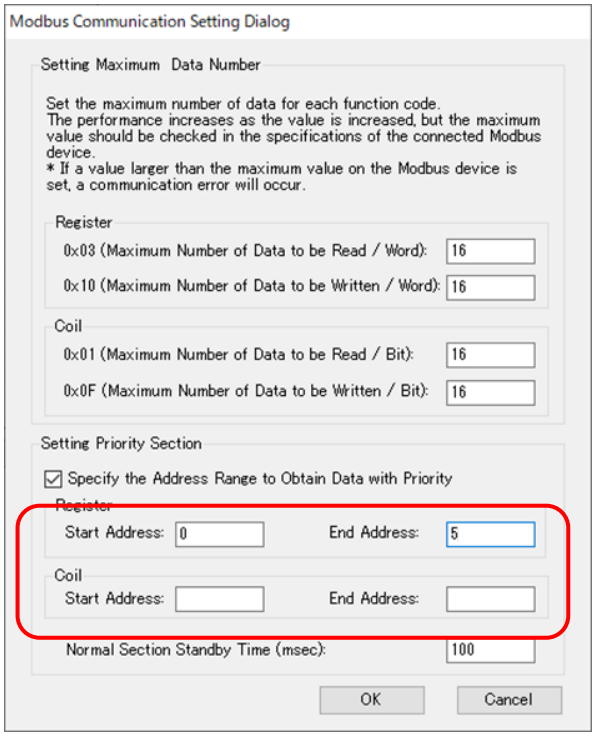
2. Click the "Modbus Comm Settings" button.



3. Select the "Specify the address range to get data with priority" check box.

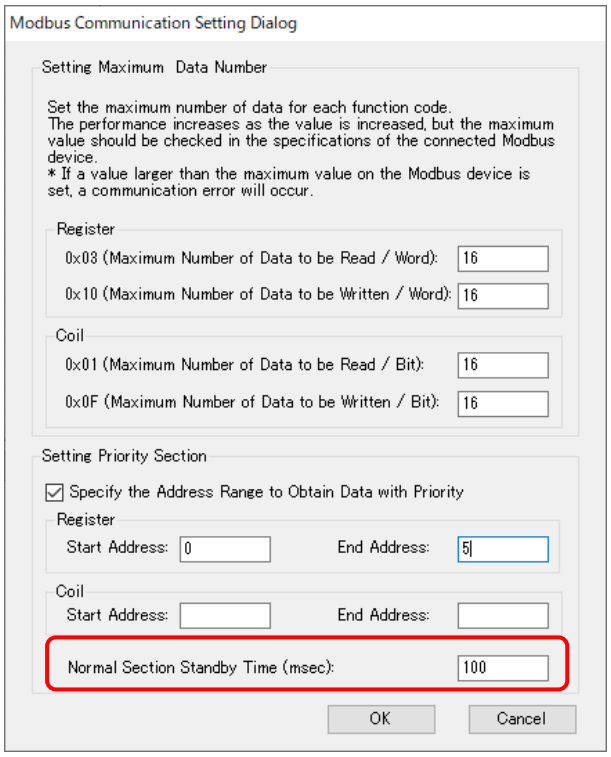


4. Please specify the "Start Address" and "End Address" to be set in the priority section. “Register” and “Coil” can be specified respectively.

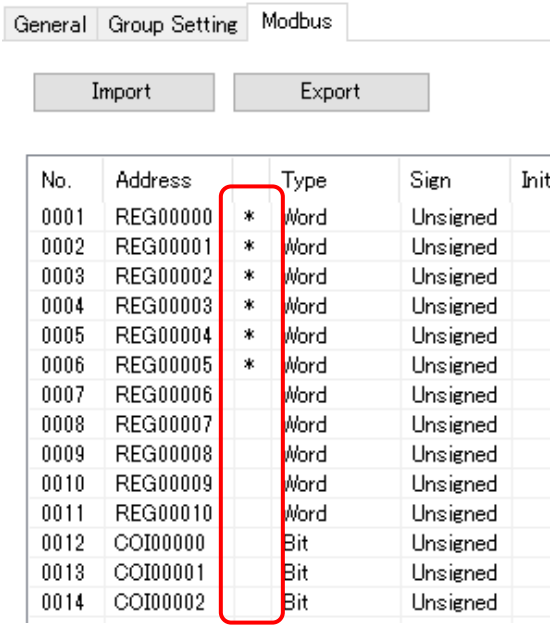


5. Please specify the waiting time for the “Normal Section”.
The unit is millisecond. It is the time to wait after reading the device value in the “Normal Section” once.

* This setting is not the synchronization time. The actual synchronization time is proportional to the number of memories set in the normal section. (Because reading starts after waiting time elapses, it will be more than the set value)



"*" is added to the address set in the priority section.



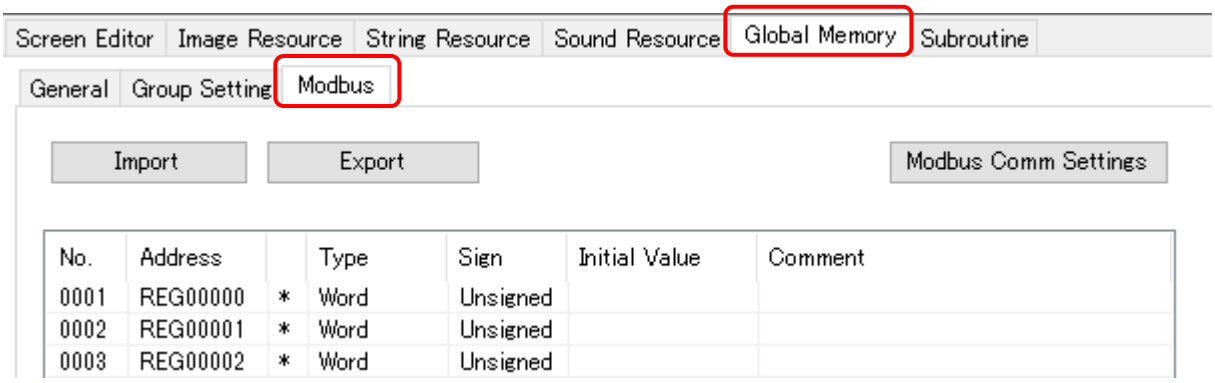
No.	Address		Type	Sign	Init
0001	REG00000	*	Word	Unsigned	
0002	REG00001	*	Word	Unsigned	
0003	REG00002	*	Word	Unsigned	
0004	REG00003	*	Word	Unsigned	
0005	REG00004	*	Word	Unsigned	
0006	REG00005	*	Word	Unsigned	
0007	REG00006		Word	Unsigned	
0008	REG00007		Word	Unsigned	
0009	REG00008		Word	Unsigned	
0010	REG00009		Word	Unsigned	
0011	REG00010		Word	Unsigned	
0012	COI00000		Bit	Unsigned	
0013	COI00001		Bit	Unsigned	
0014	COI00002		Bit	Unsigned	

3.1.4 Export Modbus Settings

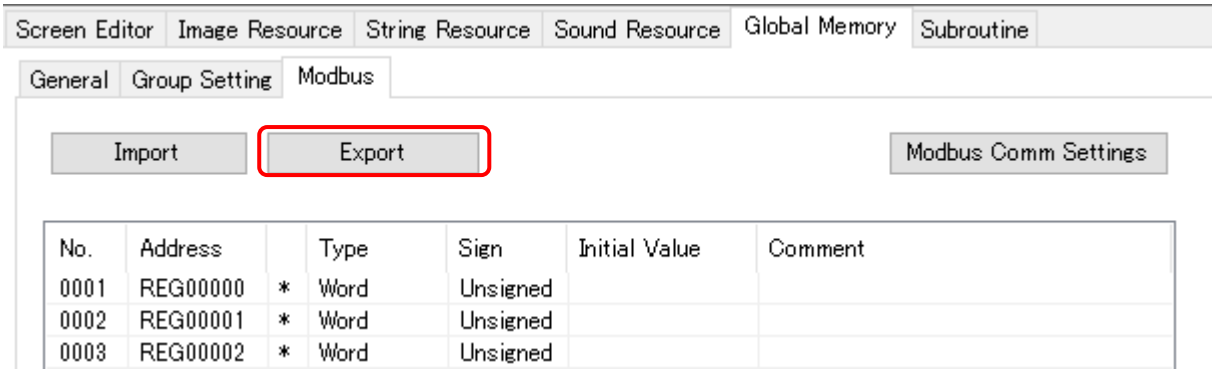
Modbus memory settings and Modbus communication settings can be exported to a csv format file.

* “On Change Value” event actions set in Modbus memory are not exported.

1. Display the “Modbus” tab in the “Global Memory” tab.



2. Click the "Export" button.



3. The csv format file is saved in the specified location.

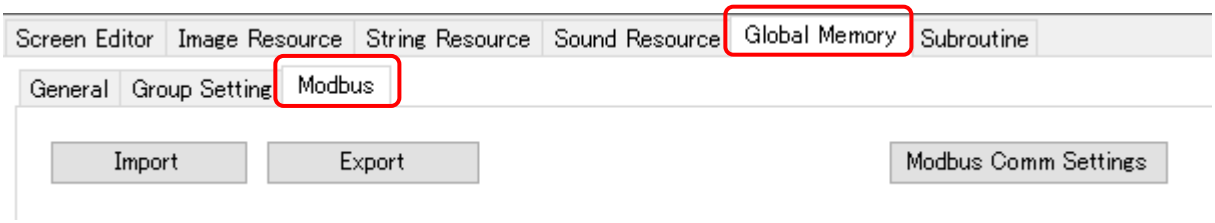
The exported csv file can be edited.
For details, refer to "4.4 Modbus setting file".

3.1.5 Importing Modbus settings

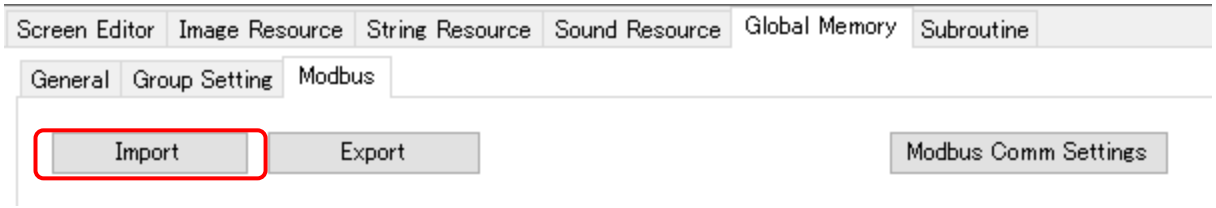
It is possible to import the exported Modbus configuration file.

* Please note that the current settings will be overwritten.

1. Display the “Modbus” tab in the “Global Memory” tab.

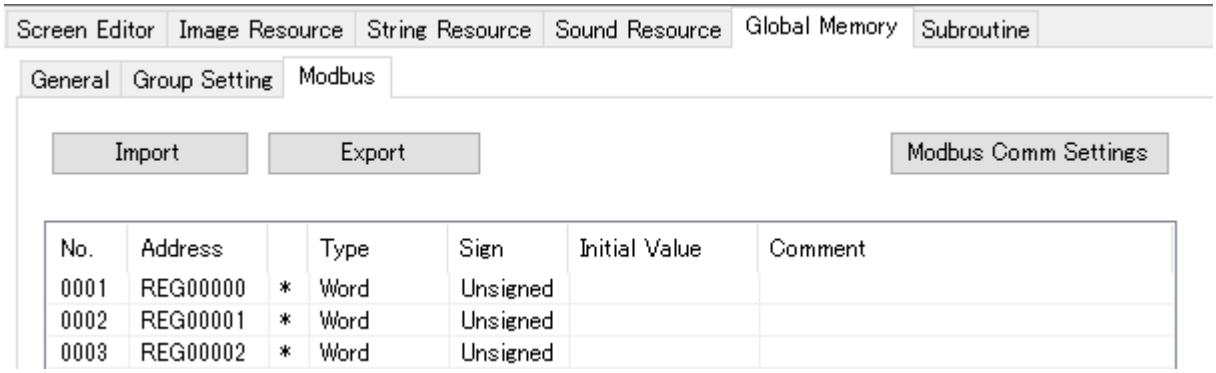


2. Click the "Import" button.



3. Select the Modbus configuration file.

4. Modbus memory settings and Modbus communication settings are set.



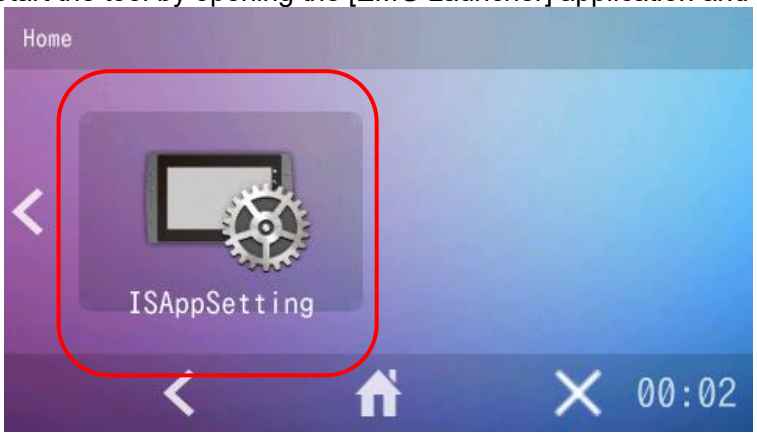
3.2 IS-APP startup settings

IS-APP configures communication settings with command line arguments at startup.
By using the "IS-APP Setting Tool", an IS-APP startup script with command line arguments set can easily be created.
If you do not use IS-APP setting tool, please refer to "4.3 IS-APP command line arguments".

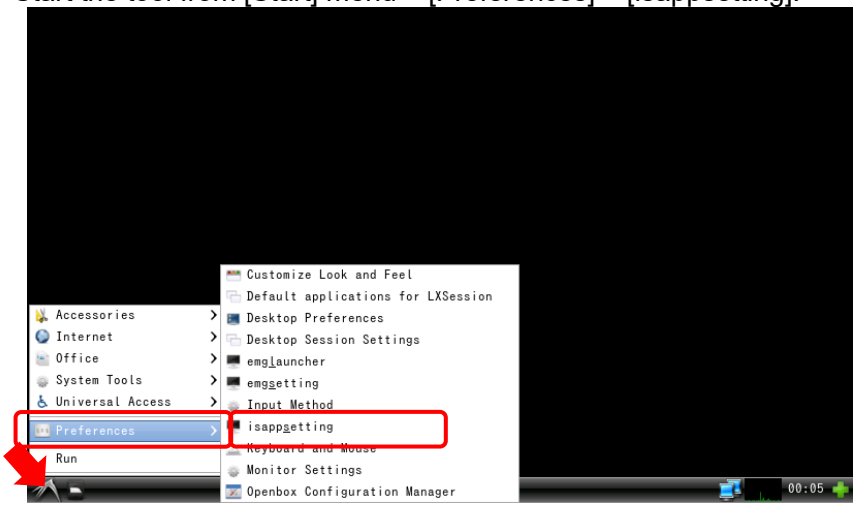
Specify the IP address of the Modbus device that will communicate.

1. Start the "IS-APP SETTING" tool on the EM Linux.

Method 1
Start the tool by opening the [EMG Launcher] application and clicking [ISAppSetting].



Method 2
Start the tool from [Start] menu > [Preferences] > [isappsetting].



Method 3
Start the tool by running the following program.

Executable file
/usr/bin/isapp_setting

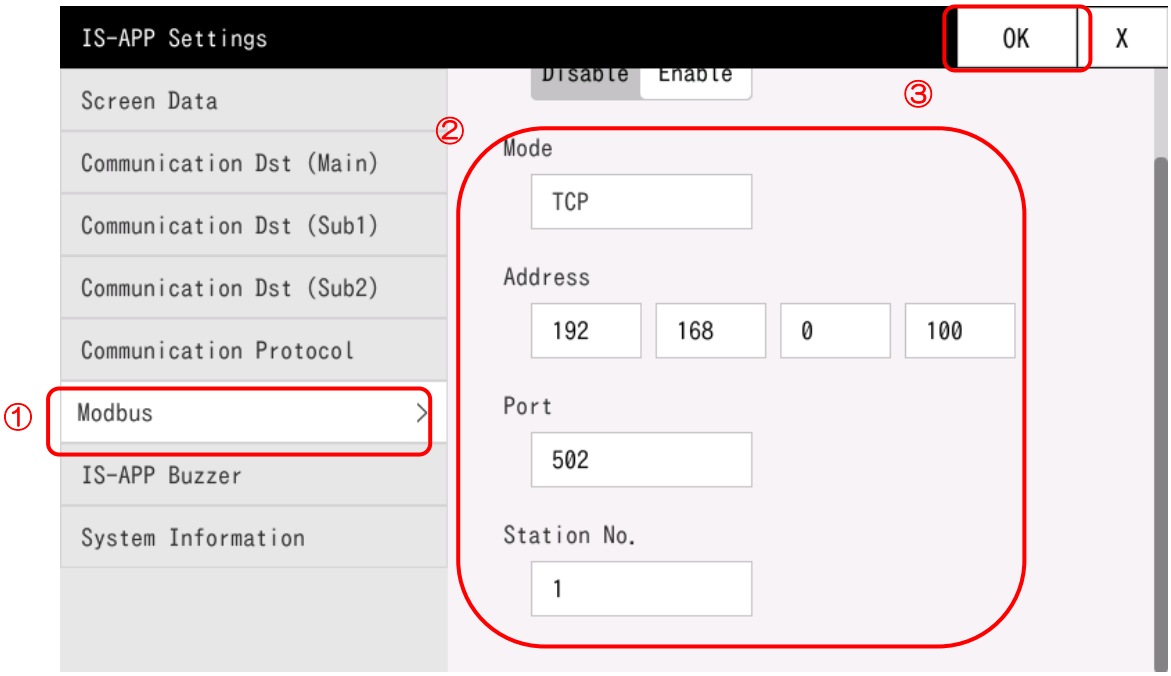
* When starting with a command, add -u to the start argument.

For detailed usage, please refer to the separate "IS-APP Startup Guide".

2. Set the following communication destination and touch OK button.

For TCP:
Address, Port, Station No.

For RTU:
Communication Port, Station No., Type, Communication Speed (bps), Parity

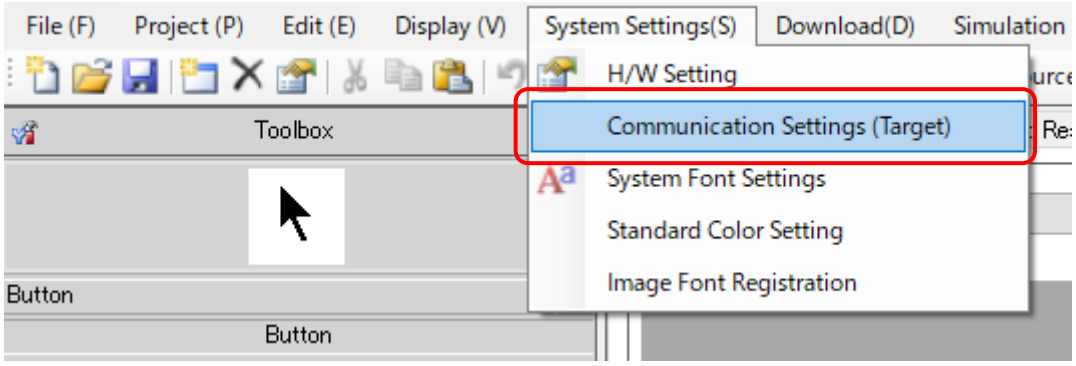


The above procedure creates the IS-APP startup script "/mnt/user/isapp_run.sh" in which the communication destination of the Modbus device is set.

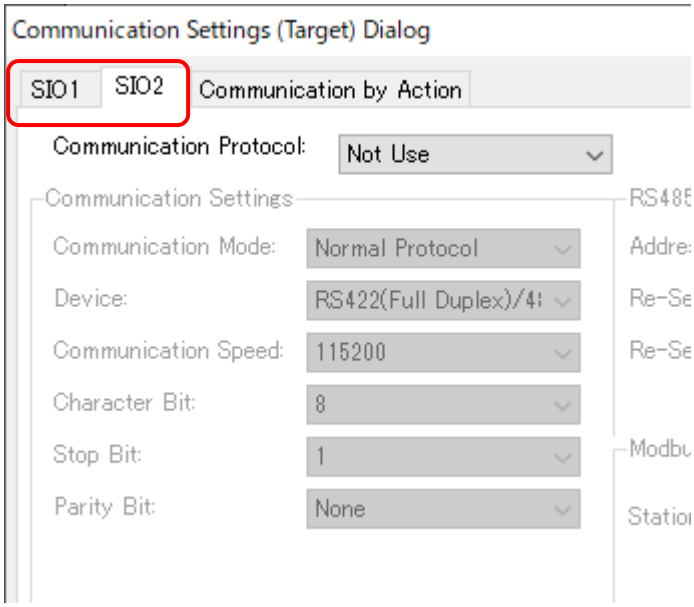
3.3 IS731 Communication Settings

For IS731 series, communication settings are made from InfoSOSA Builder.

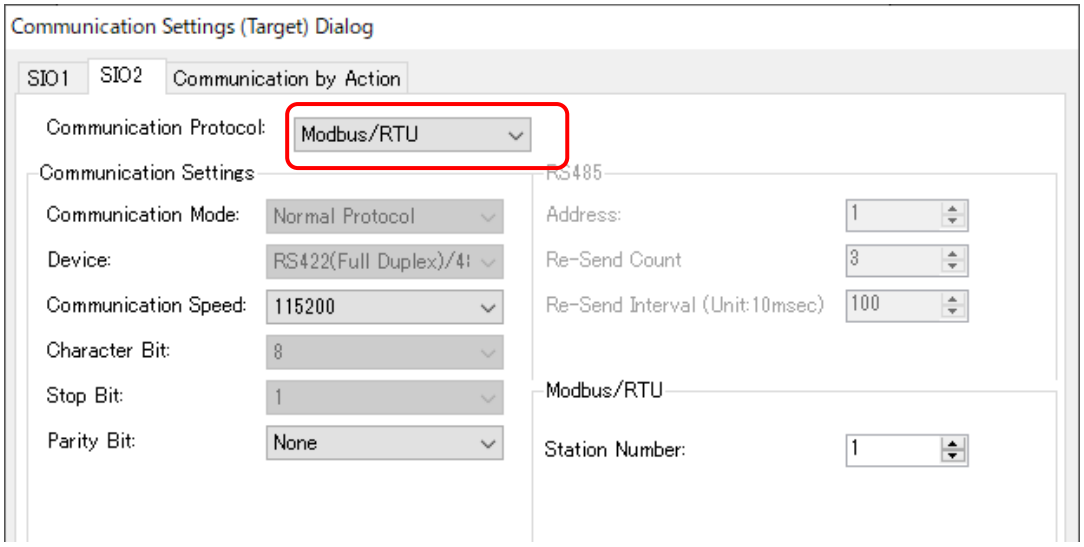
1. Open “System Settings” → “Communication Settings (Target)”.



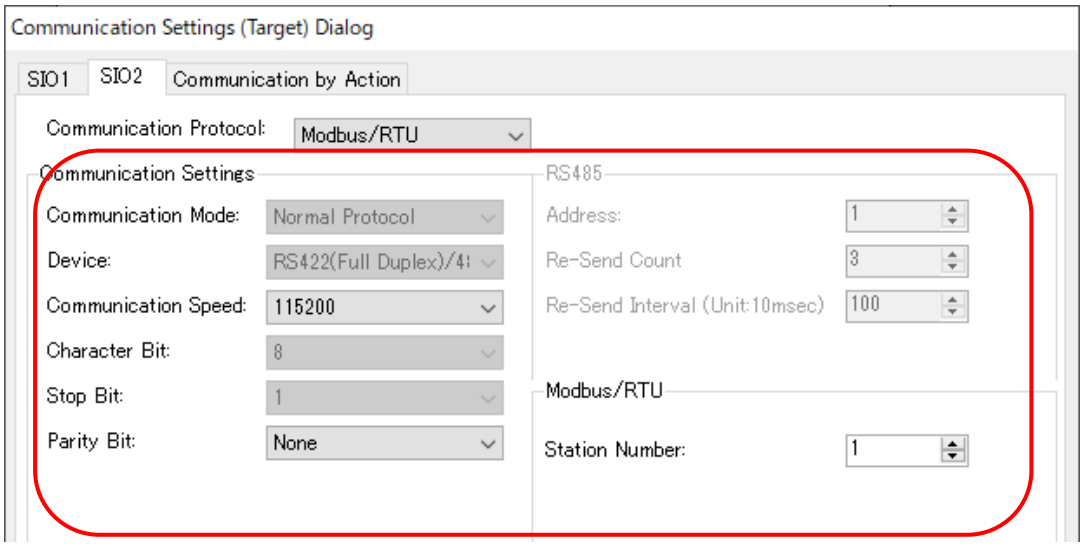
2. Select the tab (SIO1 or SIO2) of the port you want to use.



3. Select "Modbus/RTU" as the “Communication Protocol”.



4. Set the "Communication Speed", "Parity Bit", and "Station Number" according to the Modbus device that communicates.



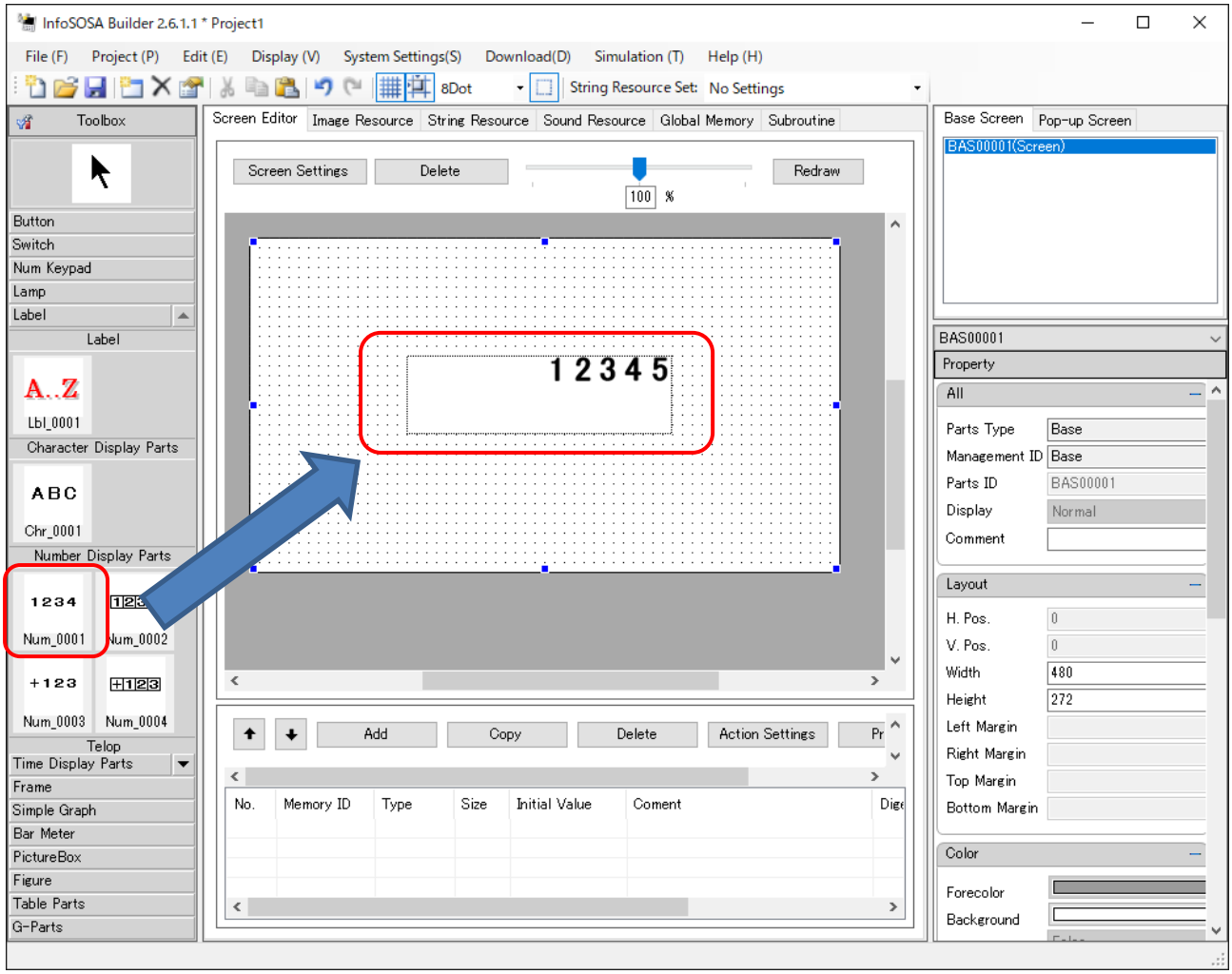
3.4 How to Read and Write Modbus Device Values

The value of the Modbus memory is automatically synchronized with the device value of the Modbus device.

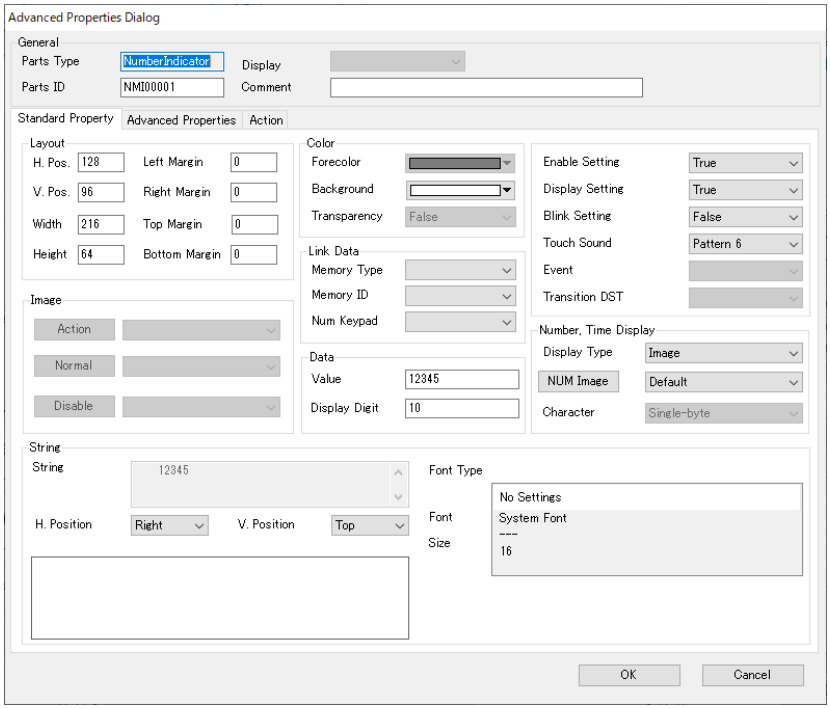
3.4.1 Display Device Values of Modbus Devices

Device value can be displayed by linking the Modbus memory to the numeric display component.

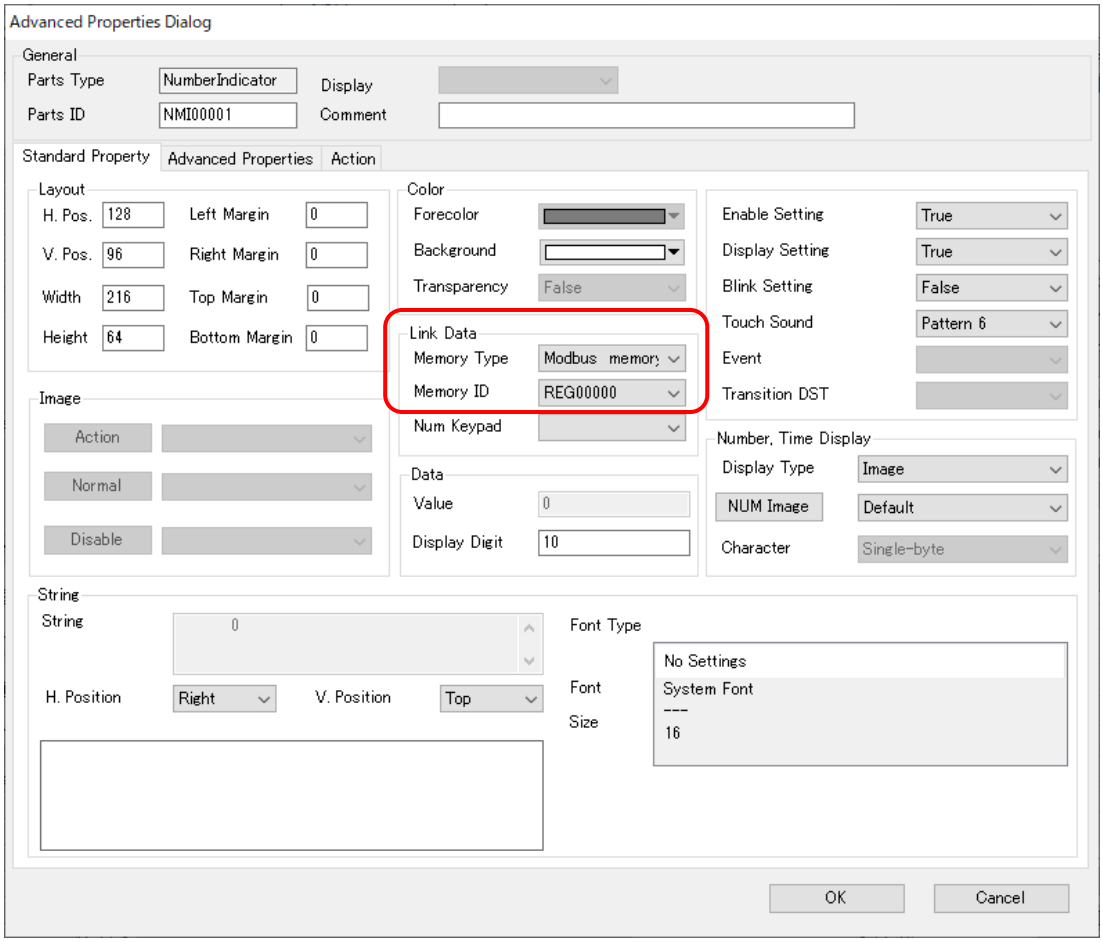
1. Place a number display part from the toolbox by drag and drop.



2. Open the detailed properties of the numerical display part. Double click to open.



3. Specify Modbus memory for “Link Data”.



item	value
Memory Type	Modbus memory
Memory ID	REG00000

With the above settings, this number display parts displays the value of the Modbus address "00000" of the register.

The memory ID is the address of the Modbus device to be displayed.

When the first three characters are “REG”, it indicates "register" and when it is “COI”, it indicates "coil".
The next five characters indicate "Modbus address (decimal notation)".

Example:

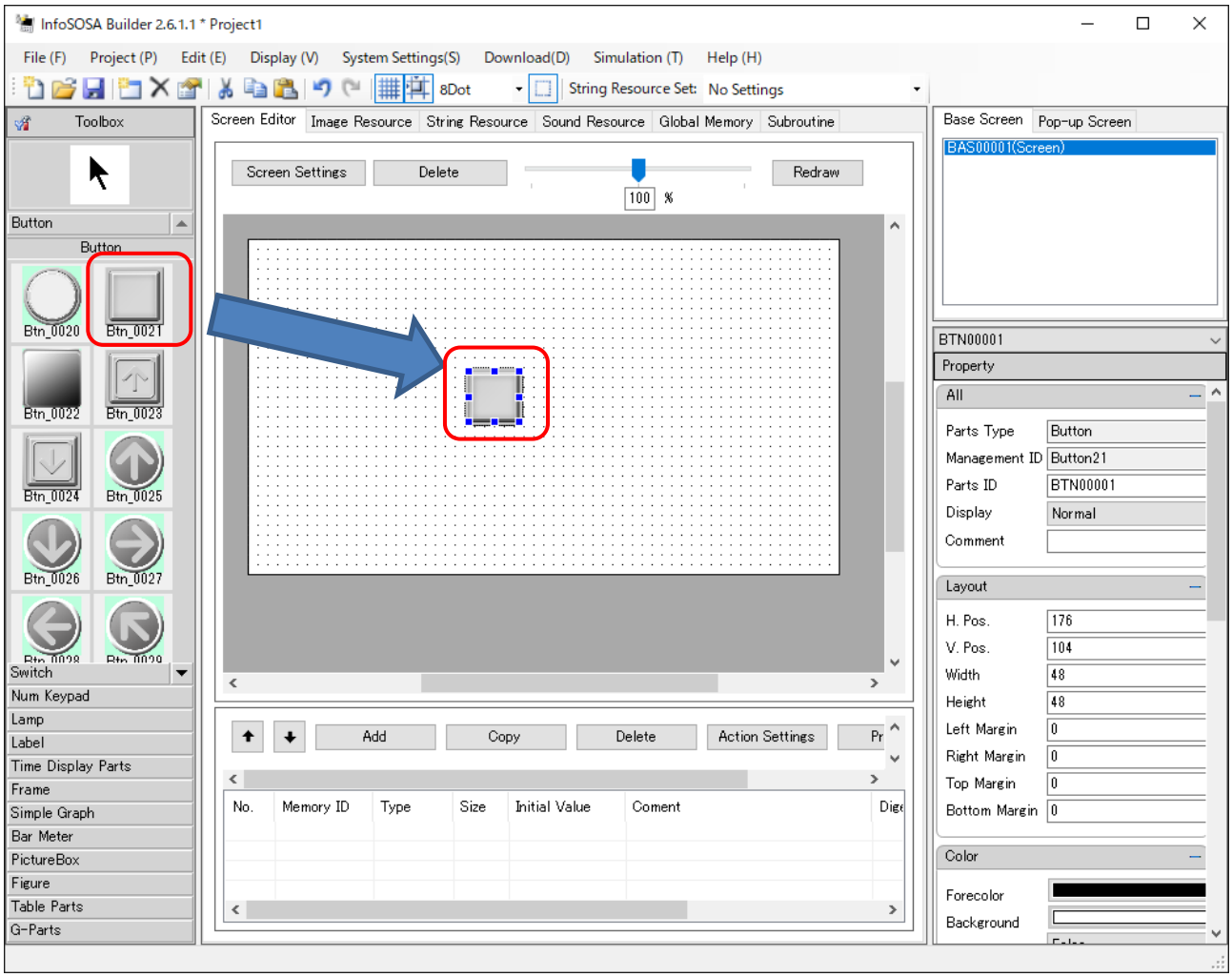
Memory ID	Modbus address (decimal notation)
REG00010	Modbus address (register) 00010
REG00250	Modbus address (register) 00250
COI00010	Modbus address (coil) 00010

* For the correspondence between the Modbus address and the register of the Modbus device, check the specifications of your Modbus device.

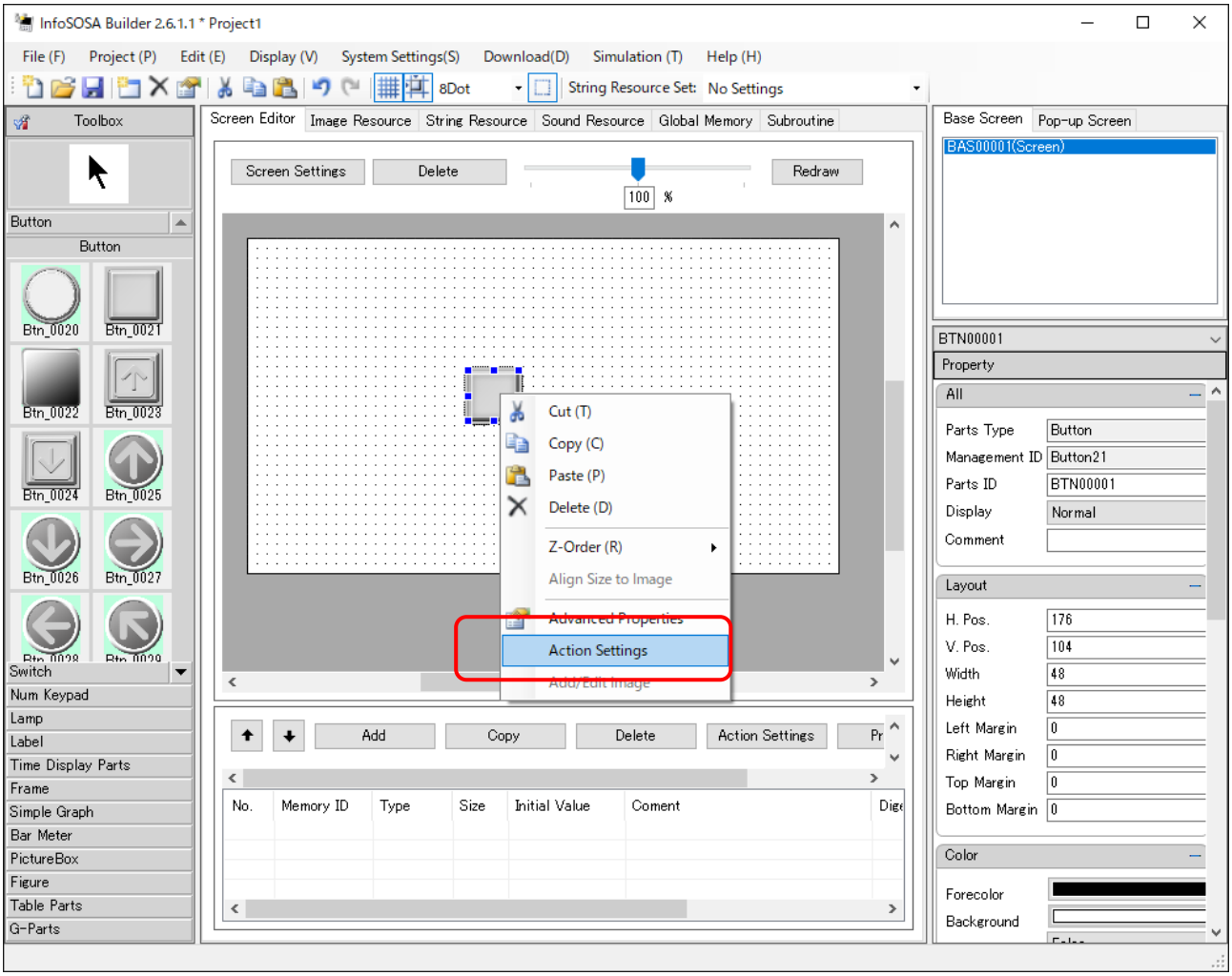
3.4.2 Write Value to Modbus Device

By setting a value to the Modbus memory, the value is automatically written to the Modbus device.

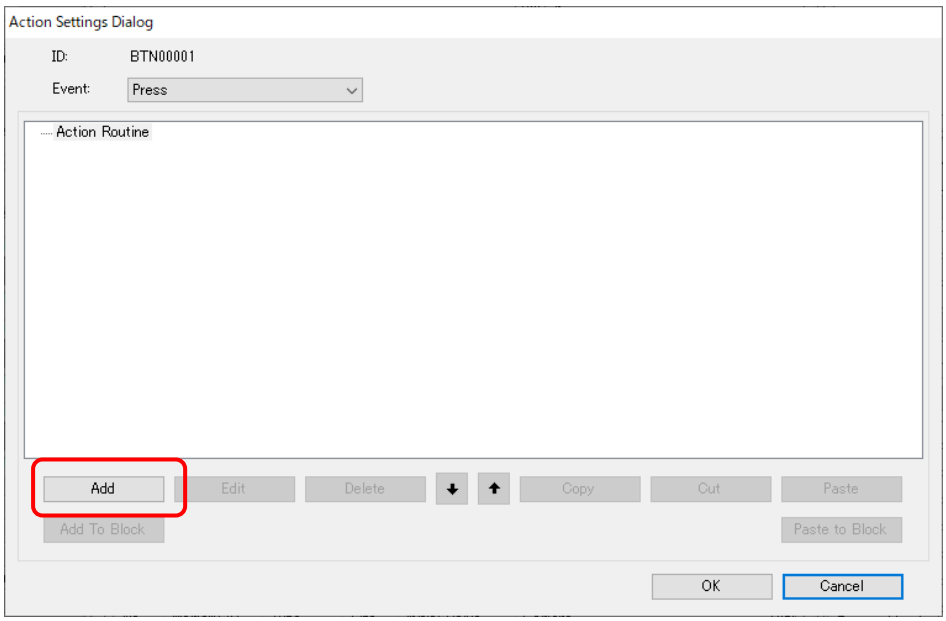
1. Place a button by drag and drop.



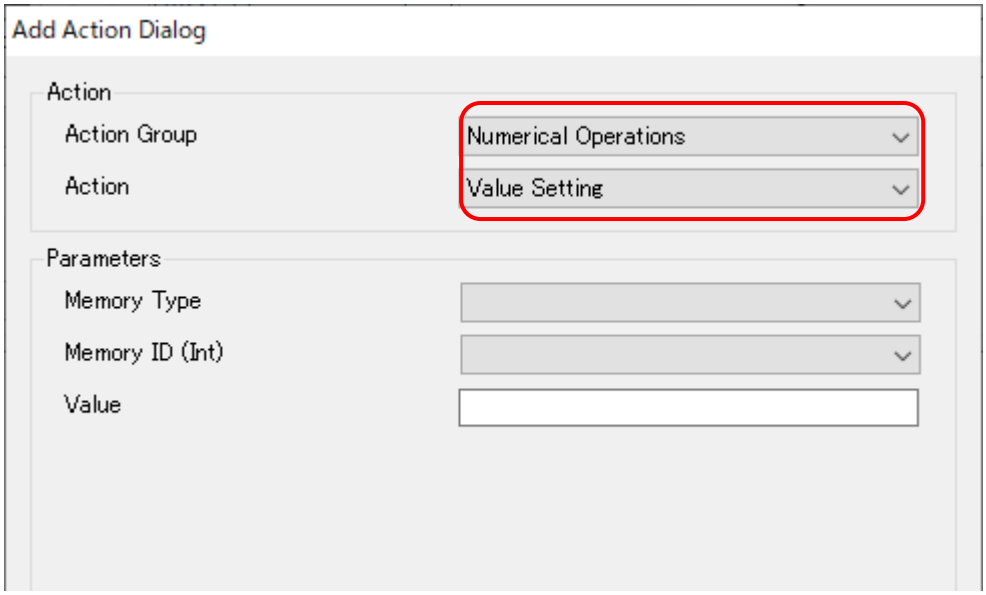
2. Right-click the button and choose “Action Settings” from the menu that appears.



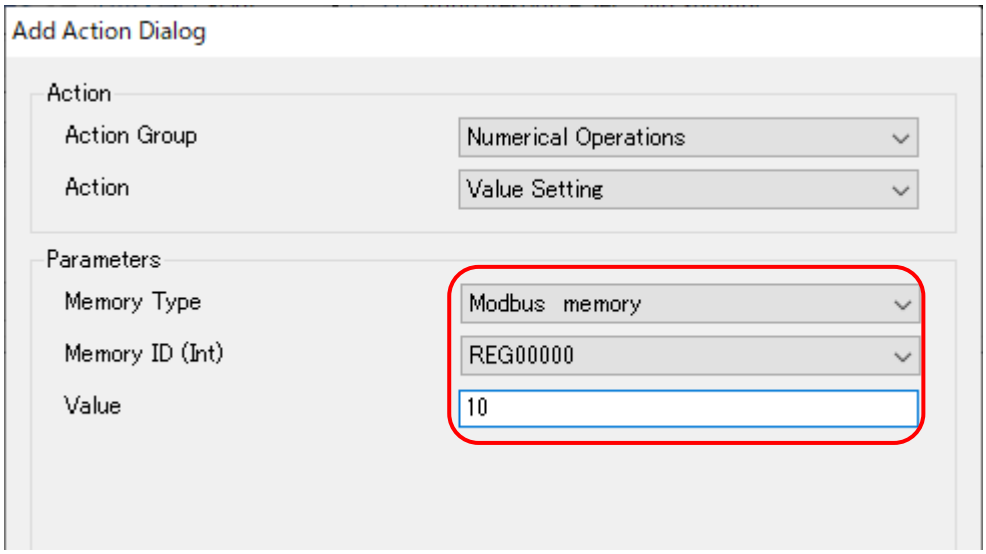
3. Click “Add”.



4. Select "Numerical Operation" for Action Group and "Value Setting" for Action.



5. Enter "Modbus memory" for Memory Type, "REG00000" for Memory ID(Int), and "10" for Value.



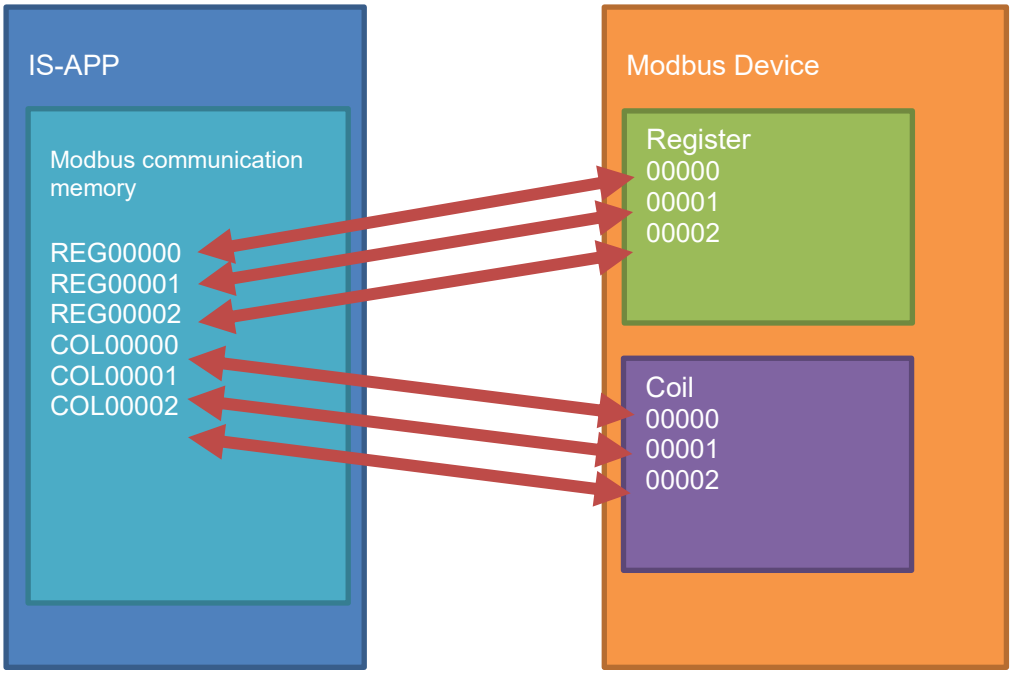
When the button is pressed with the above settings, 10 is written to the Modbus address “00000” of the Modbus device.

When value is set to Modbus memory, it automatically synchronizes.
* For the correspondence between the Modbus address and the register of the Modbus device, check the specifications of your Modbus device.
* If values are continuously written to the same Modbus memory in a short period of time, only the final value may be written to the Modbus device in order to reduce the load.

4. Reference

4.1 Modbus communication memory

The memory for Modbus communication is automatically synchronized with the value of the specified address in both directions.



The value on the Modbus device side is always read, and the memory for Modbus communication displays the value of the specified address on the Modbus device side.
If you set a value in the memory for Modub communication with an action, etc., the value will be written to the Modbus device side.

Note

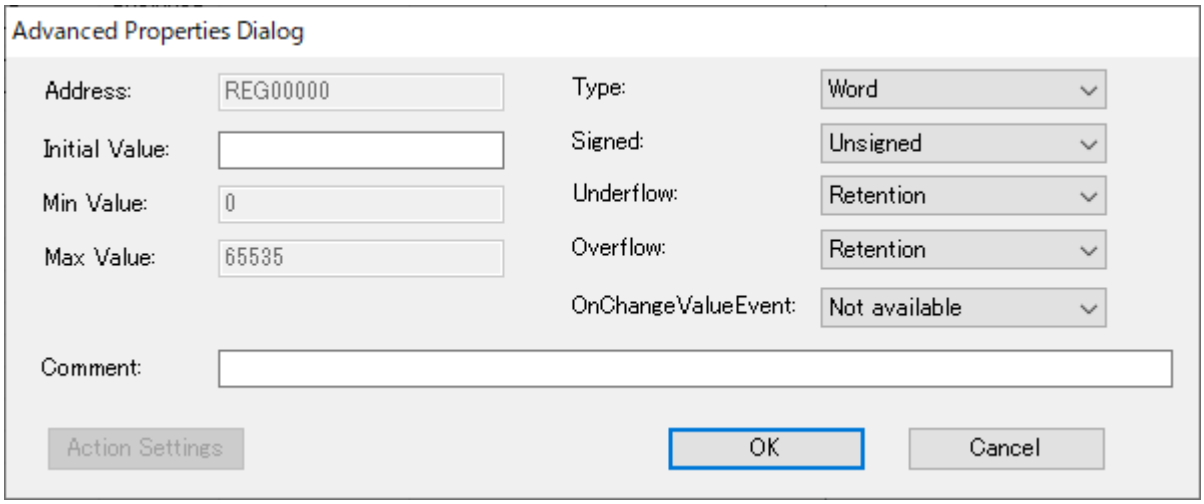
The more registered the memory for Modbus communication, the more time it takes to reflect. If the reflection is slow, you can specify the memory area to be synchronized with priority.

When the value of the memory for Modbus communication is changed, Read is stopped and Write processing is given priority.

If values are continuously written to the same Modbus memory in a short period of time, only the final value may be written to the Modbus device in order to reduce the load.

4.1.1 Property

This section describes the properties of memory for Modbus communication.



Parameter	Description
Address	If the first 3 characters are “REG”, it means "Register", and if “COI”, it means "Coil". The following 5 characters indicate the Modubs address (decimal number). The address value on the Modubs device side is synchronized with this memory. For the correspondence between the Modbus address and the register address of the device, check the specifications of the Modbus compatible device you are using. One address is 2 bytes.
Type	For registers (REG), you can select from "Byte", "Word", and "DoubleWord". The default is "Word". Type / Size Byte*1 / 1byte Word / 2byte DoubleWord*2 / 4byte *1 When byte is set, the upper data (1 byte) of the Modbus device is ignored and acquired. *2 When double word is set, the next address is used as upper data (2 bytes). (If changed to double word, it will be automatically integrated with the memory at the next address.)

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td></td><td>In the case of a coil (COI), it is fixed to "Bit".
<table><tr><th>Type</th><th>Size</th></tr><tr><td>Bit</td><td>1bit</td></tr></table></td></tr>
<tr><td>Signed</td><td>Select whether to display the acquired data as "Signed" or "Snsigned".
The default is "Unsigned".
In case of double word, it is fixed as "Signed". In case of “Coil”, it is invalid.
<table><tr><th>Type</th><th>Signed</th></tr><tr><td>Byte</td><td>Signed/Unsigned</td></tr><tr><td>Word</td><td>Signed/Unsigned</td></tr><tr><td>DoubleWord</td><td>Unsigned</td></tr></table></td></tr>
<tr><td>Initial Value</td><td>Write the initial value set at the first communication to the Modbus device.
If it is not set, the data of the Modbus device will be acquired first.
The default is "Not set".</td></tr>
<tr><td>Min Value</td><td>The minimum value of the target Modbus memory is displayed.
It cannot be edited directly.
It is determined by the properties of "Type" and "Sign".</td></tr>
<tr><td>Max Value</td><td>The maximum value of the target Modbus memory is displayed.
It cannot be edited directly.
It is determined by the properties of "Type" and "Sign".</td></tr>
<tr><td>Underflow</td><td>Defines the operation when a value smaller than the minimum value of the target Modbus memory is set.
The default is "Retention".
<table><tr><th>Setting</th><th>Action</th></tr><tr><td>Retention</td><td>Saves value right before underflow occurs.</td></tr><tr><td>Loop</td><td>Underflow memory subtracted from maximum value will be the value of target memory.</td></tr><tr><td>Clip</td><td>Set minimum value to target memory.</td></tr></table></td></tr>
<tr><td>Overflow</td><td>Defines the operation when a value larger than the maximum value of the target Modbus memory is set.
The default is "Retention".
<table><tr><th>Setting</th><th>Action</th></tr><tr><td>Retention</td><td>Saves value right before overflow occurs.</td></tr></table></td></tr>
</table>

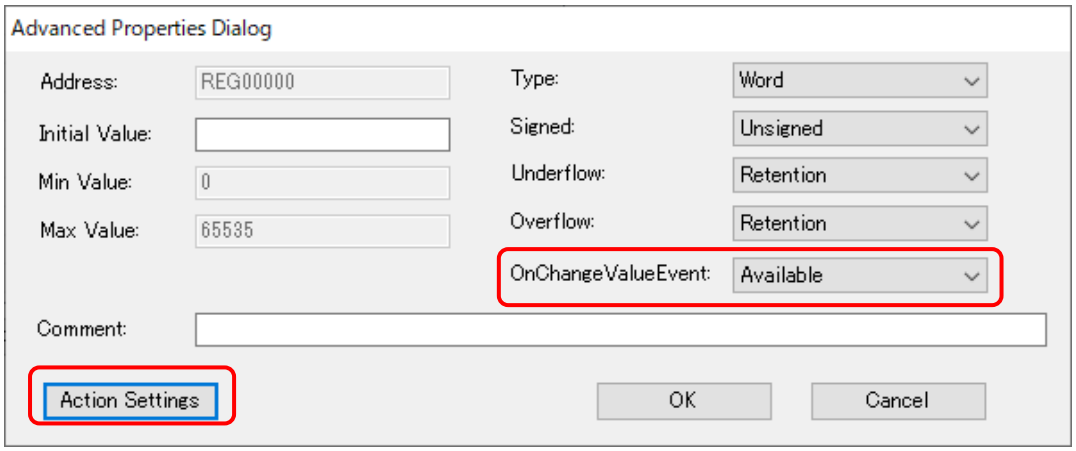
Parameter	Description
	<table><tr><td>Loop</td><td>Overflow memory added to minimum value will be the value of target memory.</td></tr><tr><td>Clip</td><td>Set maximum value to target memory.</td></tr></table>
OnChangeValueEvent	You can enable the “On Change Value” event and set the action. Arbitrary processing can be executed when the value of the target Modbus memory changes.
Comment	You can freely enter notes.

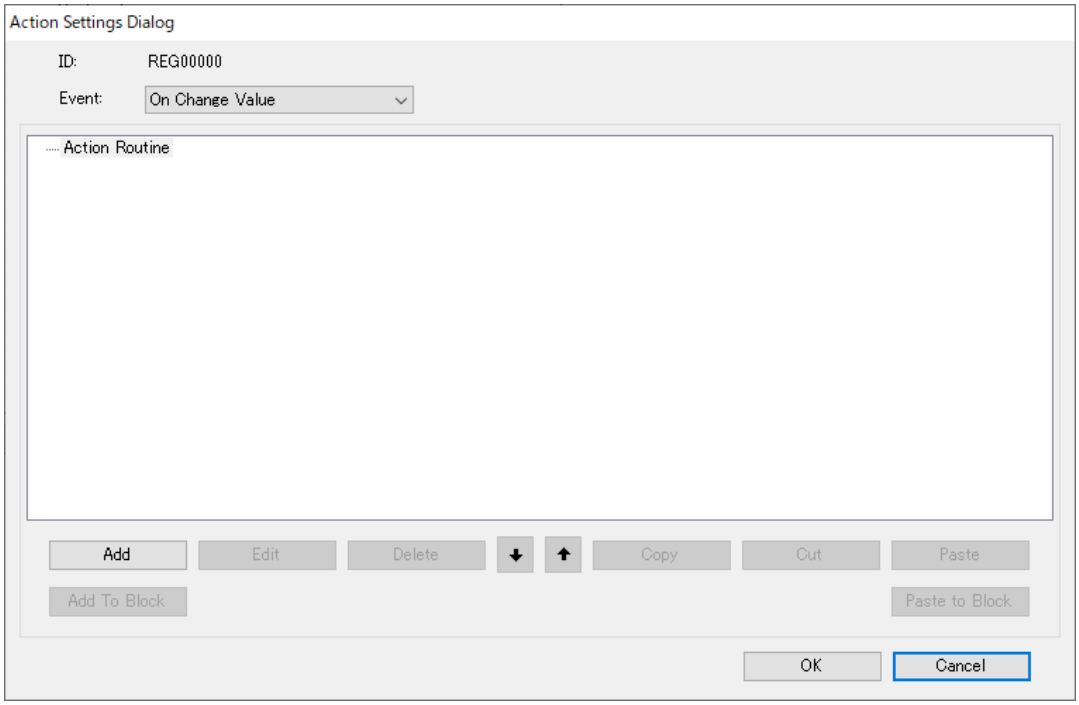
4.1.2 On Change Value Event

By using the value change event, by changing the value on the Modbus device side, you can execute the processing “Action” according to the value on InfoSOSA side.
For example, it is possible to change the display screen to a screen according to the value by changing the value of a specific register.

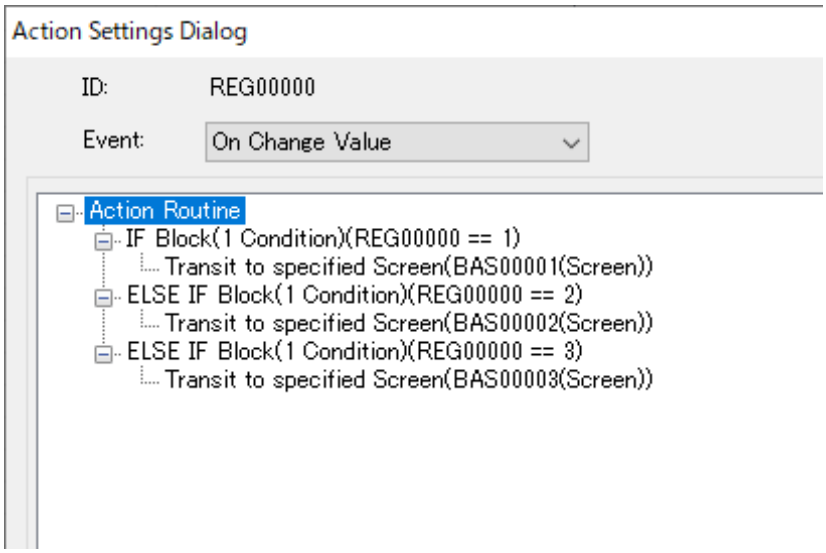
Setting the "On Change Value Event" property of the target memory to "Available" enables "Action setting".

Event ID
ON_CHANGE





By setting in this way, it is possible to switch the displayed screen according to the value of "REG00000".



Note

If you change the value of the memory in which the value change event is set at high speed, the value change event may occur repeatedly, which may result in performance, so please be careful to make the optimum change frequency.

When changing the value of the memory that set the value change event in the value change event, please be careful not to cycle.

4.1.3 Host Communication (InfoSOSA Protocol)

You can also access the “Modbus memory” with a host communication command.

The host communication command is a function to set or obtain a value in InfoSOSA using the InfoSOSA dedicated protocol, not the general-purpose Modbus protocol. It is mainly used for communication with the microcomputer boat.

Item	Value
Affiliation ID	@GLBMEM
Memory ID	Address
Property	VALUE

To access the Modbus memory "REG00000" with InfoSOSA protocol, use the following command.

[Set]

```
PA01,@GLBMEM.REG00000.VALUE,50
```

[Get]

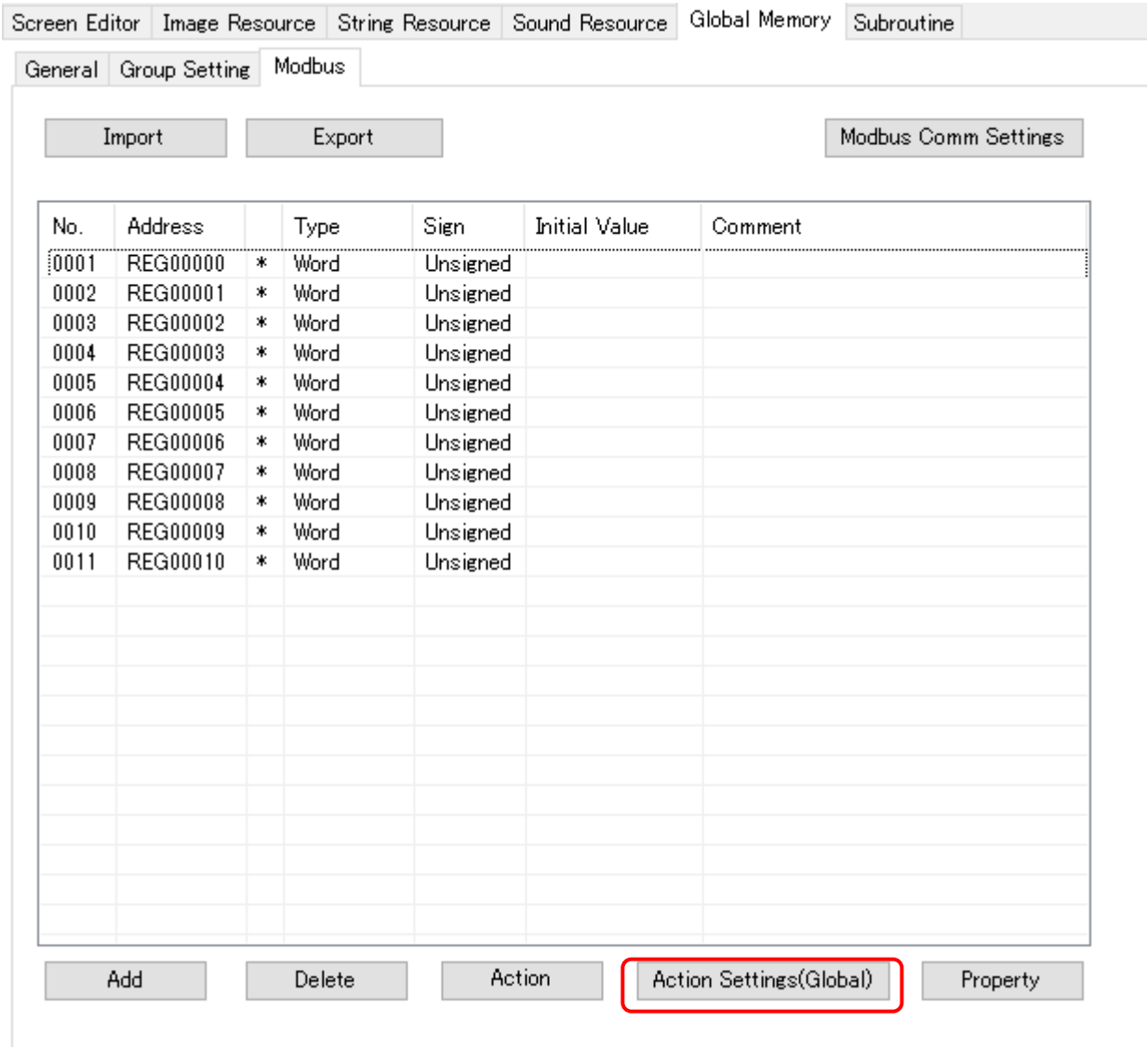
```
PA02,@GLBMEM.REG00000.VALUE
```

For details on host communication, refer to the chapter "Host Communication" in the separate "InfoSOSA Reference Manual".

4.2 Events for Modbus communication

Describes the details of events for Modbus device communication.

You can set the processing (“Action”) to be executed when the Modbus communication times out and the processing (“Action”) when the communication is restored from the "Action Setting (Global)" button on the "Modbus" tab.

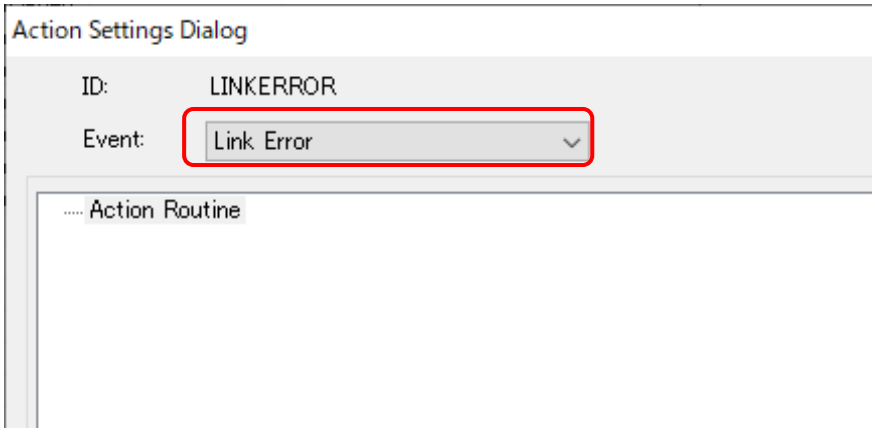


4.2.1 Link Error

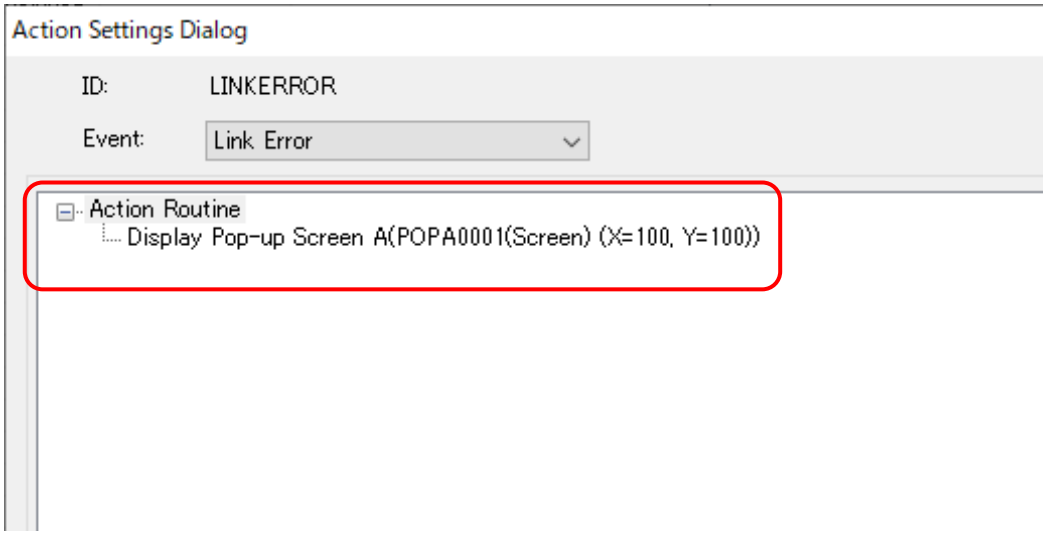
By switching the event to "Link Error", you can set the process (“Action”) to be executed when Modbus communication is disconnected.

[Note]
The Modbus device should return the response command within 100ms after receiving the command from InfoSOSA. If the response command is delayed, a Link Error event will occur.

Event ID
LINKERROR



For example, with this setting, it is possible to display an error screen when a communication error occurs.



The error details can be obtained by referring to the following environment variables.

Environment Variables ID
LINK_ERR

The error code is set in this environment variable.

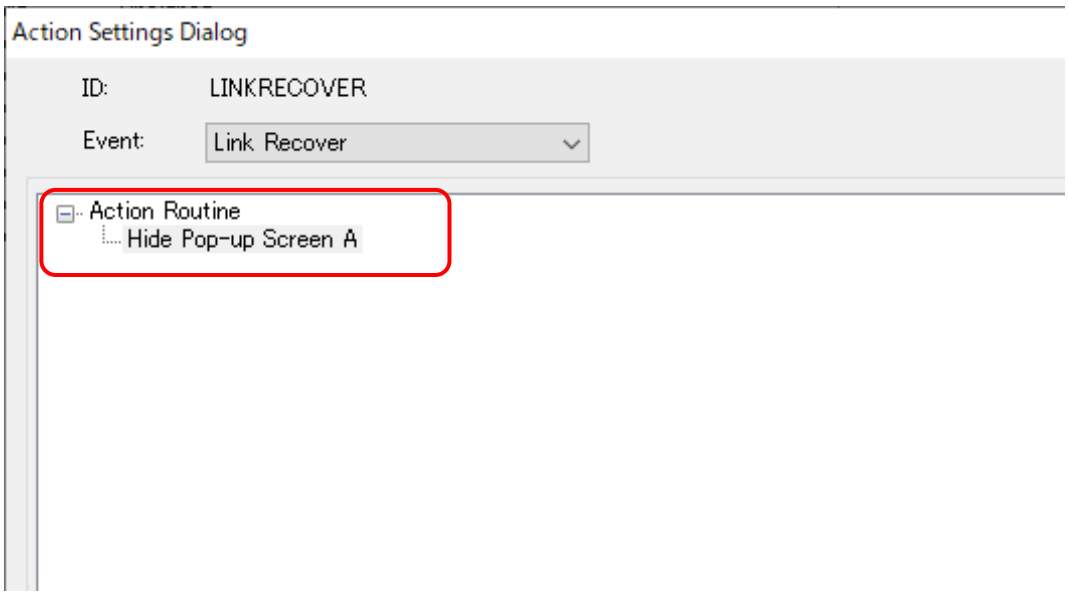
Error code	Protocol	Category	Error contents
1001	TCP	Communication error	TCP connection error, data communication timeout
1002	TCP	Communication error	TCP header error
1003	RTU	Communication error	Data communication timeout
1004	RTU	Communication error	CRC error
1101	Common	Data error	Receives error response from Modbus device Does not support function code
1102	Common	Data error	Receives error response from Modbus device The specified Modbus address is out of range
1103	Common	Data error	Receives error response from Modbus device The specified value is out of range
1104	Common	Data error	Receives error response from Modbus device Other errors
1901	Common	Other errors	The specified device file does not exist during RTU communication Other errors

4.2.2 Link Recover

When the event is switched to "Link Recover", it occurs when Modbus communication is restored after "Link Error" occurs.

Event ID
LINKRECOVER

For example, with this setting, it is possible to clear the error screen when the communication error is recovered.

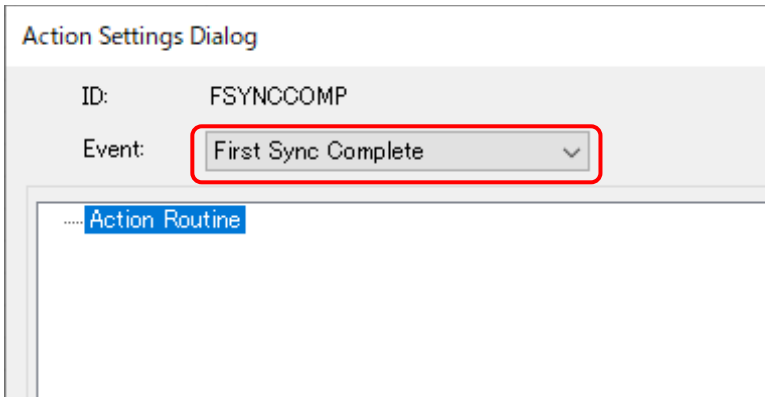


After communication is restored, the uncompleted change processing will be executed.

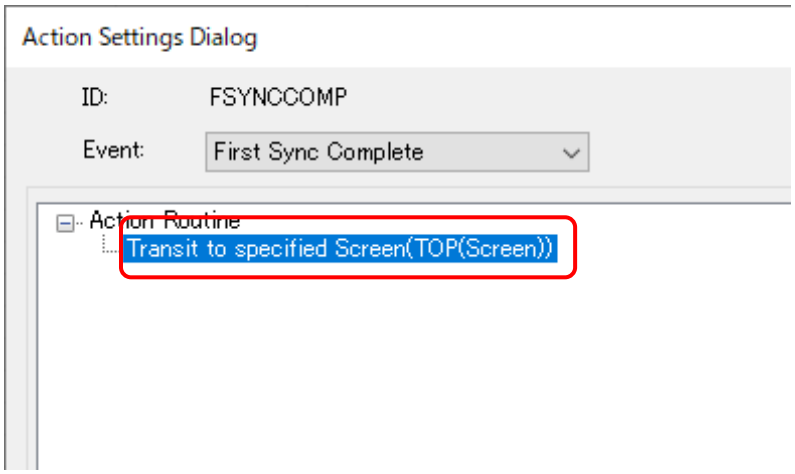
4.2.3 First Sync Complete

You can configure the processing (action) to be performed when the initial synchronization of Modbus memory is completed.
If no Modbus memory is registered, this event will not occur.
It does not occur when synchronization is completed for the second and subsequent times.

Event ID
FSYNCCOMP



It can be configured to display a loading screen at startup and automatically transition to the first screen after loading is complete.



4.3 IS-APP command line arguments

Modbus

Enable the Modbus protocol communication function of IS-APP.
If the Modbus protocol communication function is not enabled with this argument, communication with Modbus devices will not be performed.

Argument name

-u or --modbus

Format

For Modbus/TCP master: *default
-u tcp <communication address> <communication port> <station number>

For Modbus/RTU master:
-u rtu <device file> <station number> <communication standard> <communication speed> <parity>

Parameter	Contents
<communication address>	Specify the IP address of the Modbus device to communicate with. (Decimal) Default:192.168.0.100
<communication port>	Specify the communication destination port number. (Decimal) 0 to 65535 Default: 502
<device file>	Specify the serial port device file. Default: SIO2 device file is automatically set. SIO1：./dev/com1 SIO2：./dev/com2
<station number>	Specify the station number of the Modbus device to communicate with. (Decimal) 0 to 255 Default: 1
<communication standard>	Specify the communication standard of the serial interface. RS232C：rs232c RS422：rs422 RS485：rs485 *default
<communication speed>	Specify the communication speed. 4800: 4800bps

Parameter	Contents
	9600: 9600bps 19200: 19200bps 38400: 38400bps 57600: 57600bps 115200: 115200bps *default
<parity>	Specify the parity bit. none: No parity bit *default odd: Odd parity even: Even parity

example:

For Modbus/TCP master:

is_app –r /mnt/user/data –u tcp 192.168.0.100 502 1

For Modbus/RTU master:

is_app –r /mnt/user/data –u rtu /dev/com2 1 rs485 115200 none

Default

If omitted, the settings will be as follows.

Parameter	Contents
Enable/Disable	Disable

4.4 Modbus setting file

4.4.1 csv format (individual setting)

When exported, it will be saved in csv format.
Modbus memory settings can be set individually.

sample:

```
;ModbusType,ReadMaxNum,WriteMaxNum
RegisterMaxData,16,16
CoilMaxData,16,16
;PriorityAddress,StartAddress,EndAddress
PriorityRegister,0,1
PriorityCoil,0,1
;NormalAddress,WaitTime
NormalCommon,100
;ModbusType,Address,DataType,InitialValue,Signed,UnderFlow,OverFlow,Comment
Register,00000,Byte,10,Unsigned,Retention,Retention,state1
Register,00001,Word,,Unsigned,Loop,Loop,temperature
Register,00002,DWord,,Signed,Clip,Clip, Number of rotations
Coil,00000,Bit,1,Unsigned,Retention,Retention,Start
Coil,00001,Bit,,Unsigned,Retention,Retention,state2
Coil,00002,Bit,,Unsigned,Retention,Retention,Emergency stop
```

*Lines starting with a semicolon are comment lines.
*The character code is UTF-8.

1st line (maximum data count setting/register)

Parameter	Value	Explanation
ModbusType	RegisterMaxData	Fixed value
ReadMaxNum	1 to Maximum value of the connected Modbus device	Describe the maximum number of registers that can be read in one read process in decimal. The higher the value, the better the performance, but if the maximum value of the connected Modbus device is exceeded, communication will not be possible.
WriteMaxNum	1 to Maximum value of the connected Modbus device	Enter the maximum number of registers that can be written in one writing process in decimal. The higher the value, the better the performance, but if the maximum value of the connected Modbus device is exceeded, communication will not be possible.

2nd line (maximum data number setting/coil)

Parameter	Value	Explanation
ModbusType	CoilMaxData	Fixed value
ReadMaxNum	1 to Maximum value of the connected Modbus device	The maximum number of coils that can be read in one read process is described in decimal. The higher the value, the better the performance, but if the maximum value of the connected Modbus device is exceeded, communication will not be possible.
WriteMaxNum	1 to Maximum value of the connected Modbus device	Enter the maximum number of coils that can be written in one writing process in decimal. The higher the value, the better the performance, but if the maximum value of the connected Modbus device is exceeded, communication will not be possible.

3rd line (priority section setting/register)

Parameter	Value	Explanation
PriorityAddress	PriorityRegister	Fixed value
StartAddress	Values in the address range of the register to be registered from the 6th line onwards	Describe the start address and end address of the register set in the priority section. The value must be StartAddress ≤ EndAddress. If both Start Address and End Address are not input, all registers will be in the normal section.
EndAddress		

4th line (priority section setting/coil)

Parameter	Value	Explanation
PriorityAddress	PriorityCoil	Fixed value
StartAddress	Values in the address range of the coil to be registered from the 6th line onwards	Describe the start address and end address of the coil set in the priority section. The value must be StartAddress ≤ EndAddress. If both Start Address and End Address are not input, all coil will be in the normal section.
EndAddress		

5th line (normal section setting, common to register/coil)

Parameter	Value	Explanation
NormalAddress	NormalCommon	Fixed value
WaitTime	1 to 3599999	Enter the waiting time in the normal section. The unit is millisecond. *This setting is not the synchronization time. The actual synchronization time is proportional to the number of memories set in the normal section. (Because reading starts after waiting time elapses, it will be more than the set value)

6th line and later (Modbus memory setting)

Parameter	Value	Explanation
ModbusType	Register Coil	To create a communication memory for registers Please define as "Register". When defined as "Register", communication to Modbus device is performed with the following function code. Read:0x03 Write:0x10 When creating a communication memory for coils Please define as "Coil". When defined as "Coil", communication to Modbus device is performed with the following function code. Read:0x01 Write:0x0F
Address	0 to Maximum value of the connected Modbus device	Describe the Modbus address to synchronize this memory in decimal.
DataType	Byte Word DWord Bit	When ModbusType is "Register", you can select from "Byte", "Word", and "DWord". Type / Size Byte*1 / 1byte Word / 2byte DWord*2 / 4byte *1 When byte is set, the upper data (1 byte) of the Modbus device is ignored and acquired. *2 When double word is set, the next address is used as upper data (2 bytes). (If changed to double word, it will be automatically integrated with the memory at the next address.)

<table>
<tr><th>Parameter</th><th>Value</th><th>Explanation</th></tr>
<tr><td></td><td></td><td>In the case of a coil (COI), it is fixed to "Bit".
<table><tr><th>Type</th><th>Size</th></tr><tr><td>Bit</td><td>1bit</td></tr></table></td></tr>
<tr><td>InitialValue</td><td>Within the range of the type specified in DataType</td><td>Set the initial value of this memory.
When the initial value is set, the initial value set at the first communication is written to the Modbus device.
If not set, the data of the Modbus device is acquired first。</td></tr>
<tr><td>Signed</td><td>Unsigned
Signed</td><td>Select whether to display the acquired data as "Signed" or "Unsigned".
<table><tr><th>Type</th><th>Signed</th></tr><tr><td>Byte</td><td>Signed／Unsigned</td></tr><tr><td>Word</td><td>Signed／Unsigned</td></tr><tr><td>DWord</td><td>Signed</td></tr><tr><td>Bit</td><td>Unsigned</td></tr></table></td></tr>
<tr><td>UnderFlow</td><td>Retention
Loop
Clip</td><td>Defines the behavior when a value smaller than the minimum value of this memory is set.
<table><tr><th>Setting</th><th>Action</th></tr><tr><td>Retention</td><td>Saves value right before underflow occurs.</td></tr><tr><td>Loop</td><td>Underflow memory subtracted from maximum value will be the value of target memory.</td></tr><tr><td>Clip</td><td>Set minimum value to target memory.</td></tr></table></td></tr>
<tr><td>OverFlow</td><td>Retention
Loop
Clip</td><td>Defines the behavior when a value larger than the maximum value of this memory is set.
<table><tr><th>Setting</th><th>Action</th></tr><tr><td>Retention</td><td>Saves value right before overflow occurs.</td></tr><tr><td>Loop</td><td>Overflow memory added to minimum value will be</td></tr></table></td></tr>
</table>

Parameter	Value	Explanation
		the value of target memory. Clip: Set maximum value to target memory.
Comment	Up to 256 characters	You can write any description about this memory. The character code will be UTF-8.

4.4.2 ini format (collective setting)

This is an import-only format. You can register multiple Modbus memories at once.
The initial value of Modbus memory registered in ini format is registered with the following default values.

Item	Value
DataType	Word
Signed	Unsigned
InitialValue	Not set
UnderFlow	Retention
OverFlow	Retention
Priority section setting	None

Sample:

```
;Section definition
[Section]
SectionNum=3
SectionName1=Register1
SectionName2=Register2
SectionName3=Coil1

;Section1(Register)
[Register1]
Address=0,10
MaxWriteData=16
MaxReadData=16

;Section2(Register)
[Register2]
Address=30,40

;Section3(Coil)
[Coil1]
Address=0,15
MaxWriteData=16
MaxReadData=16
```

* Lines starting with a semicolons are comment lines.

[Section]

Parameters	Value	Description
SectionNum	1-	Specify the total number of sections. Modify this value when adding or removing sections.
SectionName1- Subscripts must be numbered sequentially from 1 Example: SectionName1 SectionName2 SectionName3	Register1- Coil1- Subscripts must be numbered sequentially from 1 Example: Register1 Register2 Coil1 Coil2	Define the section name. When creating a register communication memory, define it as "Register". When defined as "Register", communication with Modbus devices is made using the following function codes: Read:0x03 Write:0x10 When creating a coil communication memory, define it as "Coil". When defined as"Coil", communication with Modbus devices is made using the following function codes: Read:0x01 Write:0x0F

[Register1]

Parameters	Value	Description
Address	[Start Modbus address],End Modbus address]	Enter the start Modbus address and end Modbus address of the register to be registered in decimal. * Up to 2000 memory (registers and coils total) for communication with Modbus devices can be registered, including global memory.
MaxWriteData	1- Maximum number of Modbus devices to be connected	Describe the maximum number of registers that can be written in one writing process in decimal. The performance improves as the value is increased, but communication will not be possible if the maximum value of the connected Modbus device is exceeded.
MaxReadData	1- Maximum number of Modbus devices to be connected	Describe the maximum number of registers that can be written in one reading process in decimal. The performance improves as the value is increased, but communication will not be

Parameters	Value	Description
		possible if the maximum value of the connected Modbus device is exceeded.

[Register2]

Parameters	Value	Description
Address	0,10	If the Modbus addresses to be registered are not consecutive, describe them as separate sections. MaxWriteData and MaxReadData of "Register" are set in the [Register1] section, so they are not required in the second section. * Up to 2000 memory (registers and coils total) for communication with Modbus devices that can be registered, including the global memory.

[Coil1]

Parameters	Value	Description
Address	[Start Modbus address],[End Modbus address]	Describe the start Modbus address and end Modbus address of the coils to be registered in decimal. * Up to 2000 memory (registers and coils total) for communication with Modbus devices that can be registered, including the global memory.
MaxWriteData	1- Maximum number of Modbus devices to be connected	Describe the maximum number of coils that can be written in one writing process in decimal. The performance improves as the value is increased, but communication will not be possible if the maximum value of the connected Modbus device is exceeded.
MaxReadData	1- Maximum number of Modbus devices to be connected	Describe the maximum number of coils that can be written in one reading process in decimal. The performance improves as the value is increased, but communication will not be possible if the maximum value of the connected Modbus device is exceeded.

Inquiries

If you have any questions, feel free to contact us.

By E-mail

North South America area

technical-global@dush.co.jp

Asia Pacific area

technical-global-asia@dush.co.jp

Europe, Middle East, Africa area

technical-global-eu@dush.co.jp

FAQ

www.dush.co.jp/english/support/faq/

9th Edition May, 2023
DMC Co., Ltd.
Office hours: 9:00 - 17:00 weekdays
(except Saturdays, Sundays, national holidays, and year-end and New Year holidays)
https://www.dush.co.jp/english/
This document is protected by copyright law. Photocopying, duplicating, reproducing, and modifying of this product or document in part or by whole is prohibited.