InfoSOSA™ Series

IS-APP Startup Guide

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

Introduction

Thank you for purchasing DMC products.

This manual describes the features of the InfoSOSA Series IS-APP, provides a tutorial, as well as explains functions unique to the InfoSOSA Series IS-APP.

The InfoSOSA Series IS-APP is referred to as both InfoSOSA and IS-APP.

Target audience

- ✓ For those considering the IS-APP.
- ✓ For those new to the IS-APP.
- ✓ For those checking features and specifications of the IS-APP.

Target Version

This manual describes the following versions of InfoSOSA.

Some operations may differ depending on the version.

Please refer to "InfoSOSA ReleaseNote" for details.

InfoSOSA Builder	2.7.1
IS-APP	2.4.1
IS-API	1.3.2

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 ~

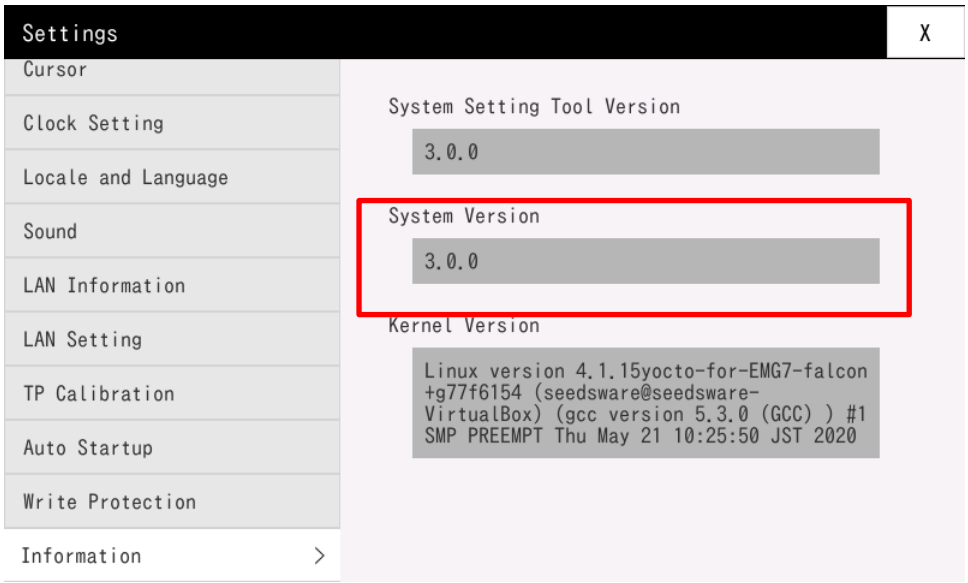


Table of Contents

1. About the IS-APP

Chapter Contents

1.1 IS-APP Overview

The IS-APP is an InfoSOSA application.

This application is for running screen data designed in the InfoSOSA Builder screen drawing software on the EM Series of DMC panel computers.

IS-APP simplifies creating HMI that displays on the panel computer.

NOTE

◆ About InfoSOSA

DMC microcontroller touch panel display unit. Comes equipped with the InfoSOSA Builder screen drawing software, a product for designing the HMI.

Includes IS-API, the application programming interface (API) for interacting with IS-APP.

Using [IS-API], the following can be done.

- Using the API you could do things like create C/C++ applications to change the IS-APP screen.
- Touch a button in IS-APP to trigger a function in your C/C++ application.

In so doing, you could use the C/C++ application for host communication and running the control part of your system, and the HMI part can be displayed by IS-APP with the screens created using [InfoSOSA Builder].

1.2 Reference Documents

Documents related to this document are as follows. Refer to the manual that meets your purpose.

IS-APP Startup Guide (this manual)

Refers to this document.
This manual describes the features of the IS-APP, provides a tutorial, as well as explains functions and specifications unique to the IS-APP.

Target audience

- ✓ For those considering the IS-APP.
- ✓ For those new to the IS-APP.
- ✓ For those checking features and specifications of the IS-APP.

InfoSOSA Reference Manual

This manual describes InfoSOSA functions and specifications.

Target audience

- ✓ For those checking details of InfoSOSA functions and specifications.
- ✓ For those checking communication specifications between InfoSOSA and host devices.

InfoSOSA Builder Operation Manual

This manual describes how to operate the InfoSOSA Builder.

Target audience

- ✓ For those setting up and operating InfoSOSA Builder.
- ✓ For those interested in convenient InfoSOSA Builder uses.

Host Communication Tester Manual

This manual describes host communication tester operations.

* Host communication tester is a software for communicating with the InfoSOSA with a PC instead of a host device.

Target audience

- ✓ For those testing communication with InfoSOSA without a host device.
- ✓ For those debugging the host device and checking communication commands.
- ✓ For those setting up and operating the host communication tester.

InfoSOSA ReleaseNote

Differences depending on the version of InfoSOSA are described.

Target audience

- ✓ For InfoSOSA users who are considering upgrading to a newer version.

The following are not included in the InfoSOSA development kit.

It can be downloaded from our homepage.

https://www.dush.co.jp/english/download/documents/

EM Series Software Development Manual

Describes the development process of software that runs on the EM Series.

Target audience

- ✓ For those developing software to run on the EM Series.

EM Series Software Tool Manual

Describes the tools installed in the EM series.

Target audience

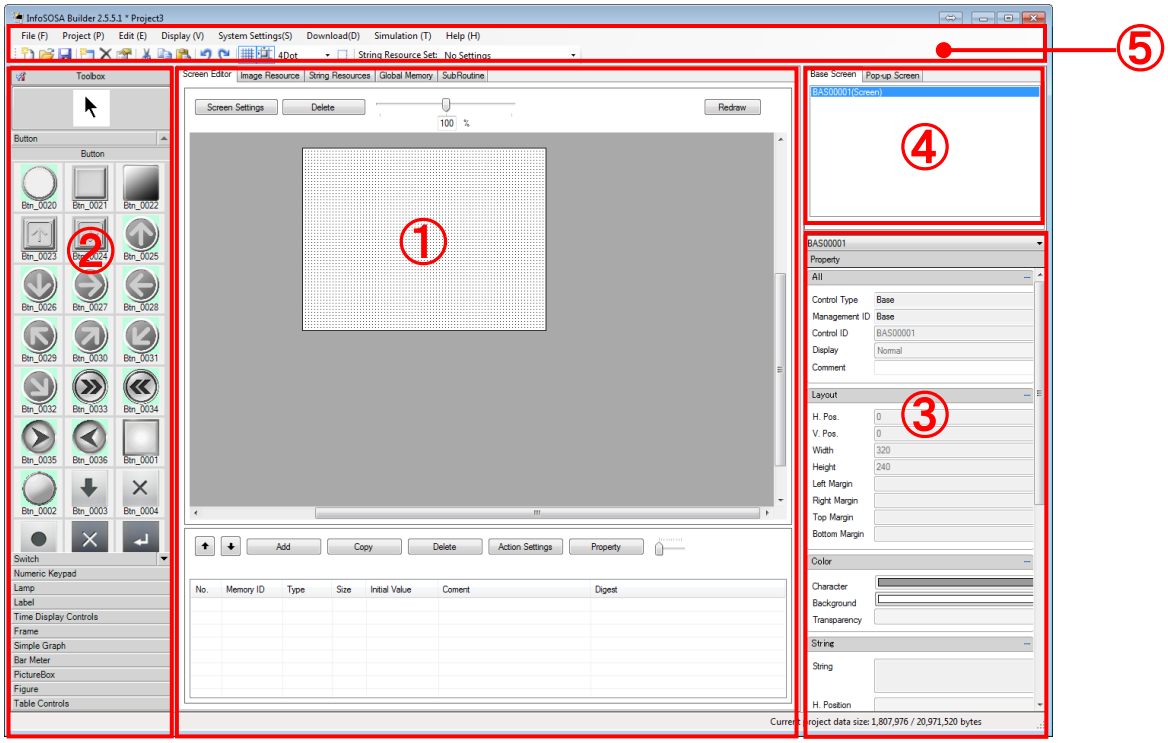
- ✓ For those who use EM Series.

1.3 About InfoSOSA Builder

InfoSOSA Builder is software for creating screens that display on InfoSOSA.

Arrangements of parts and operation setting can be done with easy mouse operations such as "drag-and-drop" and selecting commands from pull-down menus. There are no complicated operations and you do not need to write source code for settings.

InfoSOSA Builder is composed of the following screens.



① Drawing Area

Area mainly for creating screens, consisting of multiple tabs.
We will be using the [Screen Editor] tab for this demo.
Please refer to the "Reference Manual" or the "Operation Manual" for details on other tabs and how to work with them.

② Toolbox Area

Categorizes parts used to create a screen.
Arrange the parts in the drawing area by simple "drag-and-drop" operations.

③ Property Area

Displays the properties of parts selected in the screen editor.
Set a variety of settings such as layout, size, and color of parts.

④ Screen List Area

Displays the names of screens being created.
Click the name of the screen you want to work on.

⑤ Menu Toolbar Area

Use the menus to call a screen and other operations such as saving and creating files.

1.4 Install InfoSOSA Builder

This chapter describes the installation of the InfoSOSA Builder.

1.4.1 System Requirements

InfoSOSA Builder can run on a PC that satisfies the following conditions.
Make sure the PC you will be installing with the Builder satisfies the system requirements below.

Item	Contents
OS	Microsoft® Windows® 10 (64 Bit) or Microsoft® Windows® 11 (64 Bit) *
Framework/ Runtime	Microsoft® .NET Framework 3.5 Microsoft® .NET Framework 4.7 Microsoft® Visual C++ 2015 Runtime
Processor (or equivalent)	1GHz or faster
Memory	4GB or more (Recommended: 8GB or more)
HDD	850MB or more free space (Recommend:1GB or more)
Display	1024×768 or higher, True Color (32bit) preferred

* There is no warranty associated with operating in a virtual environment.
* Windows 11 compatibility applies to version 2.7.1 and later.

1.4.2 Start Installation

Caution
Please make sure that there is enough free space (more than 850MB) in your PC's hard disk before installation.

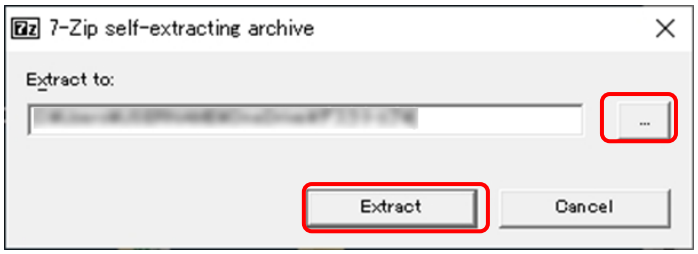
Caution
Please install with Administrator rights.

Caution
Installation occurs for all users.

Caution
In case of builder version upgrade, please uninstall the builder on PC in advance.

1. Please execute "software\builder\InfoSOSABuilder*.*.*.*.exe " in the download data.

2. A dialog for specifying the extraction destination is displayed. Click the [...] button, specify the extraction destination (desktop, etc.), and click the [Extract] button.



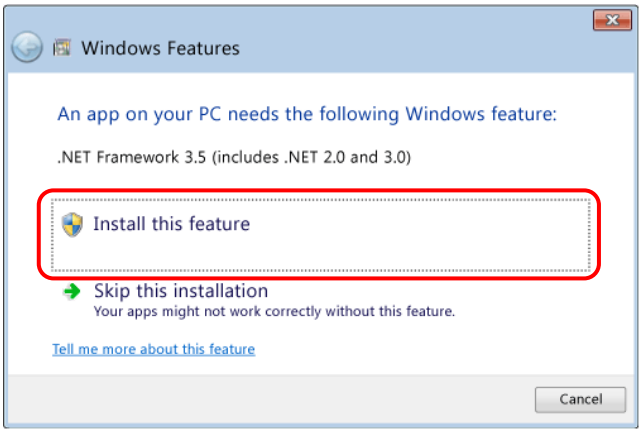
3. The "InfoSOSA Builder *. *. *. *" Folder is created in the extraction destination specified in 2. above.

4. Execute "Setup.exe" in the generated folder "InfoSOSABuilder *. *. *. *".

1.4.3 Install .NET Framework

To run InfoSOSA Builder, Microsoft® .NET Framework 3.5 or .NET Framework 4 is required.
If the PC you are using does not have it installed, before you install InfoSOSA Builder install .NET Framework.
When the InfoSOSA Builder installation screen is displayed, run Install InfoSOSA Builder.

.NET Framework 4.7 comes standard. If .NET Framework 3.5 is not installed, when the installer for the builder starts up a dialog box as shown below will appear. Select [Install this feature] to install .NET Framework 3.5.

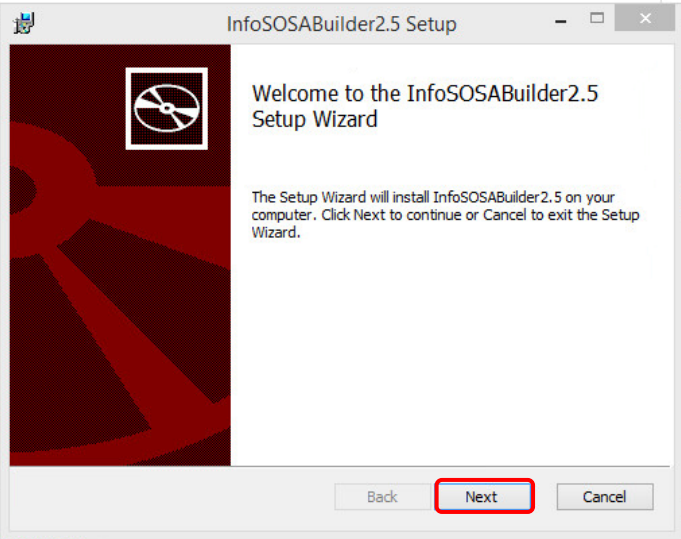


* As the .NET Framework installer uses Windows Update, an Internet connection is required.

* To install offline, you need to use the Windows installation media. (You cannot use the .NET Framework 3.5 installer bundled with the development kit data.)

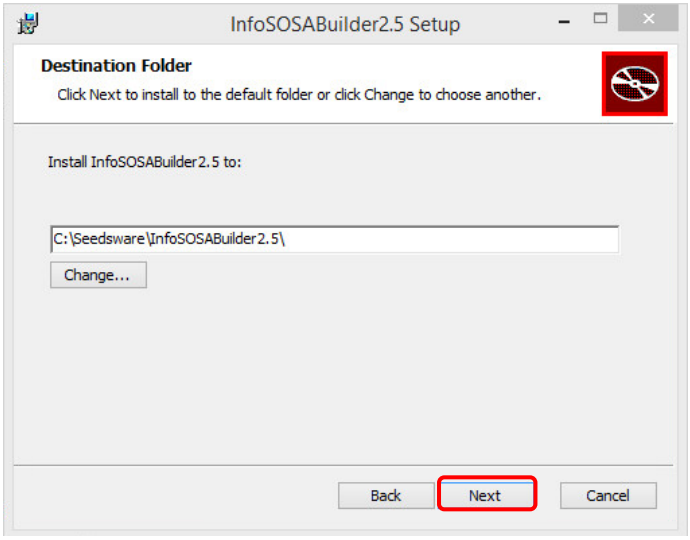
1.4.4 Install InfoSOSA Builder

1. When you start installing, the following dialog box appears.
Click [Next].

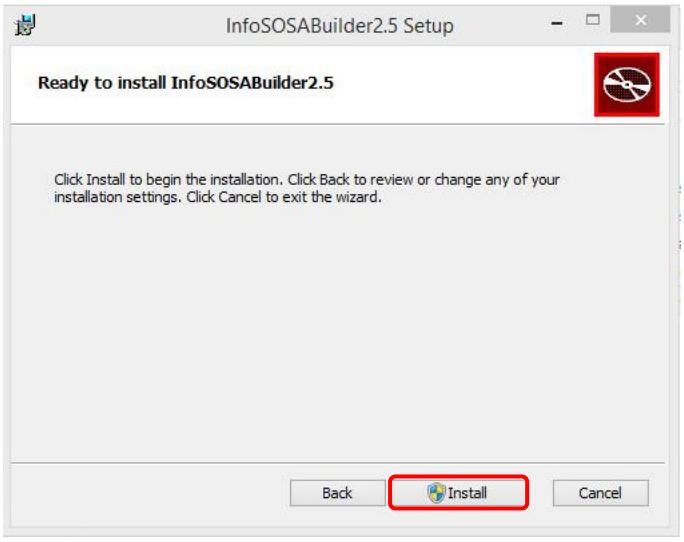


2. Review the license agreement, select the [I accept the terms in the License Agreement] check box and click [Next].

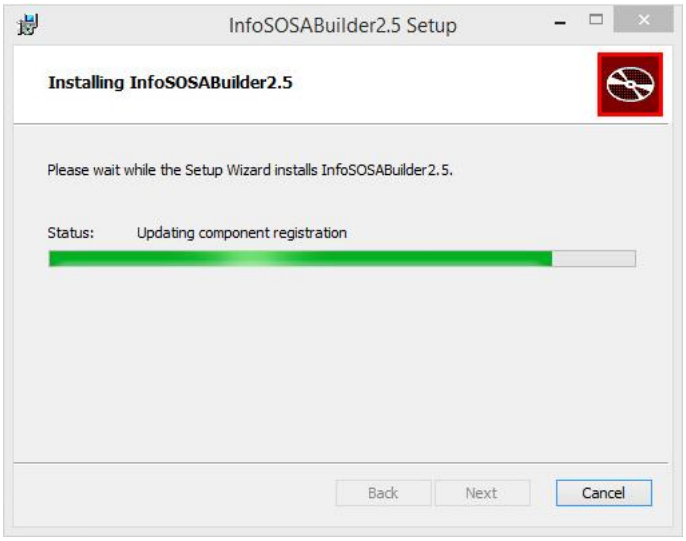
3. Specify the installation folder and click [Next].



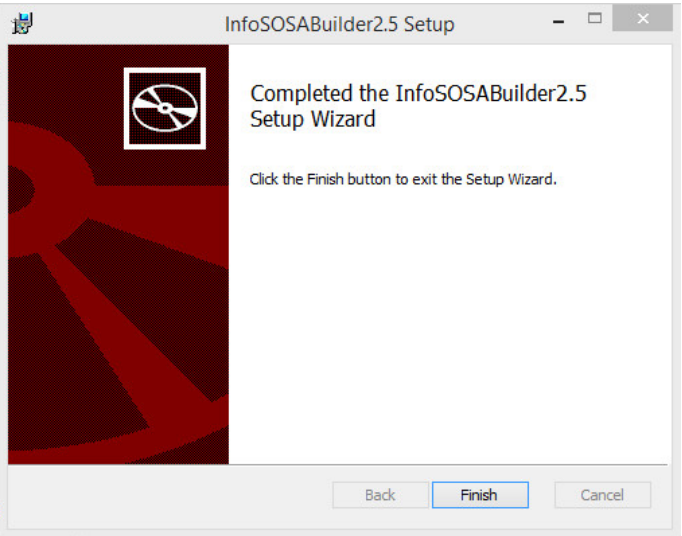
4. Click [Install].



5. Installation begins.



6. Installation is complete.



After installation is complete, a shortcut 

appears on your desktop.

1.4.5 Install Microsoft Visual C++ 2015 Runtime

If you get an error that "mfc140u.dll" is not found, follow the steps below to install "Microsoft Visual C++ 2015 Redistributable Package".

1. Access Microsoft's download page.

https://learn.microsoft.com/en-us/cpp/windows/latest-supported-vc-redist?view=msvc-170

2. Download both "X86" and "X64" of Microsoft Visual C++ 2015 Runtime.

X86	https://aka.ms/vs/17/release/vc_redist.x86.exe
X64	https://aka.ms/vs/17/release/vc_redist.x64.exe

3. Execute the downloaded "VC_redist.x64.exe" and "VC_redist.x86.exe" respectively to install.

1.5 Connection between EM series and PC

In order to transfer the project (screen data), the settings for connecting the EM series main unit to the PC are required.

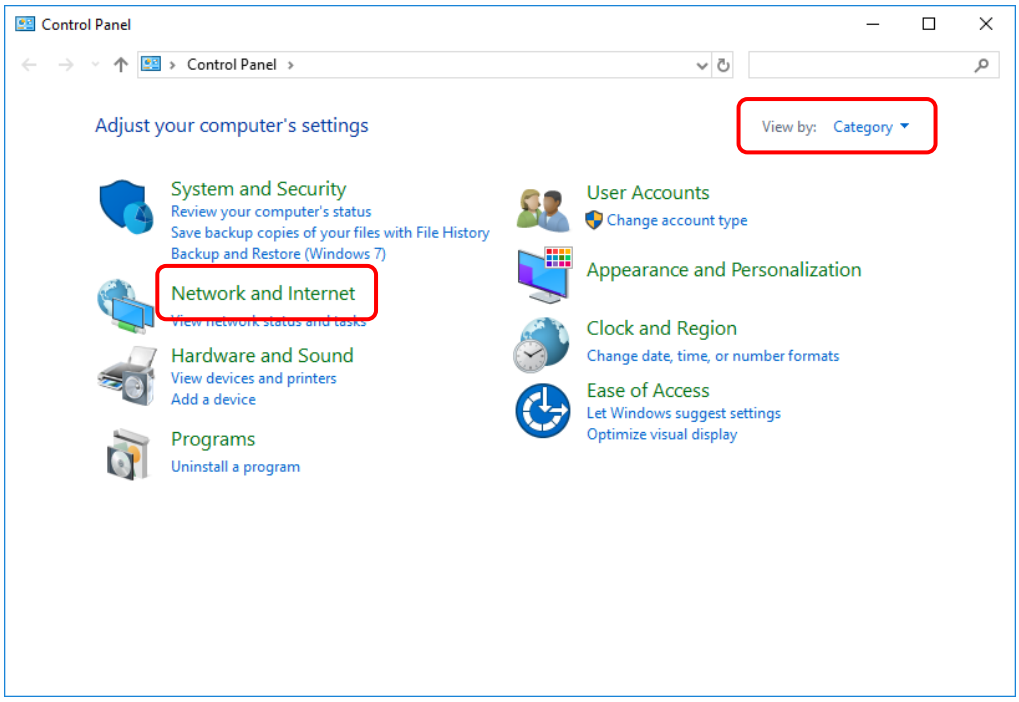
Operate the PC to change the network settings so that it can connect to EM.

Here we are connecting using a LAN cable.

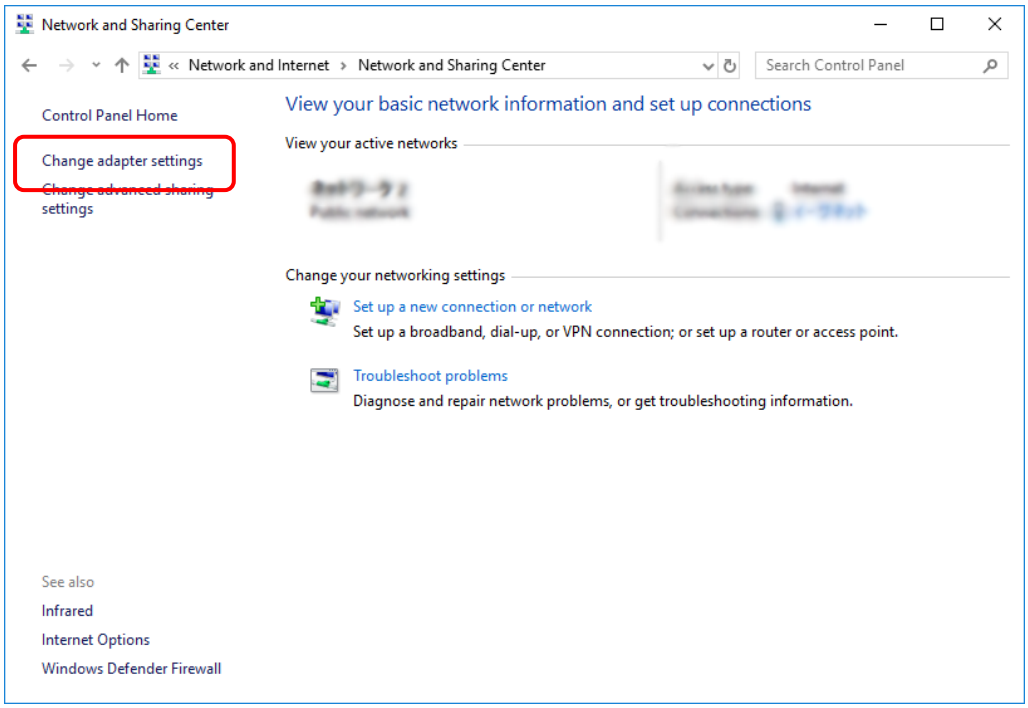
For details on how to connect using a USB cable, please refer to the attached "EM Series Software Development Manual".

1. Open the Control Panel and click [Network and Internet].

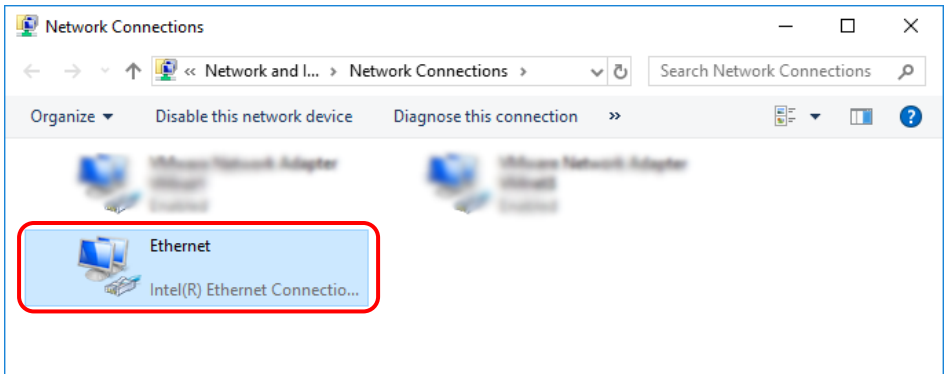
   * If your screen looks different, in the top-right corner set [View by] to [Category].



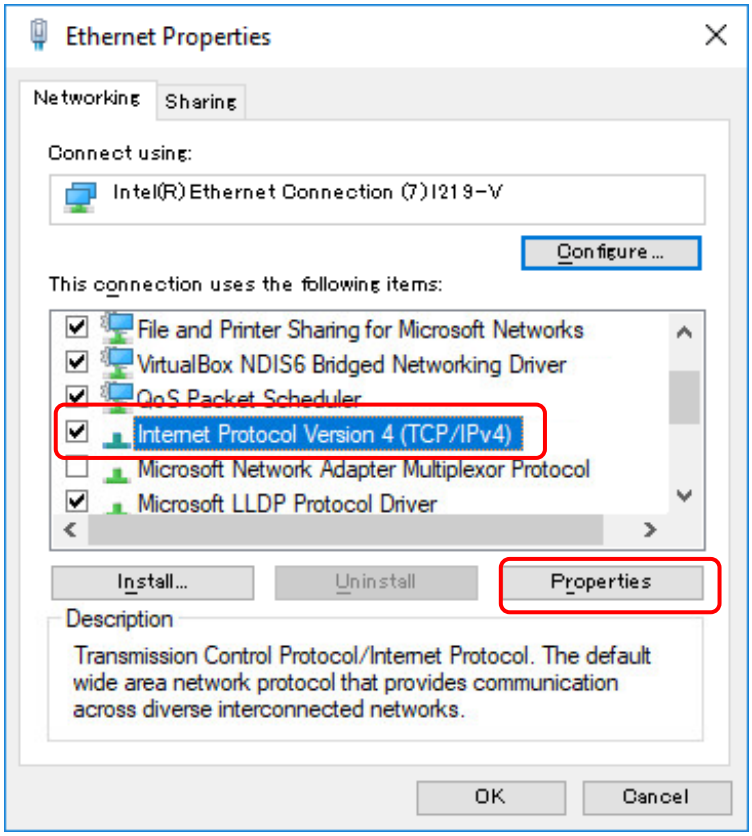
2. Click [Change adapter settings].



6. Right-click the added adapter and click "Properties".



7. Select [Internet Protocol Version 4 (TCP/IPv4)], then click [Properties].



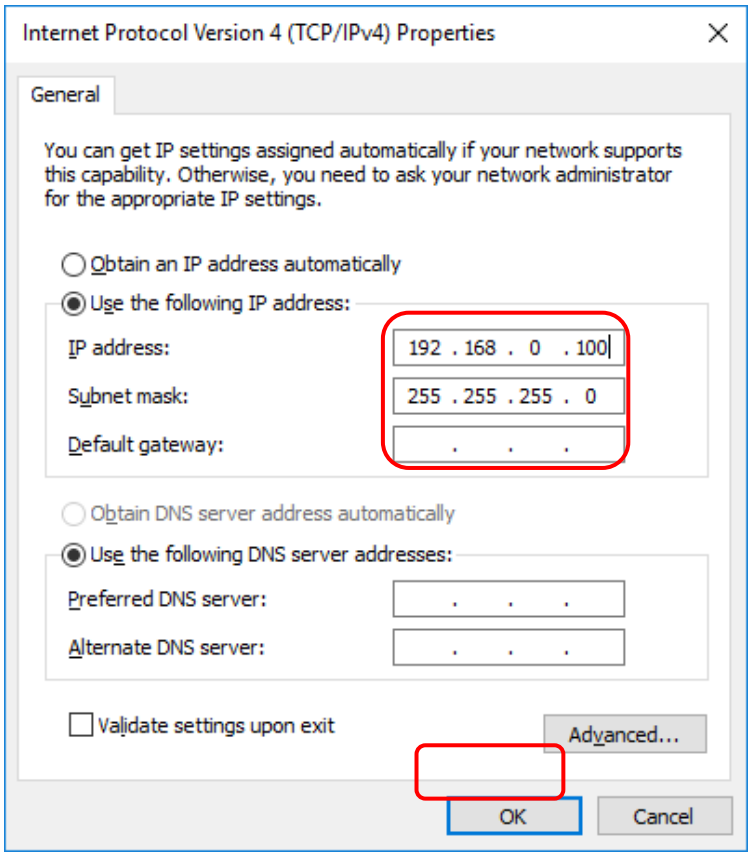
8. Set the IP address, subnet mask, and default gateway to the values below and click the OK button..

Item	Value
IP address	192.168.0.100
subnet mask	255.255.255.0
default gateway	Not set

* If you have changed the IP address of the EM, please set the PC so that it is on the same network.

* Please use a value that does not overlap the IP address of the EM unit.

* "192.168.10.*" cannot be used because it is used for USB-Ether (usb0).



1.6 Update of IS-APP/IS-API/IS-APP SETTING

The software installed on the EM series at the factory is not the latest.

Transfer the InfoSOSA application "IS-APP", the communication library for IS-APP "IS-API", and the IS-APP setting assistance tool "IS-APP SETTING" included in the InfoSOSA Development Kit on the EM series.

For the detail of the console connection and the data transfer, please refer to "EM Series Software Development Manual".

1. Connect the console to the EM series.

2. Execute the following command to disable write protection on the console.

```
# wprotect_off
```

* When restarted, it will return to read-only.

3. Please transfer (overwrite) "is_app", "libisapi.so" and "isapp_setting" to the EM unit.

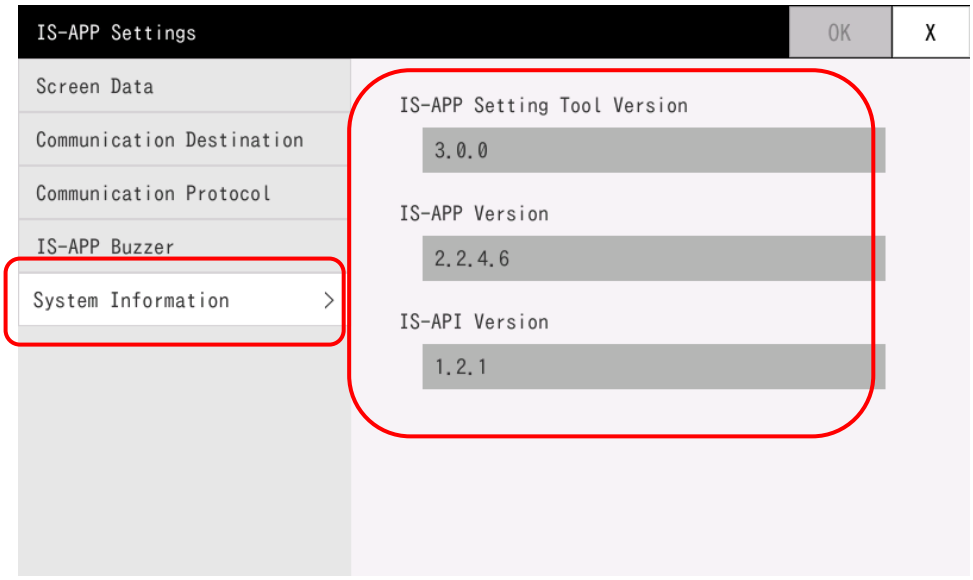
File name	Forwarding destination
is_app	/usr/bin/is_app
libisapi.so	/usr/lib/libisapi.so
isapp_setting	/usr/bin/isapp_setting

"is_app", "libisapi.so" and "isapp_setting" are in the InfoSOSA development kit data.
Please note that the supported files differ depending on the product you are using.

Model	Corresponding file
EMG7-***A8-****-**7	*[development kit data]*\softwareIS-APP\IS-APP-A8\is_app *[development kit data]*\software\IS-API\Library\IS-APP-A8\libisapi.so *[development kit data]*\\software\IS-APP_SETTING\IS-APP-A8\isapp_setting
EM8-***A7-****-**7	*[development kit data]*\softwareIS-APP\IS-APP-A7\is_app *[development kit data]*\software\IS-API\Library\IS-APP-A7\libisapi.so *[development kit data]*\\software\IS-APP_SETTING\IS-APP-A7\isapp_setting
EMG8-***A7-****-**7	

4. Please reboot the EM unit.

This completes the update.
Check that “IS-APP”, “IS-API” and “isapp_setting” have been updated to the target version from the system information of the “IS-APP settings”.



For the target version, refer to the module version in the attached "InfoSOSA Version2.5-2.7 Release Notes".

1.7 IS-APP HMI Development Process

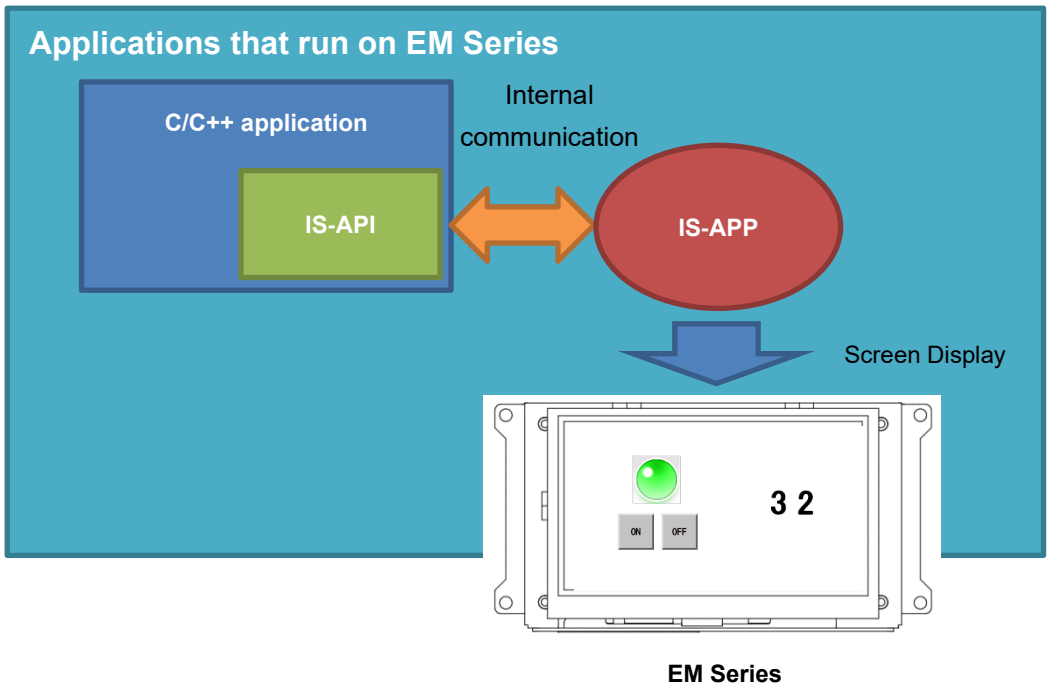
Summary

Steps on using the IS-APP to create the following system are provided.

- Screen display performed by IS-APP
- InfoSOSA Builder (no programming required) to create screens displayed on IS-APP.
- For processes that cannot be designed with IS-APP functions, create your own applications in C, C++, or other programming languages.
- For user-created applications that interact with IS-APP, use the IS-APP access API (IS-API).
- Communication with host device (external) is possible with a user-created application or IS-APP.
  * Not included in this tutorial.

This tutorial describes how to create the following operations on a screen.

- ✓ Touch the ON/OFF button on the LCD screen (IS-APP) to turn a lamp on the screen on or off.
- ✓ Touch the ON/OFF button on the LCD screen (IS-APP) to notify the console application (C++ program) of a change in state.
- ✓ Enter values from the console to change the value displayed on the LCD screen (IS-APP).



EM Series

Development Flow

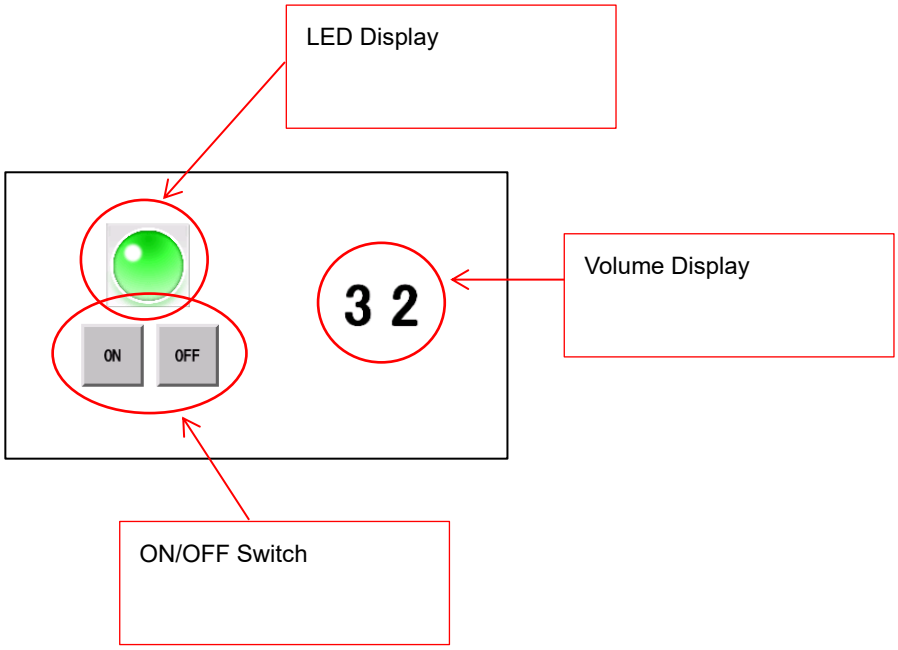
No.	Item	Contents
①	Create System Specifications	Create specifications of the entire system such as the IS-APP display, C/C++ application processes, and communication with host devices.
②	Create InfoSOSA Project	In InfoSOSA Builder, design and save the screens displayed on IS-APP.
③	Create the InfoSOSA Screens	Draw the buttons and set up operations in InfoSOSA Builder.
④	Save the InfoSOSA Project	Save screen data created in InfoSOSA Builder.
⑤	Checking Operations with InfoSOSA Simulator	Check screen data operations on the PC.
⑥	Create C/C++ Application	Using the IS-API, create the C++ application that interacts with IS-APP.
⑦	Transfer to EM Series	Transfer to the EM Series the IS-APP executable file, screen data, IS-API library, and C++ application.
⑧	IS-APP Standalone Test	Start IS-APP and verify standalone operations.
⑨	IS-APP, C++ Application Merge Test	Start the C++ application and run a test on its interactions.

1.7.1 Create System Specifications

Create specifications for the entire system.
This tutorial uses the following specifications.

Screen Display (processes run on IS-APP)

Use InfoSOSA Builder to create the screen display.
Press the ON/OFF button to turn the lamp on the screen on/off, and notify the C++ application that the button has been pressed.



Processes run on C++ application

Using the IS-API, change the volume display on the screen to the value entered on the console.
Display on the console the notification received from IS-APP (that the button was pressed).

Memory

The following two memories are defined

Memory ID	Type	Value range	Contents
GM_LED01	Boolean	0 or 1	LED status
GM_VOL01	Byte	0 to 100	Volume status

In Builder, users can create InfoSOSA memory. With this memory, you can do things like notify the host device or other application to run a process due to a change in the memory value, or use the IS-API to change the memory value which in turn changes the InfoSOSA display.

1.7.2 Create InfoSOSA Project

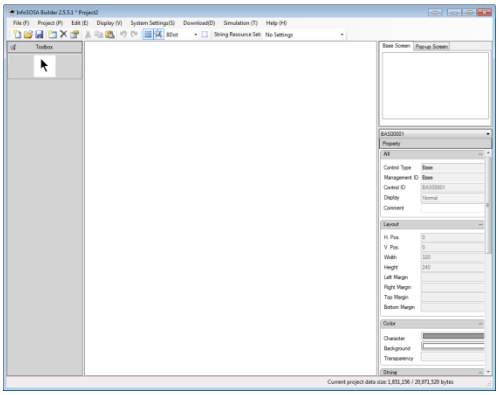
Using InfoSOSA Builder, create a project.
Before proceeding, install InfoSOSA Builder on the host PC (Windows).
Please refer to 1.4 Install InfoSOSA Builder for installation.

NOTE

◆ About Projects

Project is the file that stores screens created with InfoSOSA Builder, as well as communication settings with the host.

Start the InfoSOSA Builder



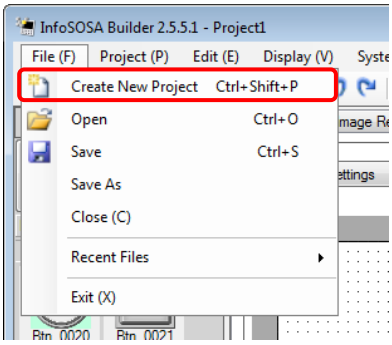
Start up the InfoSOSA Builder by clicking the 

icon on the desktop, or from the Windows Start menu click [Seedsware] - [InfoSOSA Builder*.*]

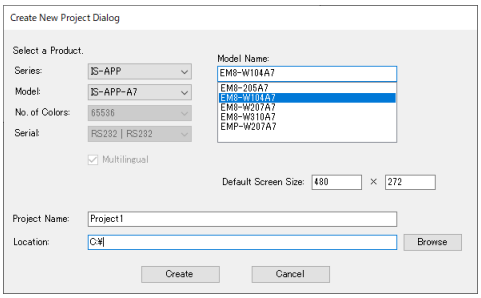
* "*.*" is the version number..

When you start the Builder, a screen as shown to the left will appear.

Create a New Project



From Builder's [File] menu, click [Create New Project].



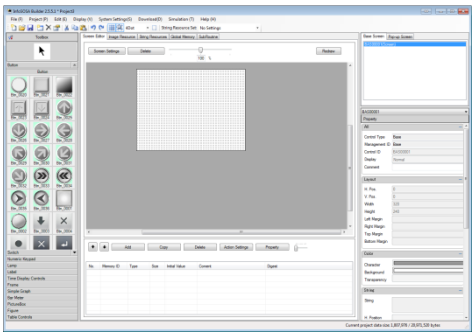
A dialog box as shown to the left will appear.
Select the model that corresponds to your product.
After selecting, enter the project name, click [OK] and create a new project.
* Project is created in the location defined in the [Location] field.

NOTE

You can set the project name and location.

In IS-APP, select the model type that corresponds to your model.
Set the series name to [IS-APP] and select the model type associated with your unit.

Product Model	Model	Model Name
EMG8-W104A7-0005-207	IS-APP-A7	EM8-W104A7
EM8-W104A7-0005-207		
EMG8-205A7-0005-207		EM8-205A7
EM8-205A7-0005-207		
EMG8-W207A7-0005-207		EM8-W207A7
EM8-W207A7-0005-207		
EMG7-W207A8-0024-107-01	IS-APP-A8	EMG7-W207A8
EMG7-312A8-00DC-107-01		EMG7-312A8



A screen as shown to the left will appear when a new project is created.

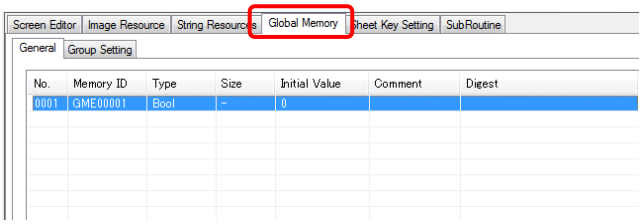
1.7.3 Create the InfoSOSA Screens

Set up parts and memory.

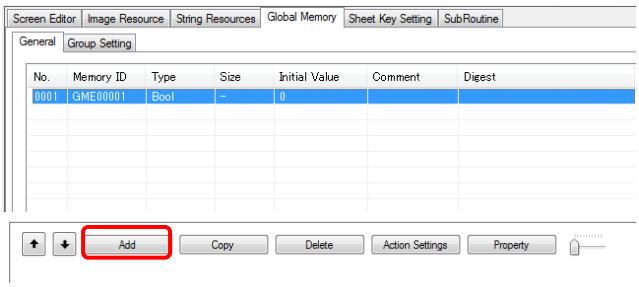
Set up Global Memory

Create InfoSOSA memory.
Memory is something that saves a value in the InfoSOSA.

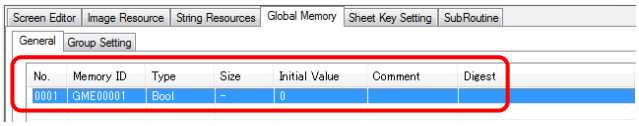
Create Memory



Select the [Global Memory] tab.



Click [Add].



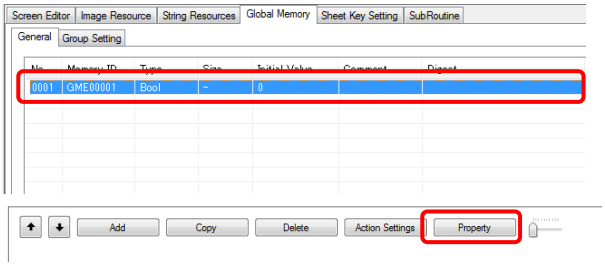
The memory is created.

NOTE

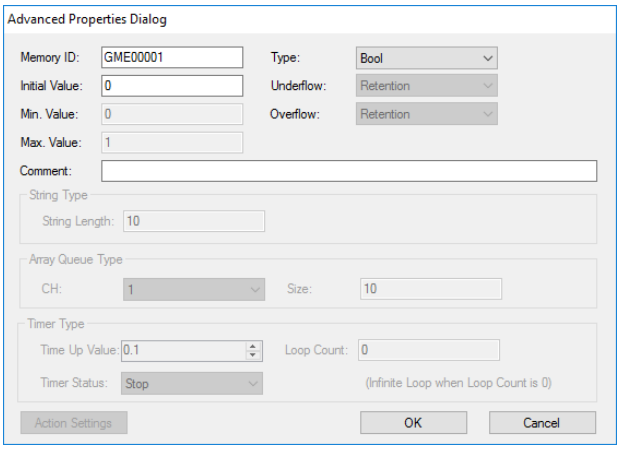
◆ Memory Type

Global Memory, whose value is always accessible and saved while the application is running. Screen Memory, whose value is only accessible when specific screens are displayed. When displaying a different screen, its value is initialized.

Property Settings



Select memory, then click [Property].



Change the properties as per the table.

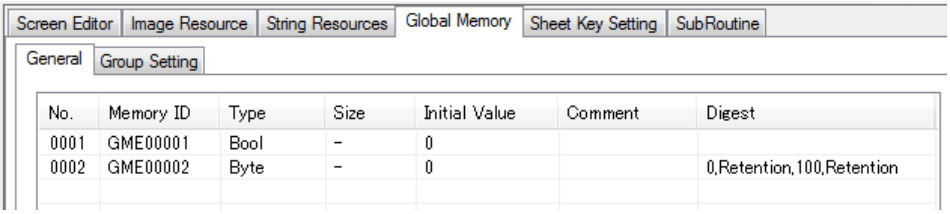
Property	Value	Property Description
Memory ID	GM_LED01	ID that identifies memory.
Type	Boolean	Memory Type
Initial Value	0	Initial value at power on

Create second memory

Follow the same procedure to create another memory. Set the properties of the second memory as follows.

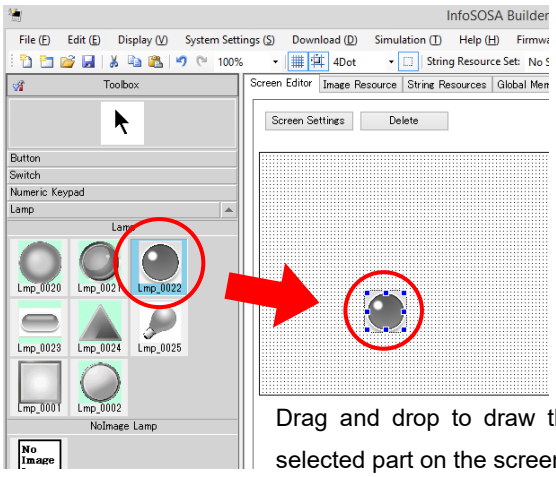
Property	Value	Property Description
Memory ID	GM_VOL01	ID that identifies memory.
Type	Byte	Memory Type
Initial Value	0	Initial value at power on
Min. Value	0	Minimum value in the memory.
Max. Value	100	Maximum value in the memory.
Underflow	Retention	Operation when a value less than the minimum value is set in the memory.
Overflow	Retention	Operation when a value greater than the maximum value is set in the memory.

After setup, the list is as follows.



Set up Lamp

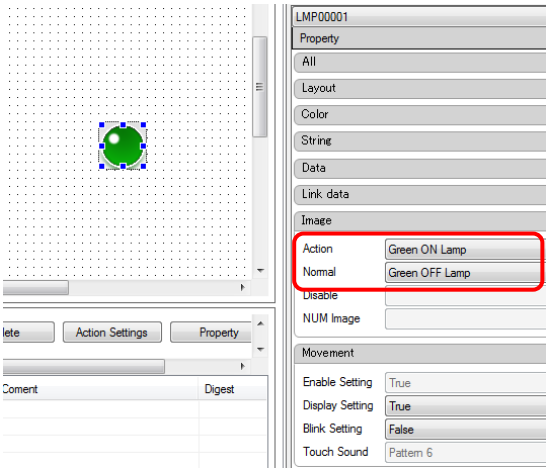
Arrangement of the Lamp



From the Toolbox on the left of the Builder, select **Lamp** then drag and drop **LMP_0022** on to the screen.

Drag and drop to draw the selected part on the screen.

Change the Lamp image



While the placed lamp is selected and active, set the [Image] in the property area on the right side of Builder. Set the [Normal] and [Action] images to change the lamp image.

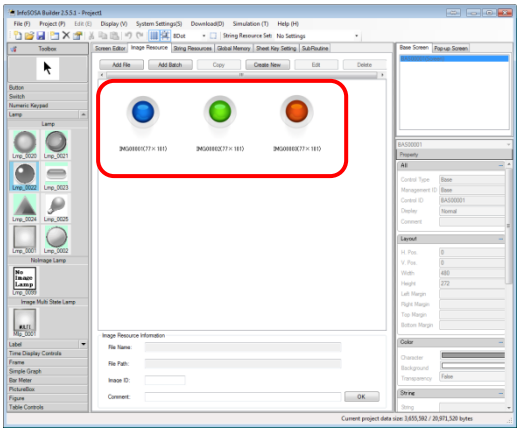
[Action] sets the lamp image when it is ON, and [Normal] sets the lamp image when it is OFF.

In this tutorial, set up as follows.

• Action: Green ON

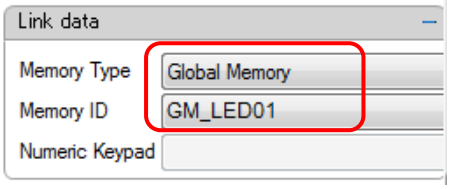
• Normal: Green OFF

The image of bit map parts can be changed as above.



By importing the image file to the Image Resources, you can create a lamp image that is not included in the default images.

Lamp Link Setting



Create an association between the lamp and Global Memory. With the link setting, the Normal image is displayed when the associated Global Memory value is 0, and the Action image is displayed when the value is 1.

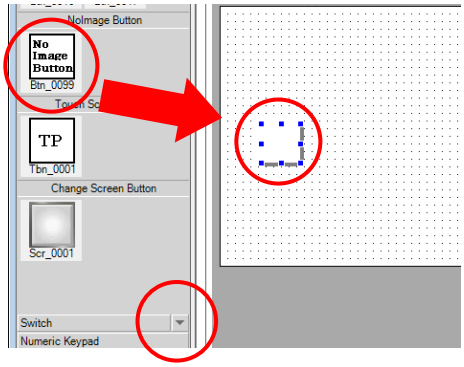
In this tutorial, set up as follows.

・Memory Type: Global Memory

・Memory ID: GM_LED01

Set up Button

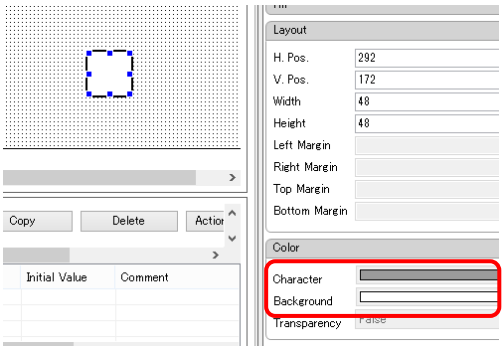
Arrangement of the Button



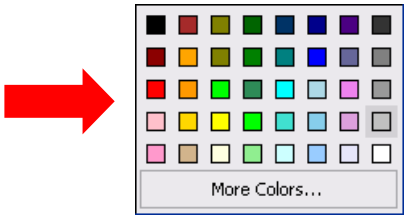
Select [Button] from the Toolbox at the left side of the Builder and drag and drop [Btn_0099] on to the screen. If you cannot see it, scroll down with ▼.

Drag and drop to draw the selected part on the screen.

Change Color of Button

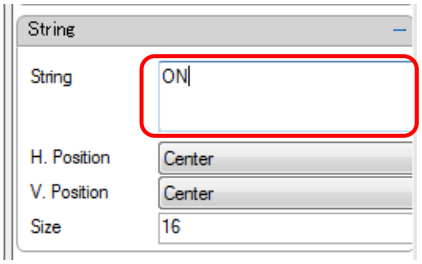


While the placed button is selected and active, set the color in the Property area at the right side of the Builder. Click the [Background] under [Color] to display the color pallet as below and change the color of the button.



You can change the color of NoImage parts as shown above.

Change Text on the Button



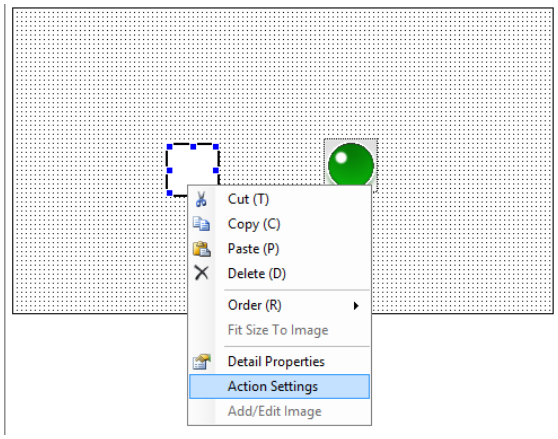
By changing the String property, you can display text on the button.

In this tutorial, set up as follows.

・String: ON

Action Settings①

Set the action of the button so that the lamp on the screen turns on when the button is pressed.

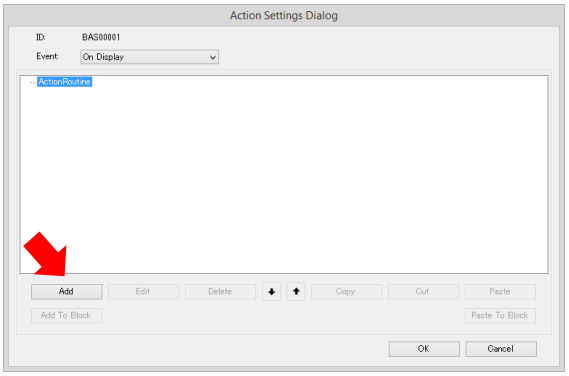


To set the action, right-click on the button placed on the screen and click [Action Settings].

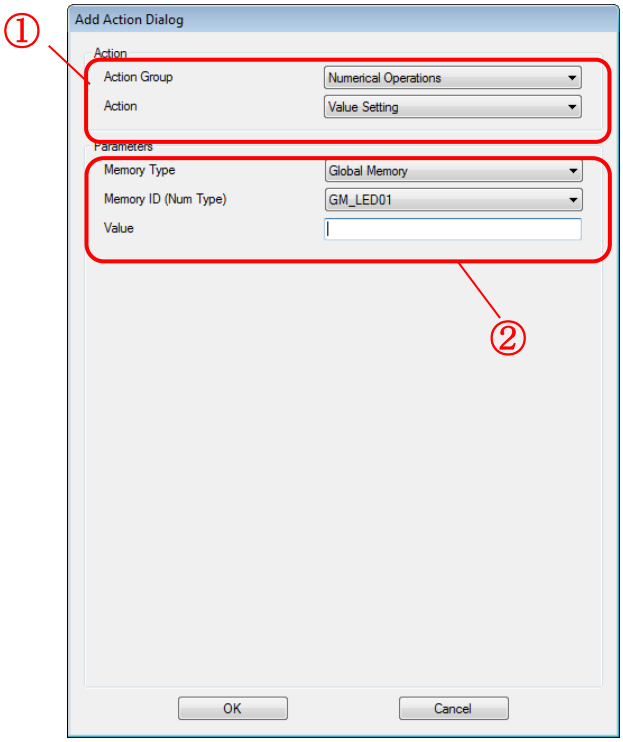
NOTE

◆ About Actions

[Action] refers to the operation of parts on the screen.



From the [Action Settings Dialog], select [Add] to open a dialog box.



You can set the following in the [Add Action Dialog].

① Action

- Action Group

  => Select the category of action you want to use.

- Action

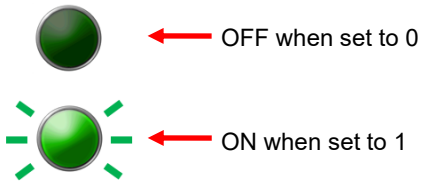
  => Select the action you want to use.

② Parameters

=> Set on which screen, to which part you want to set the action, and to what value.

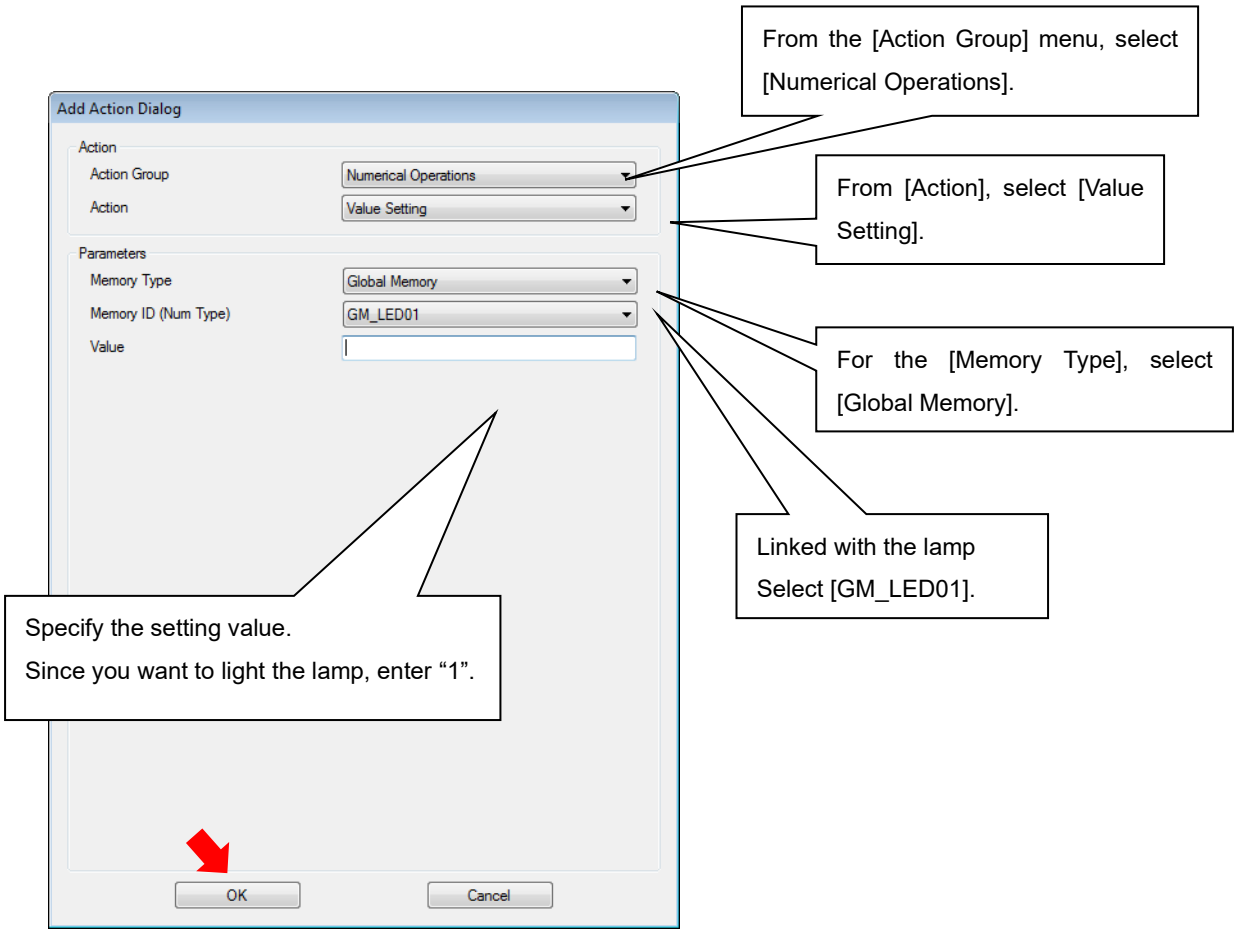
You can set items such as [Screen ID], [Parts ID] and [Setting Value].

However, items that can be set will vary according to the action that you choose.



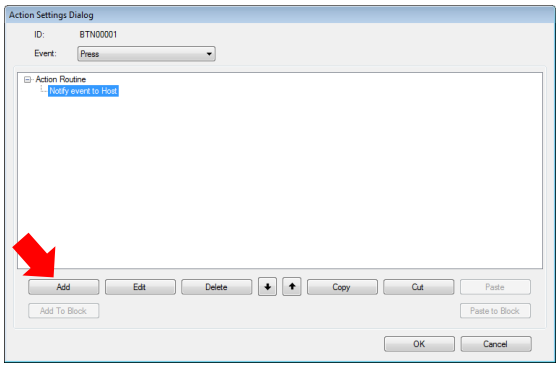
This time we will set the action "Turn ON lamp". When 0 is set to the linked memory of the lamp, it turns OFF. When 1 is set to the linked memory, it turns ON.

The action setting to make the lamp light up is as shown below:



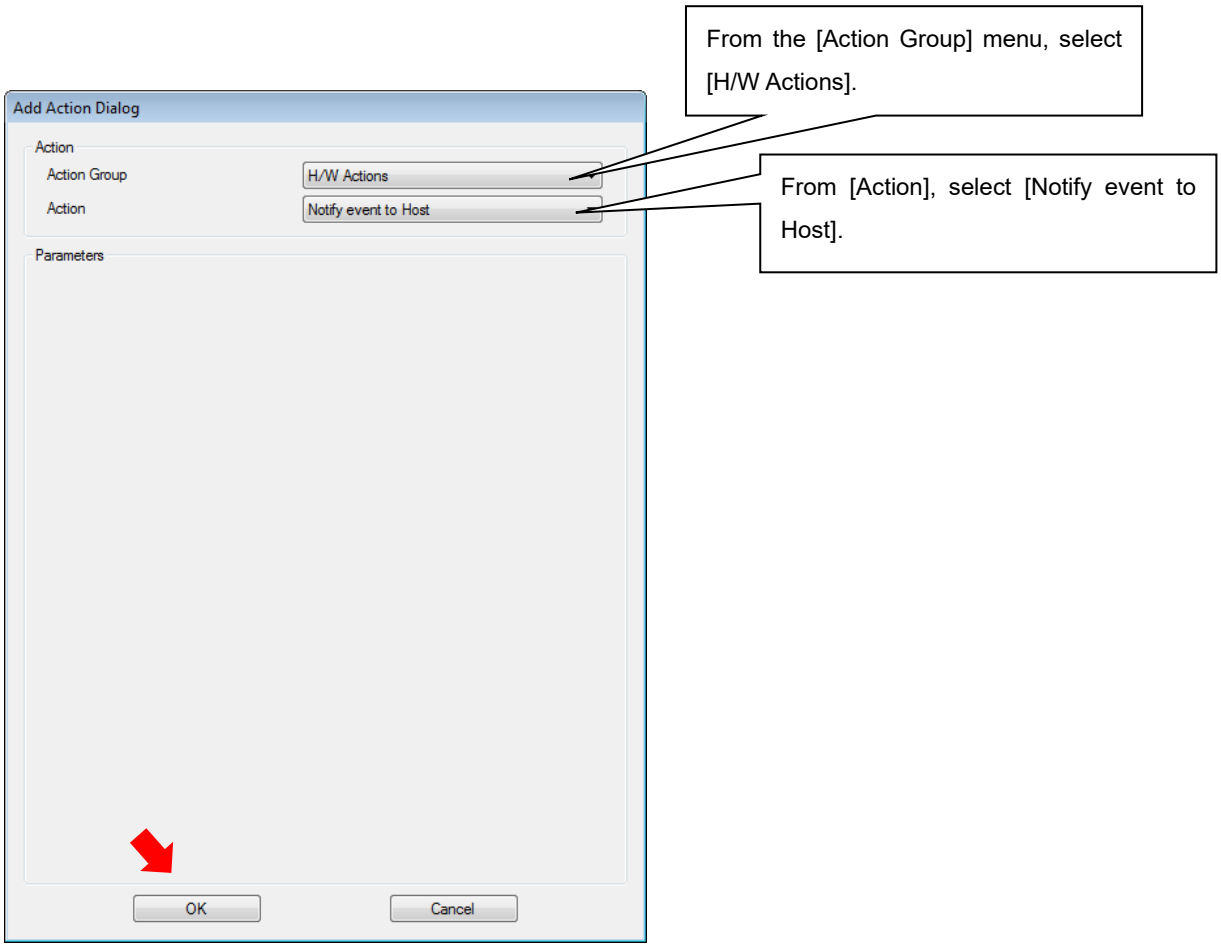
Action Settings ②

Set up a notification operation to the communication partner for a button that was pressed (notify event occurrence).



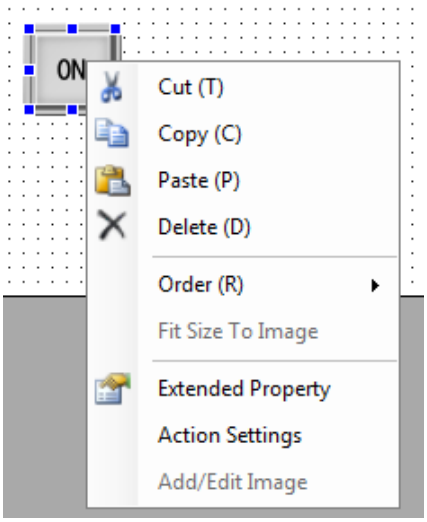
At the bottom left part of the dialog box, select [Add] and open [Add Action Dialog].

The action setting to notify the communication partner of an event is as shown below:



Create second button

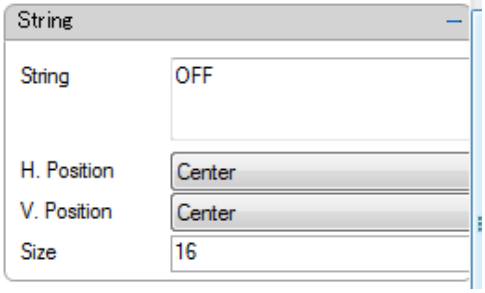
Create a button to turn off the lamp.



Right-click the ON button, select [Copy] from the shortcut menu, then right-click again and select [Paste].

The ON button is copied and you only have to set the changes.

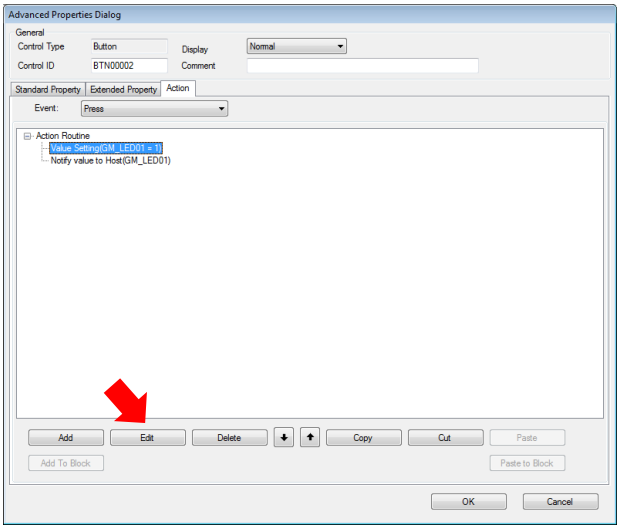
Change the display text to "OFF"



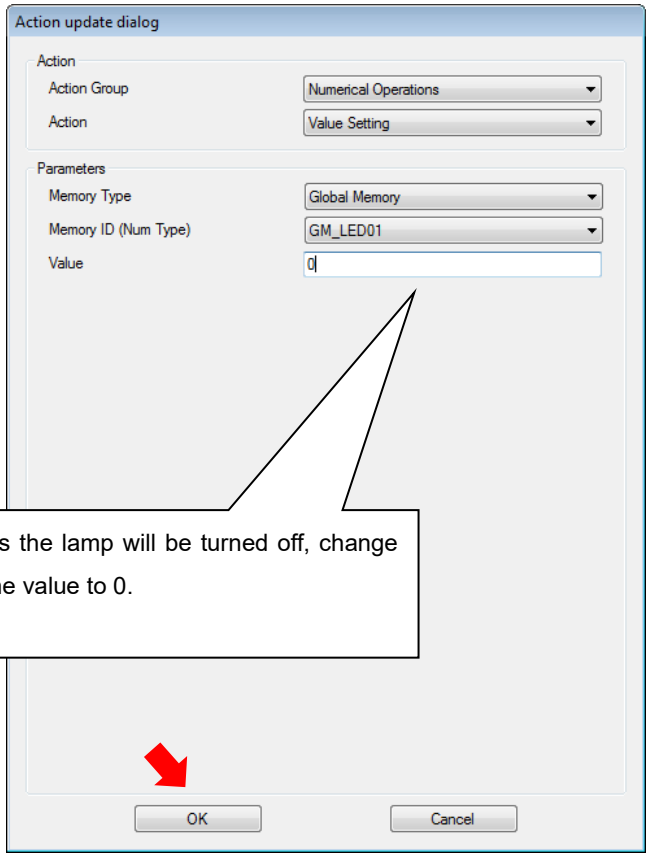
String Property OFF

・String: OFF

Change the action to turn the lamp off

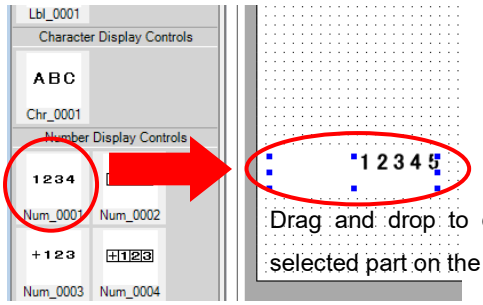


While the action to turn on the lamp is selected, at the bottom left of the dialog box select [Edit] and open [Action update dialog].



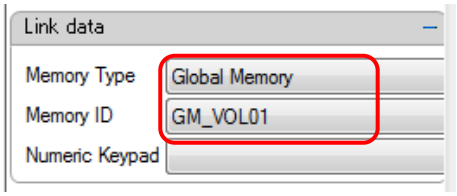
Set up Number Display Controls

Draw a Numeric Display



Select [Label] from the Toolbox at the left of the Builder and drag and drop [Num_0001] on to the screen.

Numeric Display Link Setting



Create an association between the Number Display Controls and Global Memory. When link setting is run, the associated Global Memory value is displayed.

In this tutorial, set up as follows.

・Memory Type: Global Memory

・Memory ID: GM_VOL01

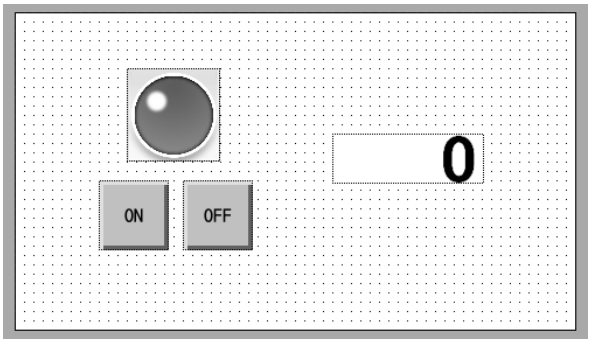
Edit the display digits for the Numeric Display

Set up the number of digits to display.

In this tutorial, set up as follows.

・Display Digit: 3

Adjust the size and position of each part

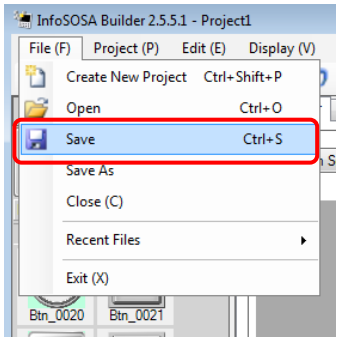


Use the mouse to adjust size and position of parts.
You can change the size of a part by dragging the edge of the part.

1.7.4 Save the InfoSOSA Project

Saving Project

Save the project



Saving Project

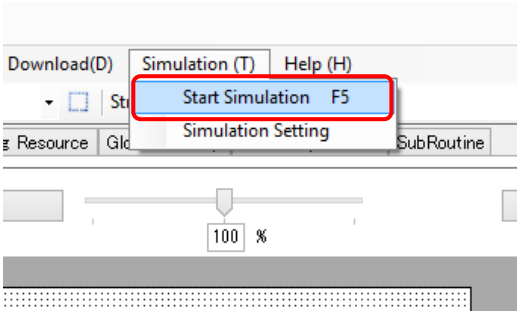
From the [File] menu, select [Save] to save the project. You can also save with the toolbar's save icon.

∗ When saving the project with a different name, click [Save As].

1.7.5 Check Operations with InfoSOSA Simulator

With the simulator, check the operation of the project you created.

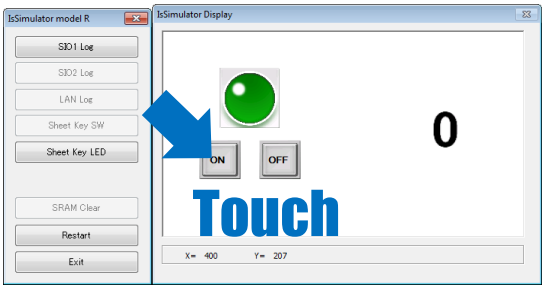
Start Up the Simulator



You can use the PC to check operations of the project you created.

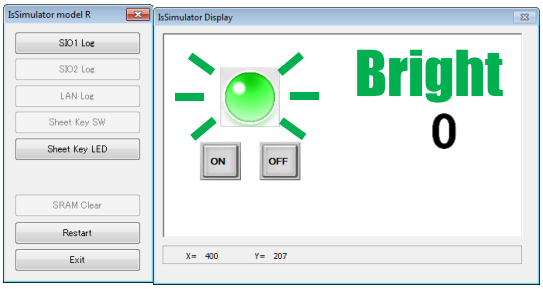
To start up the simulator, from the [Simulation] menu, select [Start Simulation.]

Operation Check with Simulator



The created screen is displayed on the simulator.

Check if the action set actually works by clicking the button displayed on the simulator.



If the action is set correctly, the lamp will light up.

If the lamp lights, simulation is complete.

1.7.6 Create C/C++ Application

Using the IS-API library, create an application that interacts with IS-APP.
From here, in the EM Series software development environment (Linux virtual machine), create source files.
The software development environment needs to be prepared beforehand.
For information about creating the software development environment, refer to the "EM Series Software Development Manual".

Copy Related Files

To use IS-APP, header files and library files are required.
Please copy the following files to the storage directory.

Storage directory *	/home/em/test/

*If the storage directory does not exist, please create it. You can change the storage directory.

Library files
libisapi.so

Header files
CIsApi.h
CIsComString.h
CIsComVariant.h
CIsComVariantList.h

The library/header files are located in the \\software\IS-API\ folder in the InfoSOSA Development Kit Data.

The library file varies depending on your model.
Use the file in the folder that corresponds to your model.

Model	Corresponding folder
EMG7-***A8-****-**7	IS-APP-A8
EM8-***A7-****-**7	IS-APP-A7
EMG8-***A7-****-**7	IS-APP-A7

Create Source Files

Start up a text editor to create the source files. Input the information as follows. Character code of source files should be UTF-8.

Save created source files as shown below.

Save directory *	/home/em/test/
File Name	tutorial.cpp
Character code	UTF8

* If the save directory does not exist, please create it.
  You can change the save directory and file name.

```
// [1] Header file include statements
#include "ClsApi.h"
#include "ClsComString.h"
#include "ClsComVariant.h"
#include "ClsComVariantList.h"
#include <iostream>
#include <string>

void event_callback01(ClsComVariantList &list);
void event_callback02(ClsComVariantList &list);

ClsApi* pIsApi = NULL;

int main()
{
    // [2] ClsApi create class object
    pIsApi = new ClsApi(51111);
    std::cout << "Connecting..." << std::flush;

    // [3] IS-API startup
    if(!pIsApi->Start()){
        return false;
    }

    // [4] check operation status
    if(!pIsApi->IsRunning()){
        return false;
    }
```

```
    // [5] test communication
    if(!pIsApi->Test()){
        return false;
    }
    std::cout << " OK" << std::endl;

    // [6] register callback function
    std::string str01("BAS00001.BTN00001.PRESS");    // ON button event ID
    std::string str02("BAS00001.BTN00002.PRESS");    // OFF button event ID
    pIsApi->registerCallback(&str01, event_callback01);    // ON button
    pIsApi->registerCallback(&str02, event_callback02);    // OFF button

    // [7] change IS-APP memory value
    CIsComVariant var;
    std::string    propertyID="@GLBMEM.GM_VOL01.VALUE";
    while(true)
    {
        int vol;
        std::cout << "Input numeric value and press [ENTER]."    << std::endl;
        for ( std::cin >> vol ; !std::cin ; std::cin >> vol){
            std::cin.clear();
            std::cin.ignore();
            std::cout << "Input is not a numeric value."    << std::endl;
        }
        var.SetValue(vol);
        pIsApi->setProperty(&propertyID, var);
        std::cout << "setProperty(" << var.GetValue() << ")" << std::endl;
    }

    // [8] end of process
    if(pIsApi){
        pIsApi->Stop();
        delete pIsApi;
    }
    return 0;
}

// [9] callback function 1
void event_callback01(CIsComVariantList &list)
{
    std::cout << "LED ON!" << std::endl;
}

// [9] callback function 2
void event_callback02(CIsComVariantList &list)
{
    std::cout << "LED OFF!" << std::endl;
}
```

Source Files Commentary

The content below will explain each part of the source file. It corresponds to the number of the comment part.

[1] ClsApi class header file include statements

Include the IS-API header file.

Include File	ClsApi.h
	ClsComString.h
	ClsComVariant.h
	ClsComVariantList.h

[2] Create ClsApi class object

To use the IS-API, first create the ClsApi class object.
You can go through this object to use the IS-API.

■ Format

ClsApi(int hPort)

■ Parameters

Parameter	Description
int hPort	IS-API port number

[3] Start IS-API

IS-API operates as a TCP/IP server.
Start standby, and wait for the IS-APP connection.

■ Format

bool Start()

■ Return Value

true: Successful
false: Failed

[4] Check operation status

Gets the IS-API operation status. You can use IS-API functions only while this application is running (when the return value of this function is true). (The operation check is not mandatory.)

■ Format
bool IsRunning()

■ Return Value
true: Running
false: Stopped

[5] Test communication

Actual test for IS-APP communication. (Testing communication is not mandatory)

■ Format
bool Test()

■ Return Value
true: Successful
false: Failed

[6] Register callback function

When you register a callback function, when a button displayed on InfoSOSA is pressed*, the registered function is executed.
* In InfoSOSA Builder, setting up the [Notify event to host] action on a button is required.

■ Format
bool registerCallback(std::string* _eventID, EventCallbackFunc _func)

■ Parameters

Parameter	Description
std::string* _eventID	Target event ID * Set up the event ID with the format: [Affiliation ID].[Part/Memory ID].[Event ID] For more information, please refer to the "Reference Manual".
EventCallbackFunc _func	Function for registration

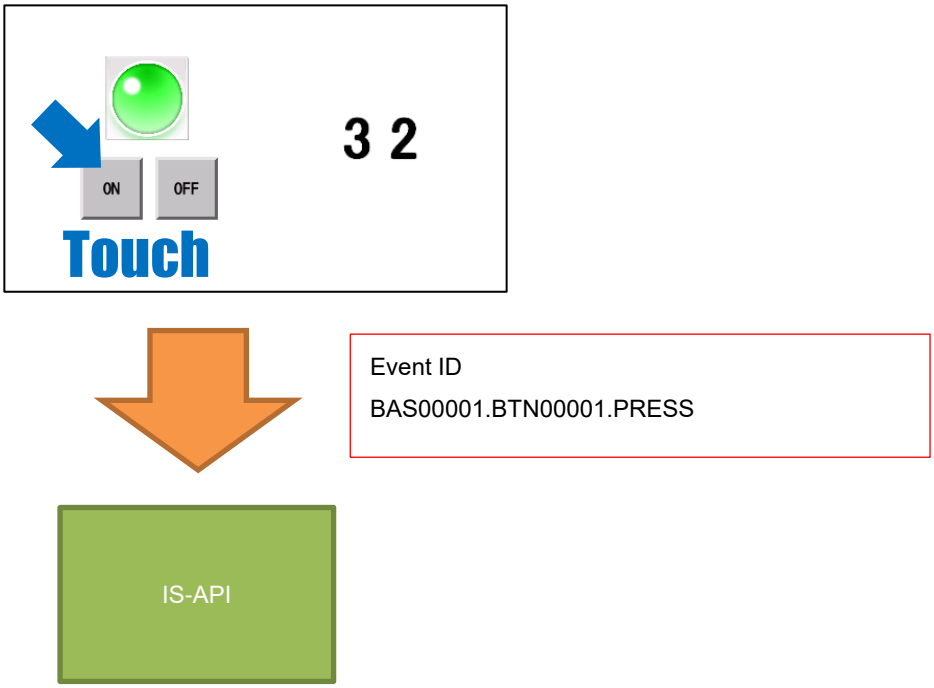
■ Return Value
true: Successful
false: Failed

For callback function registration, specify *which function to execute* and *when which event occurs*.

On InfoSOSA, an event is the trigger for executing actions (Operation Setting).
For example, when a button is pressed, a PRESS event occurs on the button. When the button is released, a RELEASE event occurs.

InfoSOSA Builder can set up actions for each of these events, such as lighting up a lamp when the button is pressed, and turning the lamp off when released.

If you set up a host notification action (such as Notify event to Host) for the event, a unique ID is sent to the host (IS-API).



As below, create an event ID string for when the button is pressed, and pass it to the registration function.

```
std::string str01("BAS00001.BTN00001.PRESS");
pIsApi->registerCallback(&str01, event_callback01);
```

For information on events and event ID specifications, please refer to the "InfoSOSA Reference Manual".

[7] Change IS-APP memory value

Waits for input from the console, and sets the entered value to the InfoSOSA memory.

■ Format

bool setProperty(std::string* _propertyID, CIsComVariant& _var)

■ Parameters

Parameter	Description
std::string* _propertyID	Property ID of the item to change * Set up the ID with the format: [Affiliation ID].[Part/Memory ID].[Property ID] For more information, please refer to the "InfoSOSA Reference Manual".
CIsComVariant& _var	Value and string to set * Use UTF-8 character encoding for strings.

■ Return Value

true: Successful
false: Failed

To change the InfoSOSA memory, specify *which property* is changed to *what value*.

Each InfoSOSA part and memory is set up with a unique ID with multiple properties.
The ID is set up with the format: [Affiliation ID].[Part/Memory ID].[Property ID]
For more information, please refer to the "InfoSOSA Reference Manual".

As below, create a string and pass it to the set function.

```
std::string propertyID="@GLBMEM.GM_VOL01.VALUE";
```

Item	Description
Affiliation ID	@GLBMEM
Parts/Memory ID	GM_VOL01
Property ID	VALUE

Create the value of type CIsComVariant.
For the CIsComVariant, define the value with the SetValue method as follows.

```
var.SetValue(vol);
```

■ **Format**
bool SetValue(int value)

■ **Parameters**

Parameter	Description
int value	Value

■ **Return Value**
true: Successful
false: Failed

[8] End of process

Stop the IS-API's TCP/IP server, and disconnect from IS-APP.

■ **Format**
bool Stop()

■ **Return Value**
true: Successful
false: Failed

[9] Callback function

Function that was registered in [6] Register callback function. When the InfoSOSA button is pressed, the console LED turns ON/OFF.

Build

Build the source files. Continue working with the software development environment (Linux virtual machine).

Start the terminal

From the toolbar, click  and start the [Terminal].

Move to the source file directory

Input the following commands into the terminal.

```
$ cd /home/em/test
```

* If you changed the save directory, move to that directory instead.

Set up build environment

Set up the environment variables and so on. Must be run every time you start up the terminal.
The script varies depending on your model.
You need to install the tool chain in the development environment beforehand. For information about installing the tool chain, refer to the "EM Series Software Development Manual".

Model	Script
EMG7-***A8*-****-**7	IS-APP-A8 script *1
EM8-***A7*-****-**7	IS-APP-A7 script *2
EMG8-***A7*-****-**7	IS-APP-A7 script *2

*1 IS-APP-A8 script

```
$ source /opt/poky/2.1.2/EM-A8/environment-setup-armv7a-neon-poky-linux-gnueabi
```

*2 IS-APP-A7 script

```
$ source /opt/poky/2.1.2/EM-A7/environment-setup-cortexa7hf-neon-poky-linux-gnueabi
```

If you changed the default folder (//opt/poky/2.1.2/) for tool chain, change to match the install destination.

Run build

Run build with the following command.

```
$ $CXX tutorial.cpp -L./ -lisapi -o tutorial
```

Item	Description
$CXX	Environment Variable $CXX Define the environment variables when you set up the build environment. The following are appropriate compiler and compile options.
tutorial.cpp	Build target source files
-L./	Library search path Define the current directory.
-lisapi	Define the library to use (libisapi.so)
-o tutorial	Output File Name

The following executable file is generated in the current directory.

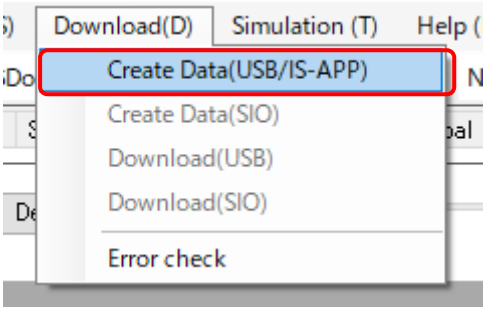
Executable file	tutorial

1.7.7 Transfer to EM Series

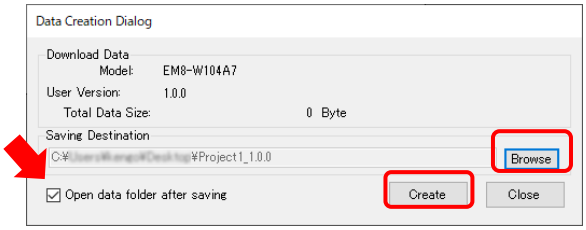
Transfer created data to EM Series.
It is necessary to update the software installed on the EM series in advance.
Please refer to "1.6 Update of IS-APP/IS-API/IS-APP SETTING".

Create IS-APP transfer data

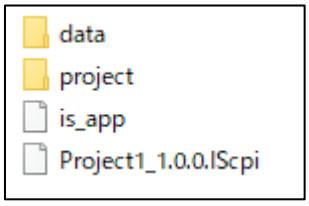
On the host PC (Windows), run InfoSOSA Builder. Following the steps below, use InfoSOSA Builder to convert the created project into transfer data.



From the [Download] menu, select [Create Data(USB/IS-APP)].



Create transfer data. With the [Open data folder after saving] check box selected, press [Save]. To change the save location of download data, click [Browse] and edit the destination folder.



The save destination folder opens.

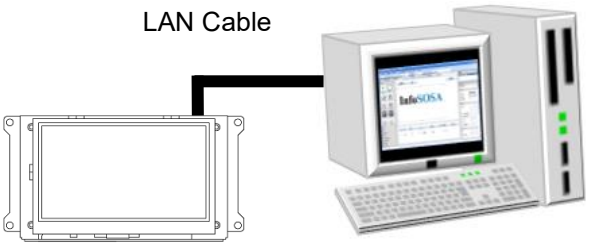
Folder/File	Description
data	Converted screen data for transfer
project	Screen data before conversion (editable)
is_app	IS-APP executable file
Project1_1.0.0.IScpi	System file that describes the contents of the transfer data. File name is [ProjectName]_[UserVersion].IScpi

Transfer process

PC Operation

Operate the PC (Windows).
SambaServer is installed in EM series, and the user folder of EM series can be accessed from Windows.

Network settings are required for transfer.
Follow the procedure in "1.5 Connection between EM series and PC ".



While the EM Series unit is on, connect to the PC with a LAN cable.

Access the following address with Explorer etc.

EM series user folder path	\\192.168.0.130\em\user

*The IP address is the default value when shipped from the factory.

Log in as the following user.

user	root
password	(none)

Use Explorer or other tool to copy the following files to the EM Series unit which was recognized as a storage device.

■ IS-APP
Copy the data created with [Create IS-APP transfer data].

Folder/File	Description
data	Converted screen data for transfer * Copy the entire folder

CAUTION
To update the screen data, delete the entire folder with the converted screen data for transfer, and then run transfer. (If previous files are in the folder, it could cause an operation failure.)

■ IS-API

Copy from the software development environment (Linux virtual machine).

Folder/File	Description
tutorial	Executable file created in step "1.7.6 Create C/C++ Application"

[If you cannot access the EM series user folder]

- Check if the IP address for the LAN cable is "192.168.10.*". It can be changed from the system setting tool.
- Please check whether there is another device with the same IP address "192.168.0.130" in your PC environment.

1.7.8 IS-APP Standalone Test

Please start IS-APP. You can also set it to start automatically when the power is turned on.

Startup method

Using the host PC (Windows), connect the console to EM Series. For information about connecting the console, refer to the "EM Series Software Development Manual".

Navigate to the folder (user folder) where data was transferred.

```
$ cd /mnt/user/
```

Run the executable file **is_app**.
IS-APP uses run time command-line arguments to run display screen data folder and communication settings.

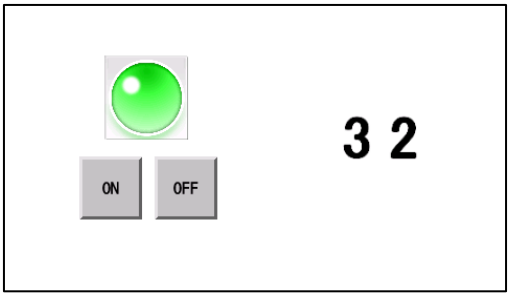
```
$ export DISPLAY=:0
$ is_app -r /mnt/user/data/ -a 51111
```

* When running the program, the first line sets the display destination.
(Set display number 0 to the **DISPLAY** environment variable)
* Screen data specifies the folder.

Argument	-r, --rscdir
Format	-r <screen data folder path>
Description	Specify the screen data folder.

Argument	-a, --api
Format	-a <connection port number>
Description	On the connected unit, define the IS-API.

A screen as shown below is displayed.



For information on other command line arguments, please refer to "2.1.6 Command line arguments".

Startup method (auto)

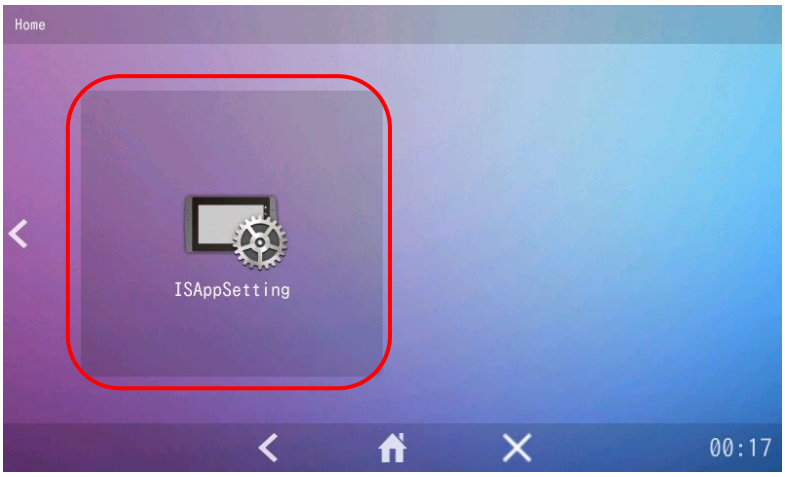
This section describes how to automatically start IS-APP when the power is turned on.

1. Startup script creation.

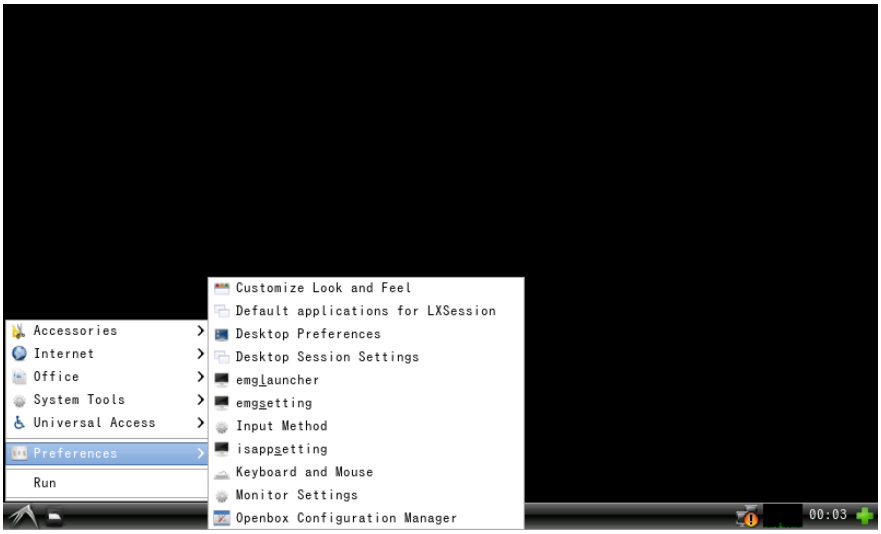
IS-APP specifies communication settings, etc. as command line arguments at startup.
ISAPP SETTING allows you to set command line arguments using the GUI.
(Create a startup script with command line arguments set)

From the EMG Launcher or the Start menu click [ISAPP SETTING].

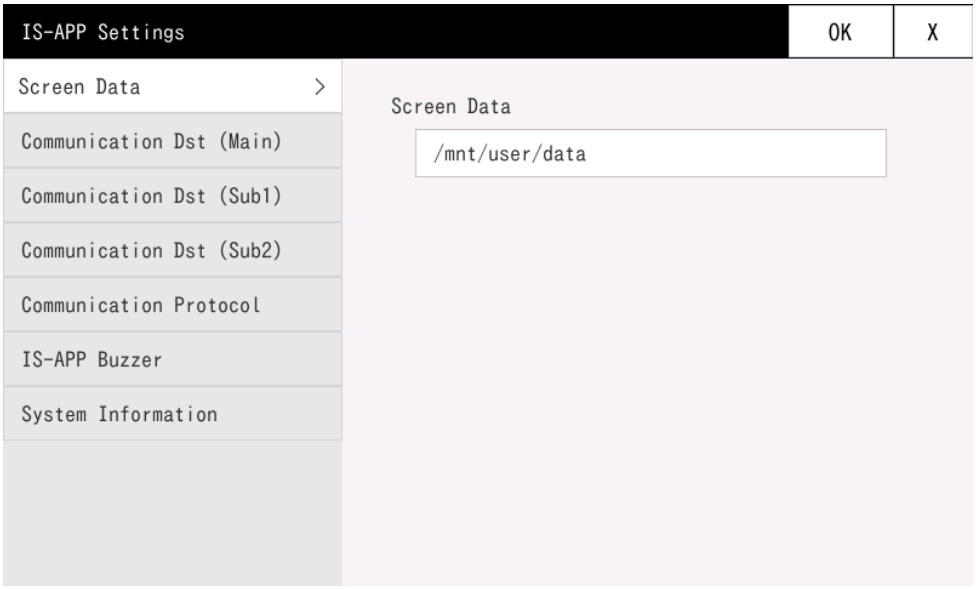
[EMG Launcher] - [ISAppSetting]



[Start menu] - [Preferences] - [isappsetting]



After completing the settings, touch the OK button at the top right to update the startup script file.
Here, leave the default settings and close by clicking the × button.



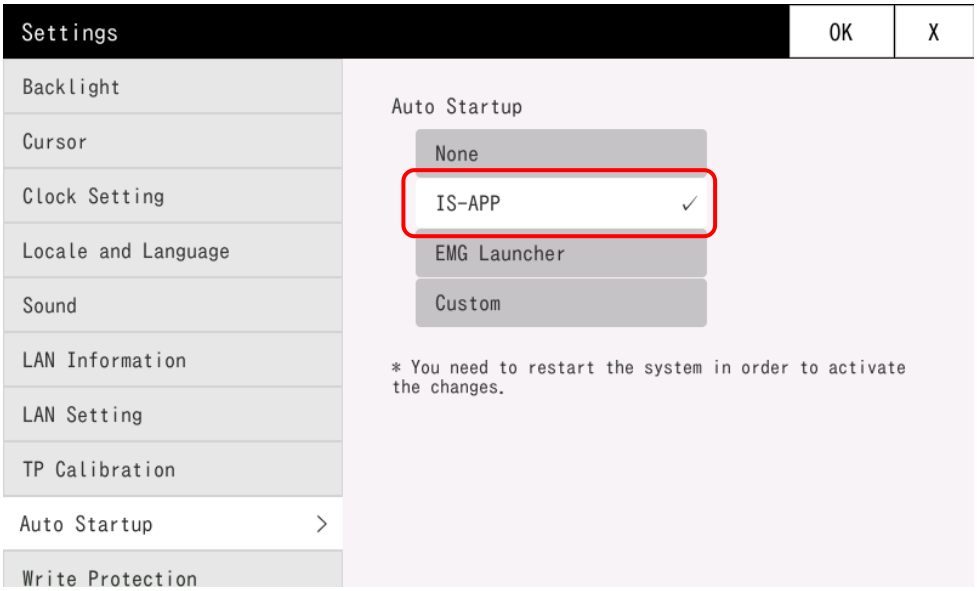
The start script is created in the following directory.

/mnt/user/isapp_run.sh

For each setting, please refer to "2.3 ISAPP SETTING".

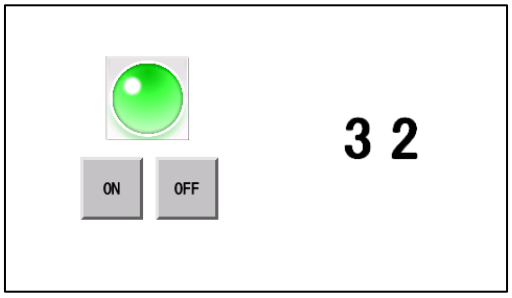
2. Startup script creation.

Settings to automatically run the startup script and start IS-APP when the power is turned on can be configured using the system settings tool.



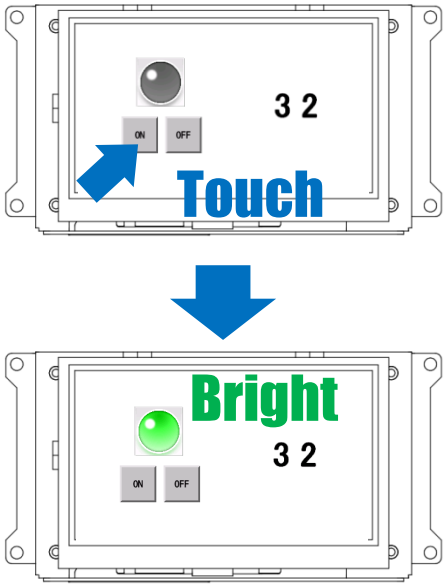
For more information, refer to the separate document “EM Series Tool Manual”.

When you turn the power back on, the following screen will appear.



Operation Check

Touch the ON/OFF button to turn a lamp on the screen on or off.
Without communication, InfoSOSA uses an action to change the value in its memory GM_LED01 (inside InfoSOSA). As there is a change in value of memory GM_LED01, the lamp associated with the link setting turns on.



1.7.9 Merge Test

Start the C++ application and connect with the IS-APP.

How to Start

Start up another console and connect with the console running the IS-APP.

Run the executable file **tutorial**.

Example:

```
$ export DISPLAY=:0
$ cd /mnt/user/
$ chmod 755 tutorial
$ ./tutorial
```

* When running the program, the first line sets the display destination.
(Set display number 0 to the **DISPLAY** environment variable)
* The second line moves to the user folder.
* The third line sets up authorization for the executable file **tutorial**.

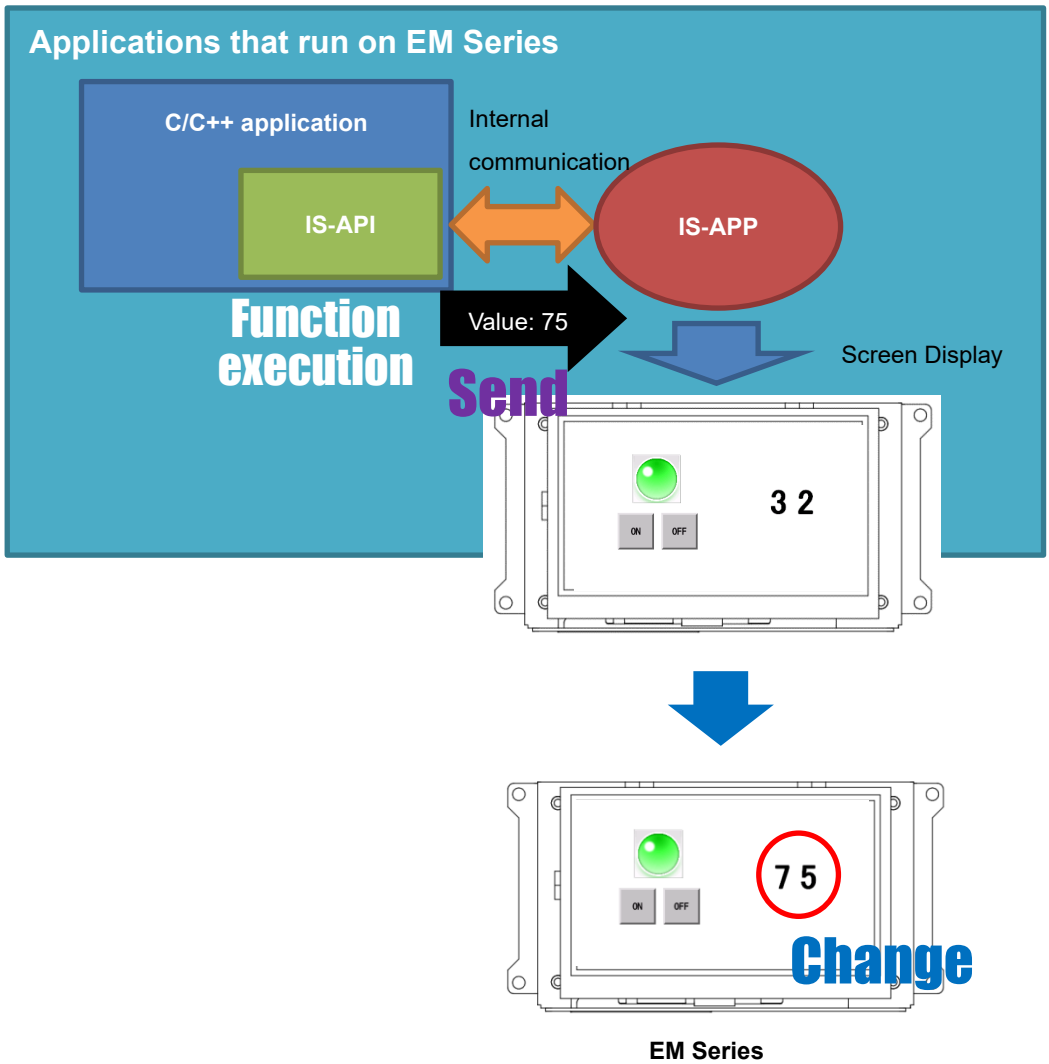
When the IS-APP connection is successful, the console screen displays as shown below.

```
Connecting...OK
Input a numeric value and press [ENTER].
```

Operation Check

Input values to the console to change values on the screen

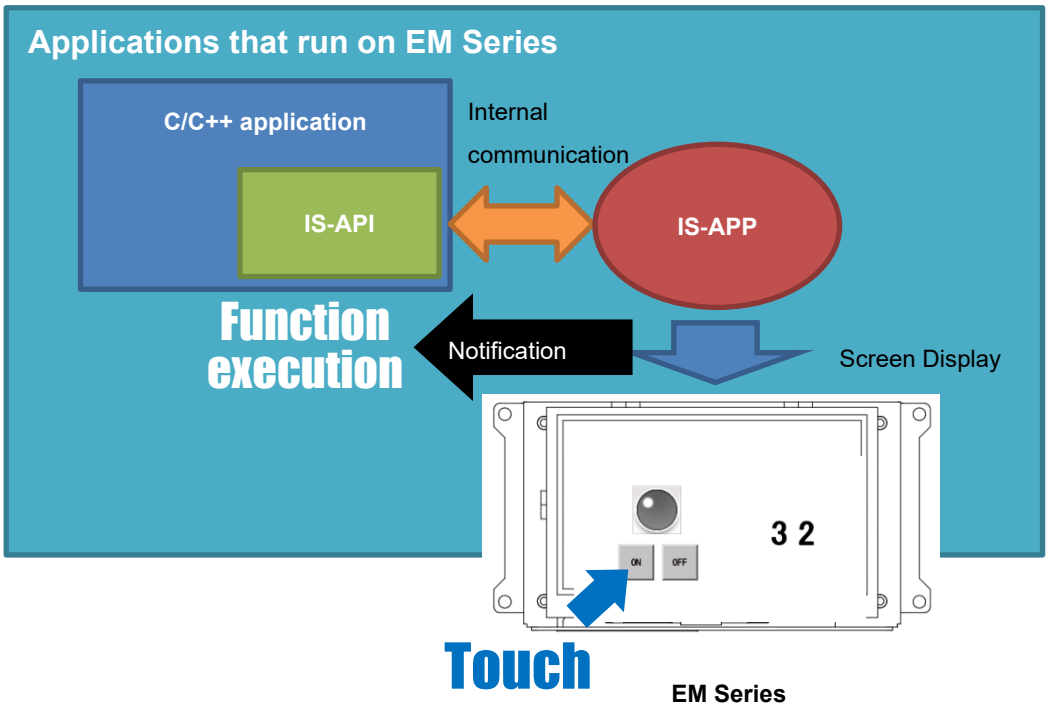
Input values to the console and press the [ENTER] key to change the value displayed on the LCD screen (IS-APP).
Using the IS-API, a C++ application is overwriting the value in IS-APP memory GM_VOL01. As there is a change in value of memory GM_VOL01, the number display control associated with the link setting is also changed.



EM Series

Touch a button on the screen to display the status on the console

Touch the ON/OFF button on the LCD screen (IS-APP) to display the status on the console.
When the button is pressed IS-APP sends the status, and the IS-API runs the registered function. With this feature, when the IS-APP button is pressed you can run any command.



1.7.10 Automatic start of IS-APP

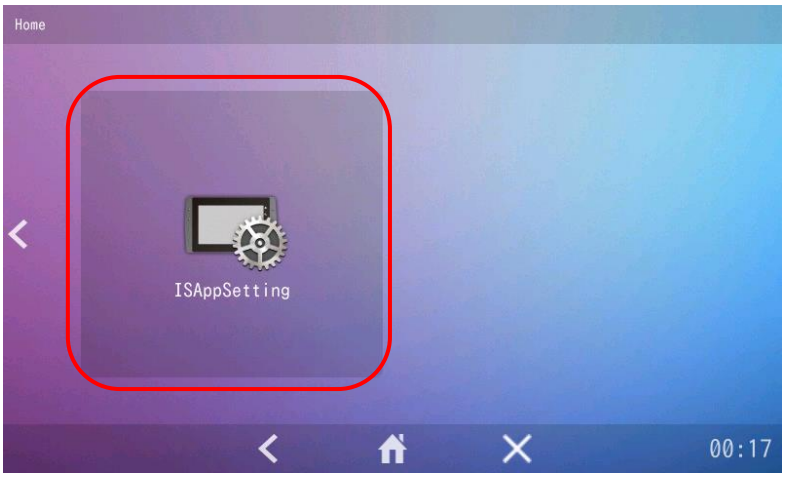
This section describes how to automatically start IS-APP when the power is turned on.

Create IS-APP startup script

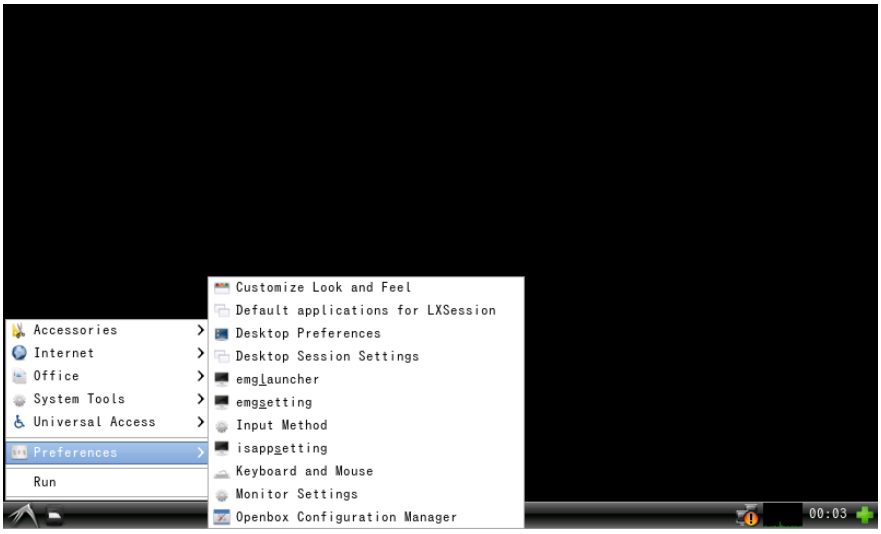
IS-APP specifies communication settings, etc. as command line arguments at startup.
ISAPP SETTING allows you to set command line arguments using the GUI.
(Create a startup script with command line arguments set)

From the EMG Launcher or the Start menu click [ISAPP SETTING].

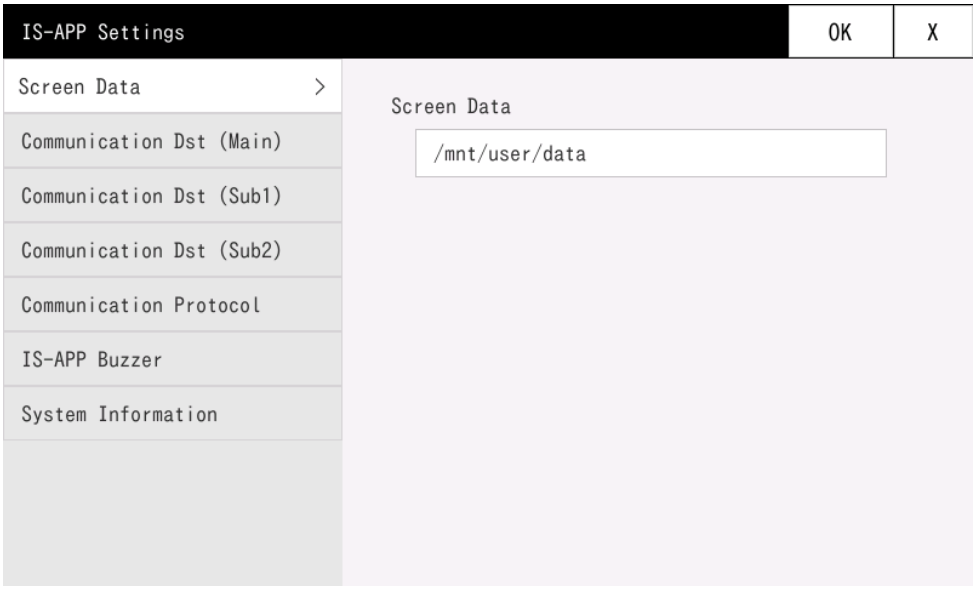
[EMG Launcher] - [ISAppSetting]



[Start menu] - [Preferences] - [isappsetting]



After completing the settings, touch the OK button at the top right to update the startup script file.
Here, leave the default settings and close by clicking the × button.



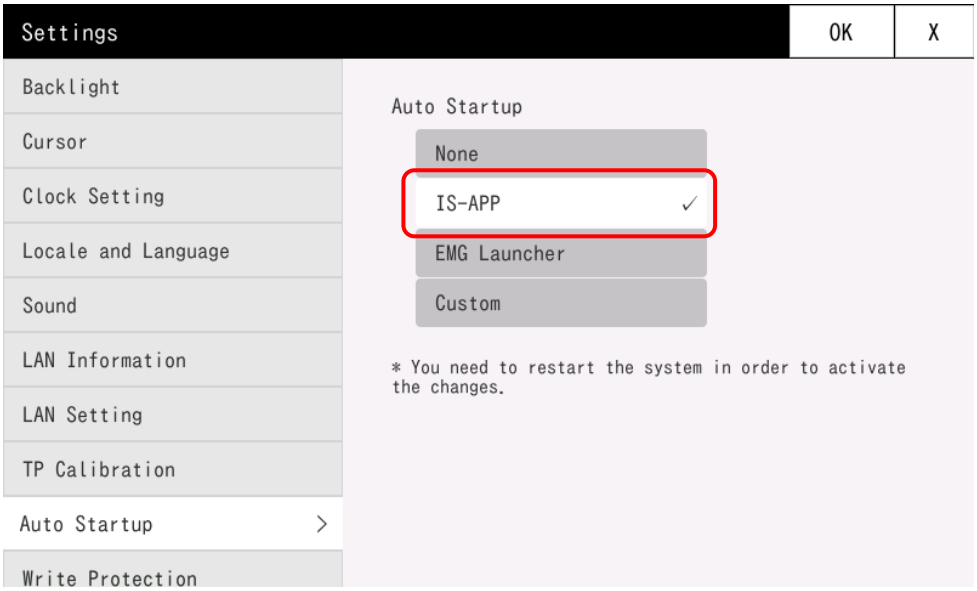
The start script is created in the following directory.

/mnt/user/isapp_run.sh

For each setting, please refer to "2.3 ISAPP SETTING".

Auto-start setting

Settings to automatically run the startup script and start IS-APP when the power is turned on can be configured using the system settings tool.



For more information, refer to the separate document “EM Series Tool Manual”.

1.7.11 Automatic startup of C++ applications

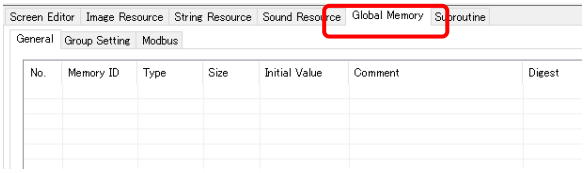
This article describes how to automatically start a C++ application when IS-APP starts.

Global Memory Settings

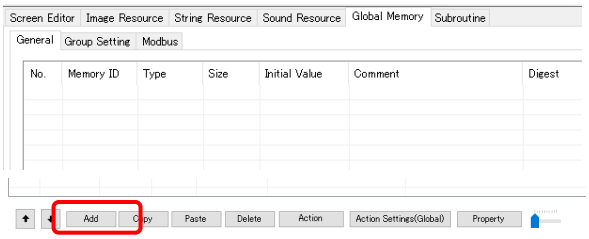
Open the project created in InfoSOSA Builder.

Creating a Memory

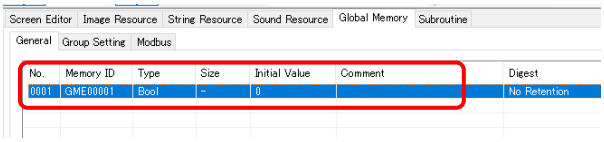
Create a memory in the same way. Set the properties of the second memory as follows:



Select the "Global Memory" tab.

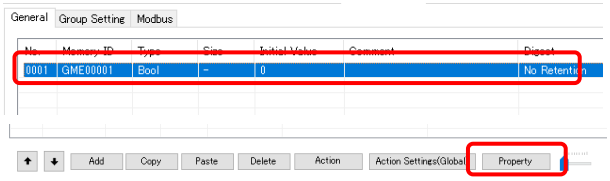


Click the “Add” button.

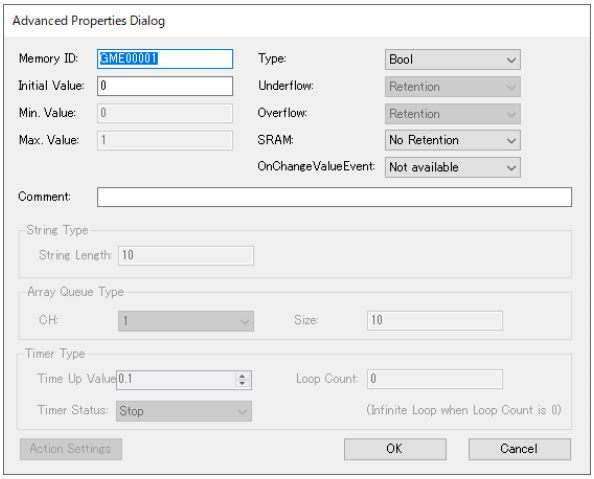


A memory was created.

Property settings



Select the memory and click the Properties button.



Change the properties to match the table.

Properties	Value	Property meaning
Memory ID	S_CMD	ID to identify the memory
Type	String	Memory Type
String Length	64 *	Maximum number of characters in a command
Initial Value	(Command to be executed)*	Command to be executed

*Please change according to your environment.
Example: If the file name of the C++ application is tutorial and it is placed in /mnt/user/
/mnt/user/tutorial

Creating a second memory

Create a memory in the same way. Set the properties of the second memory as follows:

Properties	Value	Property meaning
Memory ID	S_PID	ID to identify the memory
Type	Double Word	Memory Type

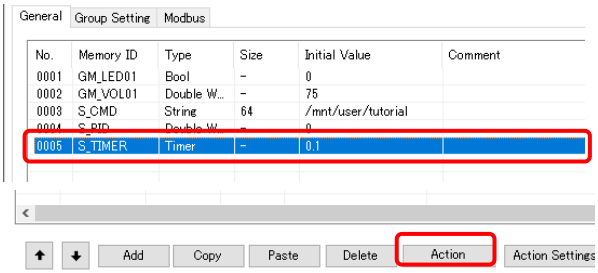
Creating the third memory

Create a memory in the same way. Set the properties of the third memory as follows:

Properties	Value	Property meaning
Memory ID	S_TIMER	ID to identify the memory
Type	Timer	Memory Type
Time Up Value	0.1*	The number of seconds before the action is performed
Loop Count	1	Repeat count
Timer Status	Start	Initial state of timer operation

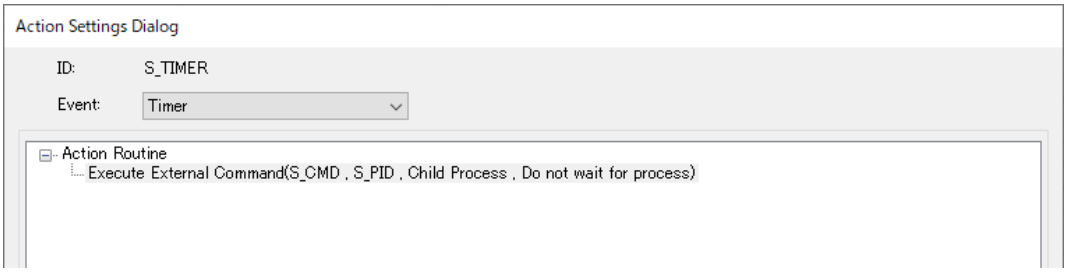
*Adjust the value if you want to delay the startup of your C++ application. (Unit: seconds)

Timer type memory action settings



Select the "S_TIMER" memory and click the "Action" button.

Set the action as follows:



Items	Setting
Action Group	External Command
Action	Execute External Command
Command Memory Type	Global Memory
Command Memory ID	S_CMD
PID Storage Memory Type	Global Memory
PID Storage Memory ID	S_PID
Ext. Command to Process relation	Child Process
IS-APP wait behaviour to Process	Do not wait for process

1.7.12 Other Features

The tutorial is complete.
For additional features that are not used in the tutorial, please refer to the following.

InfoSOSA Builder

InfoSOSA Builder has other kinds of parts and actions, and without any programming Builder allows you to easily set up conditions and branch processes.
For more information, please refer to the "InfoSOSA Reference Manual" or "InfoSOSA Builder Operation Manual".

IS-APP

IS-APP uses run time command-line arguments to set the following.

- Communication settings (communication with external devices)
- Buzzer
- Sound

For more information, please refer to "2.1 IS-APP Specifications".

IS-API

The IS-API also has functions to change display screens and to batch write multiple memory.
For more information, please refer to "2.2 IS-API Specifications".

ISAPP SETTING

The EM Series is installed with an auxiliary tool ISAPP SETTING which you can use to start IS-APP automatically without using the console.
If you do not make your own C/C ++ application, you can use this tool to start IS-APP without connecting the console.
For more information, please refer to "2.3 ISAPP SETTING".

2. Reference

Chapter Contents

2.1 IS-APP Specifications

2.1.1 Summary

IS-APP is an application for displaying screens created in InfoSOSA Builder.
Create screen data in InfoSOSA Builder, and transfer to EM Series.
(InfoSOSA Builder is screen drawing software that runs on Windows®)
With run time command-line arguments, you can define the folder where display screen data is stored, devices to use, and so on.
For specific usage, please refer to "1.7 IS-APP HMI Development Process".

2.1.2 Executable file name

Executable file
is_app

Transfer to the EM series main unit is required.

For the transfer method, please refer to "1.6 Update of IS-APP/IS-API/IS-APP SETTING".

2.1.3 Number of instances

The number of IS-APP you can run at one time is 1.
You cannot display two of more projects at the same time.

2.1.4 Supported hardware

This application runs only on the following hardware.

Model
EMG7-***A8-****-**7
EM8-***A7-****-**7
EMG8-***A7-****-**7

2.1.5 Number of available system fonts

Depending on the product model, the number of usable system fonts and the availability of Simplified Chinese fonts are different. If the screen data's number of fonts exceeds the limit, or using Simplified Chinese fonts with models not having this availability, the IS-APP will not be executed.

Model	Number of selectable font types	Simplified Chinese availability
EMG7-***A8-****-**1*7**	5	○
EM8-***A7-****-**2*7**	1	×
EMG8-***A7-****-**2*7**	1	×

2.1.6 Command line arguments

Screen data folder

Specify the screen data folder to display.

Argument name

-r or --rscdir

Format

-r <PATH>

Parameter	Contents
<PATH>	Specify the screen data path.

Example:
is_app –r /mnt/user/data

Default Value

This argument is mandatory. You cannot omit its use.

Communication Settings

Specify the IS-APP communication settings.

Up to 3 communication interfaces can be specified.
The combinations that can be specified are as follows.

Main argument	Sub argument	Sub argument
Serial	(unused)	(unused)
LAN	(unused)	(unused)
IS-API	(unused)	(unused)
Serial	LAN	(unused)
Serial	IS-API	(unused)
LAN	Serial	(unused)
IS-API	Serial	(unused)
Serial	Serial	(unused)
Serial	Serial	LAN
Serial	Serial	IS-API
LAN	Serial	Serial
IS-API	Serial	Serial

The order of Sub arguments does not affect the processing.

LAN and IS-API cannot be used at the same time.
When connecting multiple LANs (including IS-API), specify TCP/IP server as the LAN connection protocol.

If there are insufficient arguments, default values will be set for the missing items.

If you specify the serial interface, you cannot specify the same device file.

The event notification destination at the time of executing the action "Notify event to Host" or "Notify value to Host" is the interface specified in the Main argument. The Sub parameter is not notified.
Example: When communicating with serial or IS-API (events are notified to serial only)
is_app -r /mnt/user/data -s /dev/com1 rs232c 115200 none none -a 51111

The output destination when executing the action "Output string memory to Host" is output only when serial is specified as the Main argument. It is not output when LAN, IS-API is specified, or when the Sub argument is specified.

“Start Message” is sent to all interfaces.* Excluding TCP/IP server
[Start Message (Serial)]
{STX}00s0000000800010000003DC{ETX}
[Start Message (LAN)]
s000000800010000

* When connecting to IS-API, the character code setting (SI03) is automatically set to "Unicode (UTF-16LE)". When using IS-API, do not set it to "Shift JIS". IS-API will not work properly.

* The character code setting applies to all communication interfaces. When using IS-API, use "Unicode (UTF-16LE)" for serial and LAN communication.

Argument name

Serial: -s or --sio
LAN: -l or --lan
IS-API: -a or -api *default

Format (serial)

RS232C: *Default
-s <device file> rs232c <communication speed> <parity> <flow control>

RS422:
-s <device file> rs422 <communication speed> <parity>

RS485:
-s <device file> rs485 <communication speed> <parity> <address> <retry count> <retry wait time> <transmission timeout>

Parameter	Contents
<device file>	Specify the serial port device file. Default: SIO1 device file is automatically set. SIO1：./dev/com1 SIO2：./dev/com2
<communication speed>	Specify the communication speed. 4800: 4800bps 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
<flow control>	For RS232C communication, specify the flow control . none: No flow control *Default
<address>	For RS485 communication, specify the node address. 1 to 31 Default: 1
<retry count>	For RS485 communication, specify the number of times a transmission is attempted after a transmission failure. 0 to 10 Default: 3
<retry interval>	For RS485 communication, specify the interval after a transmission failure before a transmission is attempted again. 0 to 1000 (10 ms unit) Default: 100
<transmission timeout>	For RS485 communication, specify the maximum time to wait when other equipment is busy transmitting. 0 to 3000 (10 ms unit) Default: 500

Example:
RS232C
is_app –r /mnt/user/data –s /dev/com1 rs232c 115200 none none

RS422
is_app –r /mnt/user/data –s /dev/com2 rs422 115200 none

RS485
is_app –r /mnt/user/data –s /dev/com2 rs485 115200 none 1 3 100 500

Format (LAN)

UDP/IP: *Default
-l udp <communication destination address> <communication destination port>

TCP/IP Client:
-l tcp <communication destination address> <communication destination port> <connection interval>

TCP/IP Server:
-l tcps <Event notification port> <Normal port>

Parameter	Contents
<communication destination address>	Specify the communication destination IP address. Default: 192.168.0.100
<communication destination port>	Specify the communication destination port number. 0 to 65535 Default: 51111
<connection interval>	Specify the communication destination TCP/IP server connection interval. 0 to 99 (seconds) Default: 5
<Event notification port>	For TCP/IP server, specify the port number that accepts connections from TCP/IP clients. The maximum number of TCP/IP clients that can connect to this port is "1". The event is notified only to the clients connected to this port * 0 to 65535 Default: 51111 *If the LAN interface is set to the Sub argument, notification will not be sent to this port either.
<Normal port>	For TCP/IP server, specify the port number that accepts connections from TCP/IP clients. The maximum number of TCP/IP clients that can connect to this port is "3". No events will be notified to clients connected to this port. 0 to 65535 Default: 51115

Example:
UDP/IP:
is_app –r /mnt/user/data –l udp 192.168.0.100 51111

TCP/IP Client:
is_app –r /mnt/user/data –l tcp 192.168.0.100 51111 5

TCP/IP Server:
is_app –r /mnt/user/data –l tcps 51111 51115

Format (IS-API)

-a <communication destination port>

Parameter	Contents
<communication destination port>	Specify the communication destination port number. 0 to 65535 Default: 51111

Example:
is_app –r /mnt/user/data –a 51111

Default Value

If this is not defined, uses the following default settings.

Parameter	Contents
Communication Type	IS-API
Communication Destination Port	51111

Communication Protocol

Specify the IS-APP communication protocol.
This setting is enabled when the communication setting is serial or LAN.
If there is a missing argument, the default value is set for the missing item.

【Note】
-The setting of this argument is applied to all communication destinations.
-This parameter cannot be used when connecting to IS-API.
"InfoSOSA standard protocol/immediate notification" is set.

Argument name

-p or --protocol

Format

InfoSOSA Normal Protocol: *Default
-p infososa <notification method>

Normal Protocol:
-p standard <notification method> <monitoring time> <retry count>

Parameter	Contents
<notification method>	Specify the notification method. immediate: Immediate notification *Default polling: Upon request
<monitoring time>	Specify the time the normal protocol waits until resending a transmission. 1 to 5 (seconds) Default: 1
<retry count>	Specify the number of times the normal protocol resends a transmission. 0 to 3 Default: 3

Example:
is_app –r /mnt/user/data –s –p standard immediate 3 1

Default Value

If this is not defined, uses the following default settings.

Parameter	Contents
Communication Protocol	InfoSOSA Protocol
Notification Method	Immediate Notification

Buzzer

Enables the IS-APP buzzer function.
To use the buzzer in IS-APP, disable the system touch sound beforehand.
If you do not enable the buzzer function with this argument, the touch sound or buzzer sound set up in InfoSOSA Builder is disabled.

Argument name

-b or --buzzer

Format

-b <device file>

Parameter	Contents
<device file>	Specify the buzzer device file. Default: The buzzer device file (/dev/buzzer) is automatically set. * Normally you do not need to specify it.

Example:
is_app –r /mnt/user/data –b

Default Value

If this is not defined, uses the following default settings.

Parameter	Contents
Enable / Disable	Disable

Sound

Change the IS-APP sound output destination.

Argument name

-o or --sound

Format

-o <device name> <mixer name> <volume name>

Parameter	Contents
<device name>	Specify the name of the device for sound output.
<mixer name>	Specify the name of the mixer for sound output.
<volume name>	Specify the name of the volume for sound output.

Example:
is_app –r /mnt/user/data –o default default PCM

Default Value

If this is not defined, uses the following default settings.
Disabled if the device does not exist when IS-APP is launched.

Parameter	Contents
Enable / Disable	Enable
<device name>	default
<mixer name>	default
<volume name>	PCM

Display Version

On the console, displays the IS-APP version.
When this argument is specified, IS-APP is not started.

Argument name

-v or --version

Format

-v

Example:
is_app –v

Help

On the console, displays help for IS-APP arguments.
When this argument is specified, IS-APP is not started.

Argument name

-h or --help

Format

-h

Example:
is_app –h

2.2 IS-API Specifications

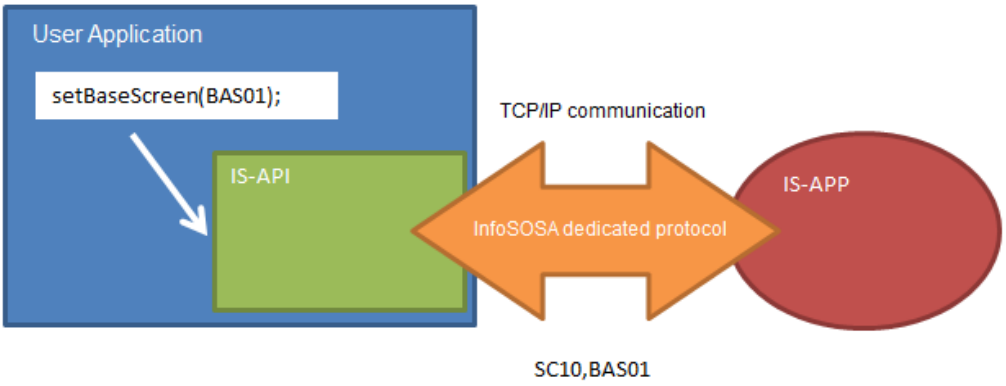
2.2.1 Summary

IS-API is a dedicated protocol for communication between host devices and InfoSOSA.
You can use this API, which is responsible for the communication part of the InfoSOSA dedicated protocol, with C/C++ applications to change the IS-APP display screen, get memory values, and so on.

IS-APP and IS-API are connected within the same device with TCP/IP communication.
To use IS-API, you need to start IS-APP communication settings with IS-API.
Please refer to "2.1.6 Command line arguments - Communication Settings" for IS-APP communication settings.

Each function corresponds to an InfoSOSA host communication command. Please refer to the "InfoSOSA Reference Manual" for specifications on host communication commands.

For specific usage, please refer to "1.7 IS-APP HMI Development Process".



2.2.2 Library/Header files

To create C/C++ applications that use the IS-API, you need to copy the library file and header files to the software development environment, and copy the library file to the EM Series unit.

Library file
libisapi.so

Header files
ClsApi.h CIsComString.h CIsComVariant.h CIsComVariantList.h

Transfer to the EM series main unit and copy to the development environment are required.

Copying to development environment is required

The library/header files are located in the \\software\IS-API\ folder in the InfoSOSA Development Kit Data.

The library file varies depending on your model.
Use the file in the folder that corresponds to your model.

Model	Folder
EMG7-***A8-****-**7	IS-APP-A8
EM8-***A7-****-**7	IS-APP-A7
EMG8-***A7-****-**7	IS-APP-A7

For the transfer method, please refer to "1.6 Update of IS-APP/IS-API/IS-APP SETTING".

2.2.3 Number of simultaneous connections

The number of IS-APP you can connect with IS-API is 1.

2.2.4 Character code

Use UTF-8 character encoding for the strings in API arguments.

2.2.5 Real-time signal

IS-API uses the following real-time signal.

Definition	ID	Operation
SIGRTMIN	34	Terminates all IS-API threads

2.2.6 API list

■ **CIsApi Class**

Function	Contents
CIsApi(int hPort)	Define the port number to use with the IS-API and create an object. In this class, IS-API is a TCP/IP server.
CIsApi_c(int hPort, unsigned int startTimeout)	Define the port number to use with the IS-API and create an object. In this class, IS-API is a TCP/IP client.
bool Start()	Start IS-API and establish communication with the IS-APP.
bool Stop()	Stop the IS-API, and disconnect from IS-APP communication.
bool IsRunning()	Gets the IS-API operation status.
bool Test()	Test access to IS-APP to check if it is working properly.
int getApiVersion()	Gets the IS-API version.
bool getInfososaVersion(std::string* _isVer, std::string* _userVer)	Gets the connected IS-APP version.
bool setBaseScreen(std::string* _screenID)	Changes the IS-APP display screen.
bool getBaseScreen(std::string* _screenID)	Gets the ID of the currently displayed IS-APP screen.
bool showPopupScreenA(std::string* _screenID, int _x, int _y)	Displays popup screen A in the IS-APP defined coordinates.
bool showPopupScreenB(std::string* _screenID, int _x, int _y)	Displays popup screen B in the IS-APP defined coordinates.
bool hidePopupScreenA()	Closes the IS-APP popup screen A.
bool hidePopupScreenB()	Closes the IS-APP popup screen B.
bool getPopupScreen(int* _a_state, int* _b_state)	Gets the IS-APP popup screen display status.
bool setBuzzer(int _freq, int _msec)	Play the buzzer at the defined frequency and buzzer length.
bool getBuzzer(int* _state)	Gets the IS-APP buzzer play status.
bool setProperty(std::string* _propertyID, CIsComVariant& _var)	Sets up IS-APP parts properties.
bool getProperty(std::string* _propertyID, CIsComVariant& _var)	Gets the IS-APP parts properties.
bool setGroupMemory(std::string* _groupID, CIsComVariantList& _varList)	Sets values to IS-APP Global Memory group.
bool getGroupMemory(std::string* _groupID, CIsComVariantList& _varList)	Gets values from IS-APP Global Memory group.
bool callSubRoutine(std::string* _subroutineID)	Runs special subroutine set up in IS-APP
bool registerCallback(std::string* _eventID,	Register the callback function that runs when a button on

Function	Contents
EventCallbackFunc _func)	the IS-APP screen is pressed.
bool unregisterCallback(std::string* _eventID)	Releases the registered callback function.

■ **ClsComVariant Class**

Function	Contents
bool SetValue(int value)	Set a value to the ClsComVariant class.
int GetValue()	Get a value from the ClsComVariant class.

■ **ClsComString Class**

Function	Contents
bool SetString(int bytes, char* str)	Set a string to the ClsComString class.
char* GetStringBuffer()	Get a pointer for the string set to the ClsComString class.
int GetStringLength()	Get the number of bytes for the string set to the ClsComString class.

■ **ClsComVariantList Class**

Function	Contents
bool Add(ClsComVariant& elem)	Add the specified ClsComVariant class to the end element of the ClsComVariantList class.
ClsComVariant* GetElem(int index)	Get the specified element from the ClsComVariantList class.
int Size()	Get the number of elements registered in the ClsComVariantList class.
bool Clear()	Delete everything in the ClsComVariantList class.
ClsComVariant* ReplaceElem(int index, ClsComVariant& elem)	Set the specified ClsComVariant class to the specified position in the ClsComVariantList class.

2.2.7 ClsApi

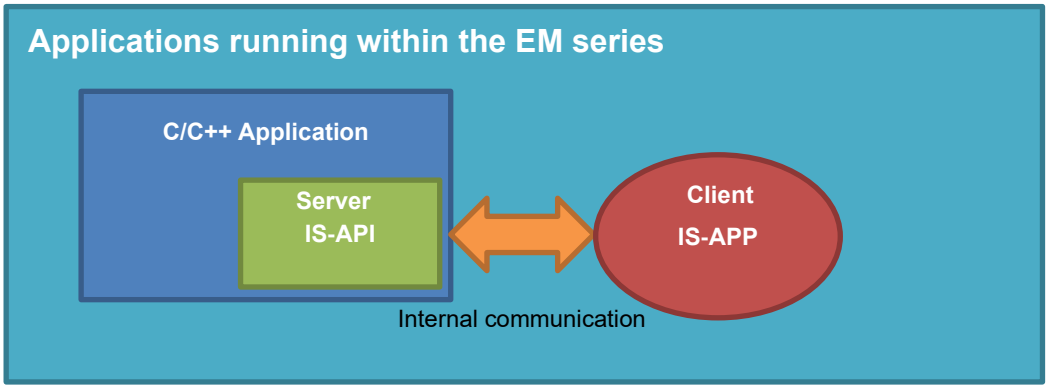
Access IS-APP via this class.
*The methods of this class are not thread safe. Please execute from one thread.

If the IS-APP side is a TCP/IP client, use the "CIsApi" class.
If the IS-APP side is a TCP/IP server, use the "CIsApi_c" class.

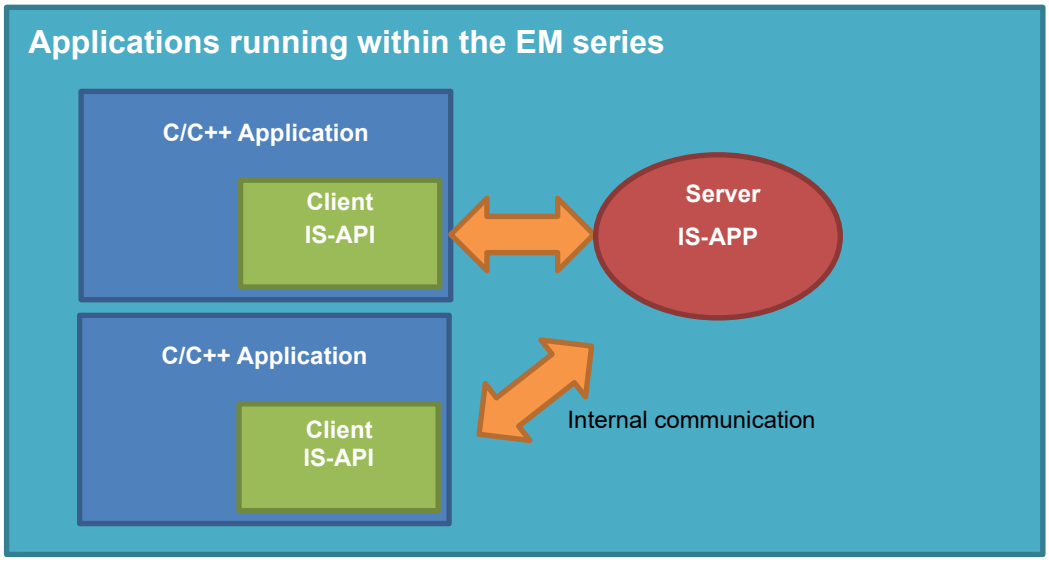
[Correspondence table]

IS-APP argument	IS-API Class
-a	CIsApi
-l tcps	CIsApi_c

If the IS-APP side is a TCP/IP client, only one IS-API can connect to IS-APP.



Multiple IS-APIs can be connected if the IS-APP side is a TCP/IP server.



Header file

ClsApi.h

Function

Constructor

Create an object by specifying the port number and timeout time.

■ **Format**

CIsApi(int hPort)
CIsApi_c(int hPort, unsigned int startTimeout)

■ **Parameters**

Parameter	Description
int hPort	CIsApi：Port number used by IS-API CIsApi_c：Port number used by IS-API *The default is 51111.
unsigned int startTimeout	CIsApi： Cannot be specified. After executing Start(), it waits until the connection with IS-APP is completed. CIsApi_c： Set the timeout time when connecting to IS-APP when you execute Start(). The unit is seconds and the value range is 0 to 300. If it is 0, it will connect once and if it fails, it will time out immediately. *The default is 15 seconds.

■ **Example (CIsApi)**

```
int port = 51111;
pIsApi = new ClsApi(port);
```

■ **Example (CIsApi_c)**

```
int port = 51111;
unsigned int timeout = 15;
pIsApi = new CIsApi_c(port, timeout);
```

Start

Start IS-API and establish communication with the IS-APP.

■ Format
bool Start()

■ Parameters
None

■ Return Value
true: Successful
false: Failed

■ Example
```
if(!pIsApi->Start()){
   std::cout << "failed to start isapi" << std::endl;
   return false;
}
```

IsRunning

Gets the IS-API operation status. You can use IS-API functions only while this application is running (when the return value of this function is true).

■ Format
bool IsRunning()

■ Parameters
None

■ Return Value
true: Running
false: Stopped

■ Example
```
if(!pIsApi->IsRunning()){
   std::cout << "library not working" << std::endl;
   return false;
}
```

Test

Runs test access for IS-APP.

■ Format
bool Test()

■ Parameters
None

■ Return Value
true: Successful
false: Failed

■ Example
```
if(!pIsApi->Test()){
   std::cout << "library not communicating" << std::endl;
   return false;
}
```

Stop

Close the IS-APP connection and exit IS-API.

■ Format
bool Stop()

■ Parameters
None

■ Return Value
true: Successful
false: Failed

■ Example
```
if(pIsApi){
   pIsApi->Stop();
   delete pIsApi;
}
```

getApiVersion

Gets the IS-API version.

■ Format
int getApiVersion()

■ Parameters
None

■ Return Value
IS-API version (XYY)
X = major version
YY = minor version

Example:
For version 1.0.0, returns 100.

■ Example
std::cout << "IS-API Version:" << pIsApi->getApiVersion() << std::endl;

getInfososaVersion

Gets the connected IS-APP version.

■ Corresponding InfoSOSA Host communication command
SI02

■ Format
bool getInfososaVersion(std::string* _isVer, std::string* _userVer)

■ Parameters

Parameter	Description
std::string* _isVer	IS-APP version
std::string* _userVer	User version set to the screen data displayed on IS-APP. * Set the user version with InfoSOSA Builder. For more information, refer to the "InfoSOSA Builder Operation Manual".

■ Return Value
true: Successful
false: Failed

■ **Example**

```
std::string isVer;
std::string userVer;
pIsApi->getInfososaVersion(&isVer,&userVer);
std::cout   << "IS-APP Version = " << isVer << std::endl;
std::cout   << "User Version = " << userVer << std::endl;
```

setBaseScreen

Change the IS-APP display to the screen associated with the specified base screen ID.

■ **Corresponding InfoSOSA Host communication command**

SC10

■ **Format**

bool setBaseScreen(std::string* _screenID)

■ **Parameters**

Parameter	Description
std::string* _screenID	ID of base screen to change to

■ **Return Value**

true: Successful
false: Failed

■ **Example**

```
std::string screenID("BAS00002");
pIsApi->setBaseScreen(&screenID);
```

getBaseScreen

Gets the ID of the displayed IS-APP base screen.

■ Corresponding InfoSOSA Host communication command
SC11

■ Format
bool getBaseScreen(std::string* _screenID)

■ Parameters

Parameter	Description
std::string* _screenID	ID of currently displayed base screen

■ Return Value
true: Successful
false: Failed

■ Example
```
std::string screenID;
pIsApi->getBaseScreen(&screenID);
std::cout << "BASE=" << screenID << std::endl;
```

showPopupScreenA

Displays the specified popup screen A in IS-APP.

■ Corresponding InfoSOSA Host communication command
SC13

■ Format
bool showPopupScreenA(std::string* _screenID, int _x, int _y)

■ Parameters

Parameter	Description
std::string* _screenID	ID of pop-up screen to display
int _x	X coordinate for pop-up screen
int _y	Y coordinate for pop-up screen

■ Return Value
true: Successful
false: Failed

■ Example

```
std::string screenID("POPA0001");
int x= 100;
int y= 50;
pIsApi->showPopupScreenA(&screenID,x,y);
```

showPopupScreenB

Displays the specified popup screen B in IS-APP.

■ Corresponding InfoSOSA Host communication command
SC14

■ Format
bool showPopupScreenB(std::string* _screenID, int _x, int _y)

■ Parameters

Parameter	Description
std::string* _screenID	ID of pop-up screen to display
int _x	X coordinate for pop-up screen
int _y	Y coordinate for pop-up screen

■ Return Value
true: Successful
false: Failed

■ Example
```
std::string screenID("POPB0001");
int x= 180;
int y= 135;
pIsApi->showPopupScreenB(&screenID,x,y);
```

hidePopupScreenA

Closes the IS-APP popup screen A.

■ **Corresponding InfoSOSA Host communication command**
SC15

■ **Format**
bool hidePopupScreenA()

■ **Parameters**
None

■ **Return Value**
true: Successful
false: Failed

■ **Example**
pIsApi->hidePopupScreenA();

hidePopupScreenB

Closes the IS-APP popup screen B.

■ **Corresponding InfoSOSA Host communication command**
SC16

■ **Format**
bool hidePopupScreenB()

■ **Parameters**
None

■ **Return Value**
true: Successful
false: Failed

■ **Example**
pIsApi->hidePopupScreenB();

getPopupScreen

Gets the IS-APP popup screen display status.

■ Corresponding InfoSOSA Host communication command
SC17

■ Format
bool getPopupScreen(int* _a_state, int* _b_state)

■ Parameters

Parameter	Description
int* _a_state	Display state of Pop-up Screen A (0:OFF / 1:ON)
int* _b_state	Display state of Pop-up Screen B (0:OFF / 1:ON)

■ Return Value
true: Successful
false: Failed

■ Example
```
int a,b;
pIsApi->getPopupScreen(&a, &b);
std::cout << "POPA=" << a << " POAB=" << b << std::endl;
```

setBuzzer

Sends an instruction to IS-APP to emit the buzzer.

* When the buzzer is disabled on IS-APP, the return value is true even though the buzzer does not sound.

■ Corresponding InfoSOSA Host communication command
BZ01

■ Format
bool setBuzzer(int _freq, int _msec)

■ Parameters

Parameter	Description
int _freq	Buzzer Frequency (Hz) Range: 500 to 5000
int _msec	How long the buzzer sounds (ms) Range: 100 to 10000 (increments of 100)

■ **Return Value**
true: Successful
false: Failed

■ **Example**
```
int freq=500;
int msec=100;
pIsApi->setBuzzer(freq,msec);
```

getBuzzer

Gets the IS-APP buzzer play status.

* When the buzzer is disabled on IS-APP, the value is 0 (off).

■ **Corresponding InfoSOSA Host communication command**
BZ02

■ **Format**
bool getBuzzer(int* _state)

■ **Parameters**

Parameter	Description
int* _state	Buzzer Status (0:OFF / 1:ON)

■ **Return Value**
true: Successful
false: Failed

■ **Example**
```
int state;
pIsApi->getBuzzer(&state);
if(state == 0){
   std::cout << "Buzzer=OFF" << std::endl;
}
else{
   std::cout << "Buzzer=ON" << std::endl;
}
```

setProperty

You can change the IS-APP parts and memory properties. You can do things such as change memory values, and change text colors.
The property ID format and setting values conform to InfoSOSA communication specifications.
Please refer to the "InfoSOSA Reference Manual" for specifications on InfoSOSA communication.

■ Corresponding InfoSOSA Host communication command
PA01

■ Format
bool setProperty(std::string* _propertyID, CIsComVariant& _var)

■ Parameters

Parameter	Description
std::string* _propertyID	Property ID of the item to change * Set up the ID with the format: [Affiliation ID].[Part/Memory ID].[Property ID] For more information, please refer to the "InfoSOSA Reference Manual".
CIsComVariant& _var	Value and string to set * Use UTF-8 character encoding for strings. * “,” comma (0x2C), and control characters cannot be used.

■ Return Value
true: Successful
false: Failed

■ Example (numeric value)

```
std::string propertyID;
CIsComVariant var; // when setting a numeric value create a CIsComVariant type variable
propertyID="@GLBMEM.GME00001.VALUE";
int value = 100;
var.SetValue(value); //set a numeric value to CIsComVariant type
pIsApi->setProperty(&propertyID, var);
```

■ Example (String)

```
std::string propertyID;
std::string strtmp;
CIsComString var; // when setting a string create a CIsComString type variable
propertyID="@GLBMEM.GME00002.TEXT";
strtmp = "test";
var.SetString(strtmp.size(),(char*)strtmp.c_str()); //set string to CIsComString type
pIsApi->setProperty(&propertyID, (CIsComVariant&)var); // cast to CIsComVariant type
```

getProperty

You can get IS-APP parts and memory properties.
The property ID format and setting values conform to InfoSOSA communication specifications.
Please refer to the "InfoSOSA Reference Manual" for specifications on InfoSOSA communication specifications.

■ Corresponding InfoSOSA Host communication command
PA02

■ Format
bool getProperty(std::string* _propertyID, CIsComVariant& _var)

■ Parameters

Parameter	Description
std::string* _propertyID	Property ID of the item to get * Set up the ID with the format: [Affiliation ID].[Part/Memory ID].[Property ID] For more information, please refer to the "InfoSOSA Reference Manual".
CIsComVariant& _var	Retrieved value, string * Uses UTF-8 character encoding for strings.

■ Return Value
true: Successful
false: Failed

■ Example (numeric value)

```
std::string propertyID;
CIsComVariant var; // when getting a numeric value create a CIsComVariant type variable
propertyID="@GLBMEM.GME00001.VALUE";
pIsApi->getProperty(&propertyID, var);
std::cout << propertyID << "=" << var.GetValue() << std::endl;
```

■ Example (String)

```
std::string propertyID;
std::string recv_str;
CIsComString var; // when getting a string create a CIsComString type variable
propertyID="@GLBMEM.GME00002.TEXT";
pIsApi->getProperty(&propertyID, (CIsComVariant&)var); // cast to CIsComVariant type
recv_str=std::string(var.GetStringBuffer(),var.GetStringLength());
std::cout << propertyID << "=" << recv_str << std::endl;
```

setGroupMemory

Sets values to IS-APP Global Memory group.
The Global Memory group must be created in InfoSOSA Builder beforehand.
Please refer to the "InfoSOSA Reference Manual" for information on Global Memory group.

■ Corresponding InfoSOSA Host communication command
PA05

■ Format
bool setGroupMemory(std::string* _groupID, CIsComVariantList& _varList);

■ Parameters

Parameter	Description
std::string* _groupID	Group ID that is target of set value operation
CIsComVariantList& _varList	Stored list data of the value or string to set * Use UTF-8 character encoding for strings.] *The string cannot contain “,” comma (0x2C), or control characters.

■ Return Value
true: Successful
false: Failed

■ Example

```
std::string groupID;
std::string strtmp;
CIsComVariantList varList;
CIsComVariant var1; // when setting a numeric value create a CIsComVariant type variable
CIsComString var2; // when setting a string create a CIsComString type variable

var1.SetValue(50);
strtmp = "test";
var2.SetString(strtmp.size(),(char*)strtmp.c_str());

groupID = "GRP00001";
varList.Add(var1);
varList.Add((CIsComVariant&)var2); // cast to CIsComVariant type and store

pIsApi->setGroupMemory(&groupID,varList);
```

getGroupMemory

Gets values from IS-APP Global Memory group.
The Global Memory group must be created in InfoSOSA Builder beforehand.
Please refer to the "InfoSOSA Reference Manual" for information on Global Memory group.

■ Corresponding InfoSOSA Host communication command
PA06

■ Format
bool getGroupMemory(std::string* _groupID, CIsComVariantList& _varList);

■ Parameters

Parameter	Description
std::string* _groupID	Group ID of values to get
CIsComVariantList& _varList	List data of the value or string to store * Must by empty. * Uses UTF-8 character encoding for strings.

■ Return Value
true: Successful
false: Failed

■ Example

```
int size;
std::string groupID;
std::string recv_str;
CIsComString* pisstr;
CIsComVariantList varList;

groupID = "GRP00001";
varList.Clear(); //getGroupMemory stored list data is empty
pIsApi->getGroupMemory(&groupID,varList);

std::cout << groupID << "=" ;
size = varList.Size();
for(int i = 0; i < size; ++i){
   // since you do not know if list stored data is numeric or string, use dynamic_cast to differentiate
   pisstr = dynamic_cast <CIsComString*>(varList.GetElem(i));
   if(pisstr){ // when string(CIsComString) dynamic_cast is successful
      recv_str = std::string(pisstr->GetStringBuffer(),pisstr->GetStringLength());
      std::cout << recv_str ;
   }
   else{ // when numeric value CIsComString dynamic_cast does not run
      std::cout << varList.GetElem(i)->GetValue();
   }
   if(i<size-1){
      std::cout << "," ;
   }
}
std::cout << std::endl;
```

callSubRoutine

Runs IS-APP subroutine.
The subroutine must be created in InfoSOSA Builder beforehand.
Please refer to the "InfoSOSA Reference Manual" for information on subroutines.

■ Corresponding InfoSOSA Host communication command
PA07

■ Format
bool callSubRoutine(std::string* _subroutineID)

■ Parameters

Parameter	Description
std::string* _subroutineID	ID of the subroutine to run

■ Return Value
true: Successful
false: Failed

■ Example
```
std::string subroutineID("SUB00001");
pIsApi->callSubRoutine(&subroutineID);
```

registerCallback

Register the callback function that runs when you receive a IS-APP event notification or value notification. By registering a callback function, you can run any function when a button on the InfoSOSA screen is pressed. For value notification, a list of values (CIsComVariantList) is passed to the callback function. For event notification, the list of values is passed in the same way, but the number of elements is 0.
You can register one callback function for one event.
To run event notification or value notification, the event notification / value notification action must be set up in InfoSOSA Builder beforehand.
Please refer to the "InfoSOSA Reference Manual" for information on actions.

■ Format

bool registerCallback(std::string* _eventID, EventCallbackFunc _func)

■ Parameters

Parameter	Description
std::string* _eventID	Target event ID * Set up the event ID with the format: [Affiliation ID].[Part/Memory ID].[Event ID] For more information, please refer to the "Reference Manual".
EventCallbackFunc _func	Function for registration

* Make sure that the function to be registered completes processing immediately. We recommend that you only do things like "setting a processing flag" and "adding an event to the queue" in the function, and doing the actual processing in a separate thread.

■ Return Value

true: Successful
false: Failed

■ Example (Event notification)

```
void event_callback(CIsComVariantList &list) // function to run when the target event ID is received
{
   // process run when the IS-APP button is pressed
   [...]
}

int main()
{
   [...]
   std::string eventID("BAS00001.BTN00001.PRESS"); // target event ID
   pIsApi->registerCallback(&eventID, event_callback);
   [...]
}
```

■ **Example (Value notification)**

```
void event_callback(CIsComVariantList &list) // function to run when the target event ID is received
{
   // process run when the IS-APP button is pressed
   [...]

   // display notified value
   int size;
   CIsComString* pisstr;
   std::string str;

   size = list.Size();
   std::cout << "RECV=" ;
   for(int i = 0; i < size; ++i){
      // since you do not know if notified data is numeric or string, use dynamic_cast to differentiate
      pisstr = dynamic_cast<CIsComString*>(list.GetElem(i));
      if(pisstr){ // when string(CIsComString) dynamic_cast is successful
         str = std::string(pisstr->GetStringBuffer(),pisstr->GetStringLength());
         std::cout << str ;
      }
      else{ // when numeric value CIsComString dynamic_cast does not run
         std::cout << list.GetElem(i)->GetValue() ;
      }
      if(i<size-1){
         std::cout << "," ;
      }
   }
   std::cout << std::endl;
}

int main()
{
   [...]
   std::string eventID("BAS00001.BTN00001.PRESS"); // target event ID
   pIsApi->registerCallback(&eventID, event_callback);
   [...]
}
```

unregisterCallback

Releases the registered callback function.

■ **Format**

bool unregisterCallback(std::string* _eventID)

■ **Parameters**

Parameter	Description
std::string* _eventID	Target event ID

■ **Return Value**

true: Successful

false: Failed

■ **Example**

pIsApi-> unregisterCallback(&eventID);

2.2.8 CIsComVariant

In the IS-API, class for storing values or strings. When setting or getting a string, use the CIsComString derived class.

Header file

CIsComVariant.h

Function

SetValue

Set a value to the CIsComVariant class.

■ Format

bool SetValue(int value)

■ Parameters

Parameter	Description
int value	Value

■ Return Value

true: Successful
false: Failed

■ Example

```
CIsComVariant var;
var.SetValue(100);
```

GetValue

Gets a value from the ClsComVariant class.

■ Format
int GetValue()

■ Parameters
None

■ Return Value
Value

■ Example
```
CIsComVariant var;
std::string propertyID;
propertyID="@GLBMEM.GME00001.VALUE";
pIsApi->getProperty(&propertyID, var);
std::cout << propertyID << "=" << var.GetValue() << std::endl;
```

2.2.9 CIsComString

In the IS-API, class for storing strings. Derived class of the ClsComVariant class. When used as arguments in ClsApi::setProperty(), cast to CIsComVariant class.

Header file

CIsComVariant.h

Inherited class

CIsComVariant

Function

SetString

Set a string to the ClsComString class.

■ Format

bool SetString(int bytes, char* str)

■ Parameters

Parameter	Description
int bytes	String length (bytes)
char* str	Pointer to the string * Use UTF-8 character encoding for strings. *The string cannot contain commas (0x2C), or control characters.

■ Return Value

true: Successful
false: Failed

■ Example

```
CIsComString var;
std::string strtmp;
strtmp = "test";
var.SetString(strtmp.size(),(char*)strtmp.c_str());
```

GetStringBuffer

Get a pointer for string of ClsComString class.
Uses UTF-8 character encoding for strings.

■ Format
char* GetStringBuffer()

■ Parameters
None

■ Return Value
Pointer to string data

GetStringLength

Gets the string length for string of ClsComString class.

■ Format
int GetStringLength()

■ Parameters
None

■ Return Value
String length (bytes)

■ Example

```
CIsComString var;
std::string propertyID;
std::string recv_str;
propertyID="@GLBMEM.GME00001.VALUE";
pIsApi->getProperty(&propertyID, (CIsComVariant&)var);
std::cout << propertyID << "=" << var.GetValue() << std::endl;
recv_str = std::string(var.GetStringBuffer(),var.GetStringLength());
std::cout << propertyID << "=" << recv_str << std::endl;
```

2.2.10 CIsComVariantList

Container class of the ClsComVariant class. Use when handling multiple CIsComVariant data such as group setting and value notification.

Header file

CIsComVariantList.h

Function

Add

Add the ClsComVariant data to the end of the CIsComVariantList class.

■ Format

bool Add(CIsComVariant& elem)

■ Parameters

Parameter	Description
CIsComVariant& elem	CIsComVariant data to add

■ Return Value

true: Successful
false: Failed

GetElem

Gets the ClsComVariant data in the specified position from the CIsComVariantList class.

■ Format

CIsComVariant* GetElem(int index)

■ Parameters

Parameter	Description
int index	Position of CIsComVariant data to get

■ Return Value

Pointer to ClsComVariant data

Size

Gets the number of ClsComVariant data registered in the CIsComVariantList class.

■ **Format**
int Size()

■ **Parameters**
None

■ **Return Value**
Number of ClsComVariant data registered in the CIsComVariantList class

Clear

Deletes all the data registered in the CIsComVariantList class.

■ **Format**
bool Clear()

■ **Parameters**
None

■ **Return Value**
true: Successful
false: Failed

ReplaceElem

Overwrites CIsComVariant data in the specified position in the CIsComVariantList class. Returns pointer to the data that is overwritten.

■ Format

CIsComVariant* ReplaceElem(int index, CIsComVariant& elem)

■ Parameters

Parameter	Description
int index	Registration position of CIsComVariant data
CIsComVariant& elem	CIsComVariant data to register

■ Return Value

Pointer to position of registered CIsComVariant data that was overwritten.

■ Example

```
CIsComVariantList varList;
   [...]
//swap varList elements 3 and 1
CIsComVariant *vartmp = varList.ReplaceElem(1, (CIsComVariant&)(varList.GetElem(3)));
varList.ReplaceElem(3, (CIsComVariant&)(*vartmp));
```

2.3 ISAPP SETTING

2.3.1 Summary

This is an auxiliary tool for using IS-APP. Supports creation of IS-APP execution scripts.

2.3.2 Executable file name

Executable file
isapp_setting

Transfer to the EM series main unit is required.

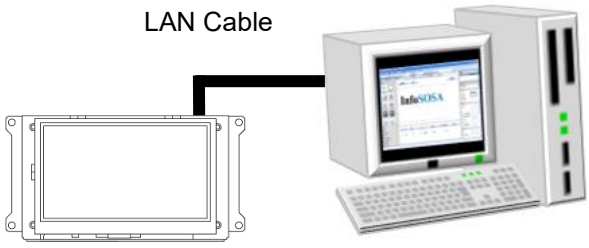
For the transfer method, please refer to "1.6 Update of IS-APP/IS-API/IS-APP SETTING".

2.3.3 Usage

Data Transfer over LAN Cable

Transfer from the PC to EM Series IS-APP executable files and screen data for IS-APP display.
Operate the host PC (Windows).
SambaServer is installed in EM series, and the user folder of EM series can be accessed from Windows.

Network settings are required for transfer.
Follow the procedure in "1.5 Connection between EM series and PC ".



While the EM Series unit is on, connect to the PC with a LAN cable.

Access the following address with Explorer etc.

EM series user folder path	\\192.168.0.130\em\user

*The IP address is the default value when shipped from the factory.

Log in as the following user.

user	root
password	(none)

Use Explorer or other tool to copy the following data to the EM Series unit which was recognized as a storage device.
For the data creation process, refer to the "InfoSOSA Builder Operation Manual" or "1.7　IS-APP HMI Development Process".

Folder/File	Description
data	Converted screen data for transfer * Copy the entire folder

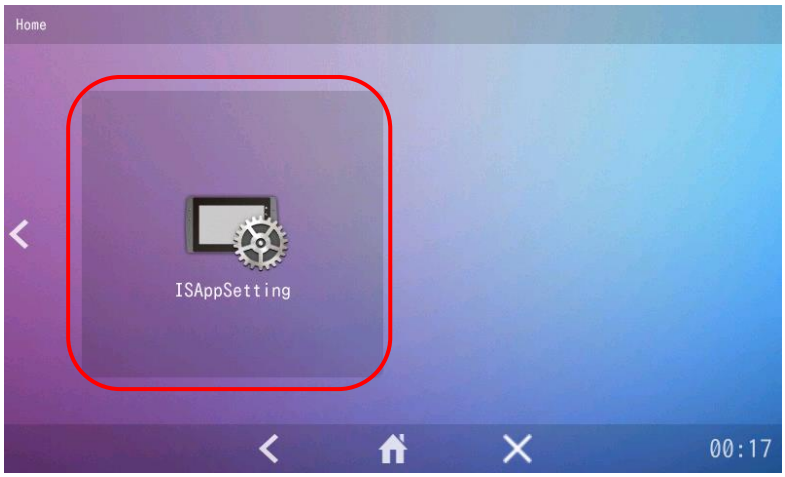
[If you cannot access the EM series user folder]

- Communication is not possible if the IP address for the LAN cable is "192.168.10.*". Please change it using the system setting tool.
- Please check whether there is another device with the same IP address "192.168.0.130" in your PC environment.

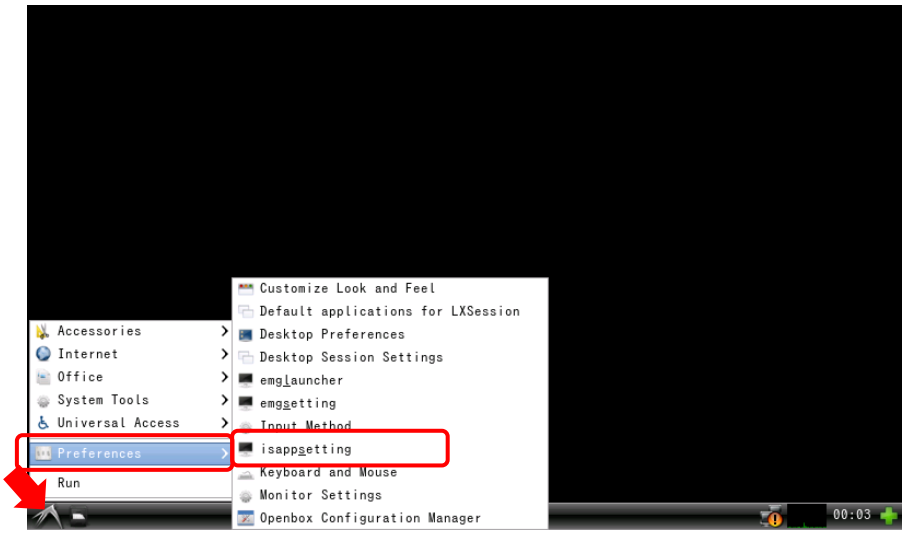
The following describes EM Series unit operation.

From the EMG Launcher or the Start menu click [ISAPP SETTING].

[EMG Launcher] - [ISAppSetting]

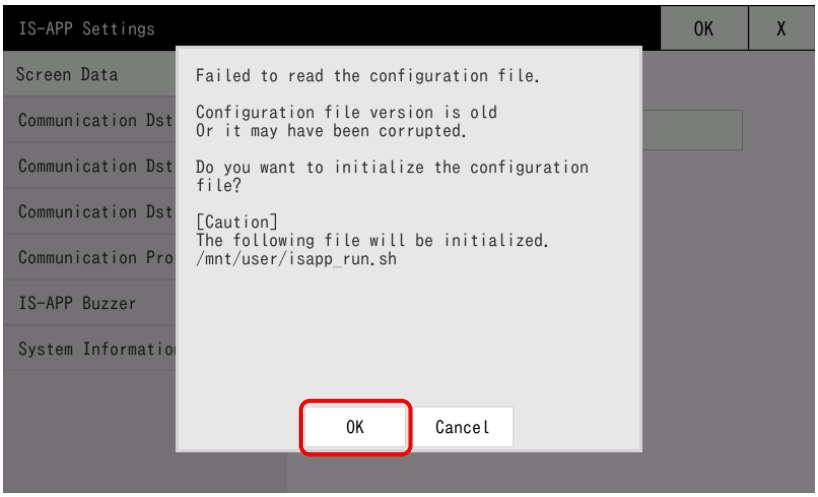


[Start menu] - [Preferences] - [isappsetting]



In the following cases, the dialog shown below will be displayed.

- The version of the configuration file is old.
- Configuration file is corrupted.



Please touch the OK button.

* The following file below will be initialized.
  /mnt/user/isapp_run.sh

Create IS-APP execution script

Specify IS-APP as a command-line argument when starting communication settings.
If you use ISAPP SETTING you can set up command-line arguments in a graphical user interface.
(Create startup script with command line arguments set up)

The startup script is created to run the IS-APP in the following location.

```
/usr/bin/is_app
```

List of command line arguments you can set up

Item	Description
Screen data folder	Specifies the screen data folder to display in IS-APP.
Communication Settings	Specify the IS-APP communication settings for the external device.
Communication Protocol	Specify the IS-APP communication protocol for the external device.
Buzzer	Set IS-APP to use the buzzer device.

List of fixed command line arguments

Item	Description	Setting
Sound	Sets the IS-APP sound function.	Device name: default Mixer name: default Volume name: PCM * Disabled when the model does not support sound.

* To change the above setting, refer to "2.1.6 Command line arguments" and edit the startup script.

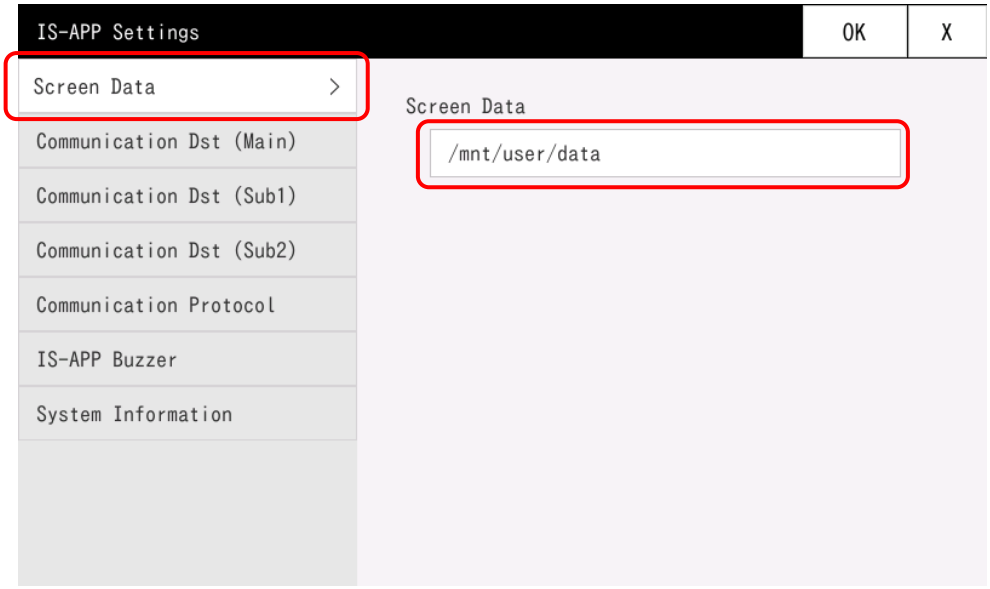
After completing the setting, touch the OK button in the upper right to update the start script file.

The start script is created in the following directory.

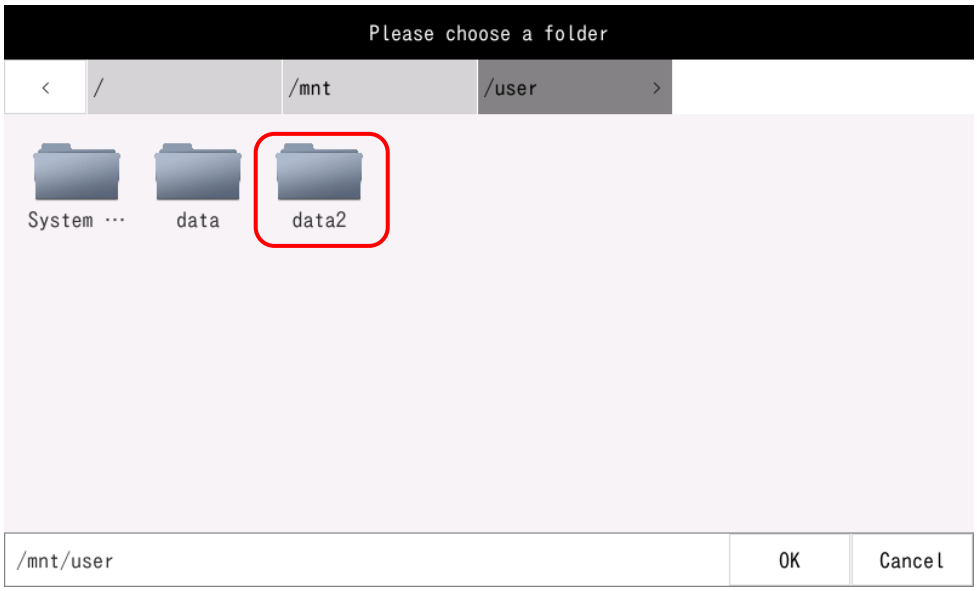
/mnt/user/isapp_run.sh

Screen data

To change the screen data folder, select “Screen data” from the menu and touch the area where the folder path is displayed.



Please select the screen data folder.



Please touch the OK button.

Communication Dst (Main) / Communication Dst (Sub1) / Communication Dst (Sub2)

You can set up to three IS-APP communication destinations.
The combinations that can be specified are as follows.

Communication Dst (Main)	Communication Dst (Sub1)	Communication Dst (Sub2)
SIO	None	None
LAN	None	None
IS-API	None	None
SIO	LAN	None
SIO	IS-API	None
LAN	SIO	None
IS-API	SIO	None
SIO	SIO	None
SIO	SIO	LAN
SIO	SIO	IS-API
LAN	SIO	SIO
IS-API	SIO	SIO

Even if “Sub1” and “Sub2” are reversed, processing will not be affected.

LAN and IS-API cannot be used at the same time.
When connecting multiple LANs (including IS-API), specify TCP/IP server as the LAN connection protocol.

If you specify the SIO, you cannot specify the same communication port (COM1/COM2).

The event notification destination at the time of executing the action "Notify event to Host" or "Notify value to Host" is the interface specified in the ”Communication Dst (Main). The “Communication Dst (Sub1) and “Communication Dst (Sub2)” are not notified.

The output destination when executing the action "Output string memory to Host" is output only when serial is specified as the Main argument. It is not output when LAN, IS-API is specified, or when the Sub argument is specified.

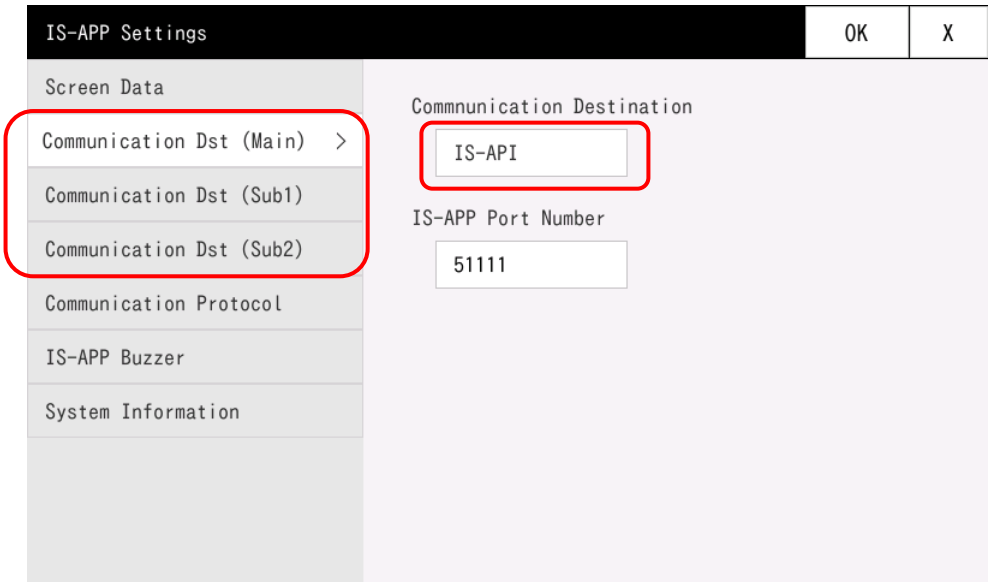
“Start Message” is sent to all interfaces.* Excluding TCP/IP server
[Start Message (Serial)]
{STX}00s0000000800010000003DC{ETX}
[Start Message (LAN)]
s000000800010000

* When connecting to IS-API, the character code setting (SI03) is automatically set to "Unicode (UTF-16LE)". When using IS-API, do not set it to "Shift JIS". IS-API will not work properly.

* The character code setting applies to all communication interfaces. When using IS-API, use "Unicode (UTF-16LE)" for serial and LAN communication.

Communication destination (IS-API)

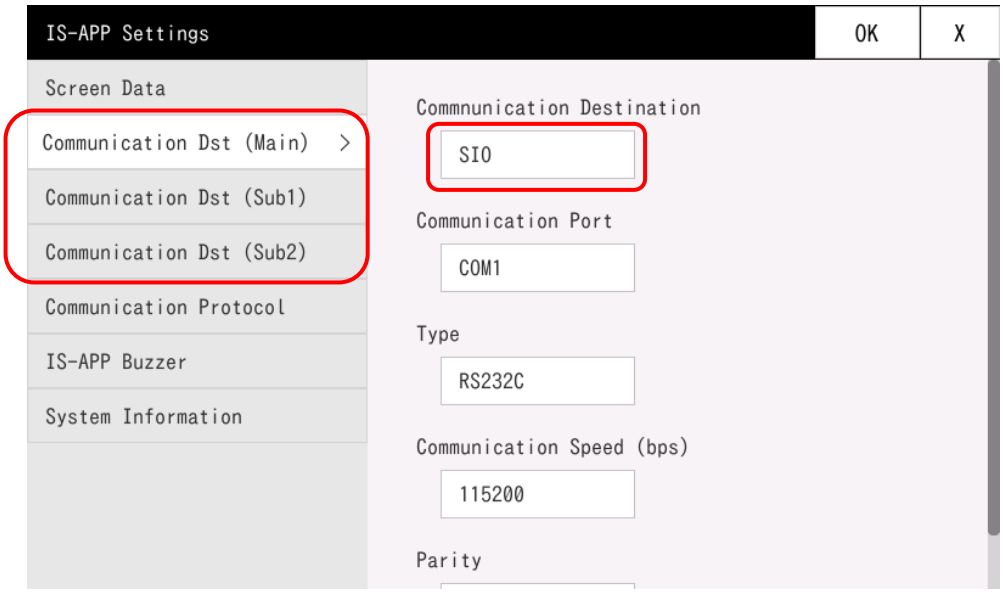
When setting the communication to internal communication (IS-API), select "IS-API" as the communication destination.



Item	Description
IS-API port number	Specify the standby port of the communication destination application.

Communication destination (SIO)

To set the communication to serial communication (SIO), select "SIO" as the communication destination.



Item	Description
Communication port *1	Select from COM1(SIO1), COM2(SIO2).
Type *1	Select from RS232C／RS422／RS485.
Communication speed (bps) *1	Set the communication speed.
Parity *1	Set parity.
Stop bit	1 is fixed.
Local address *2	Set the local address of the device.
Collision retransmission count *2	Set the number of retransmissions when communication collision occurs.
Collision retransmission interval (ms) *2	Set the retransmission interval when communication collision occurs.
Transmission authority timeout *2	Set the timeout of the waiting for free communication line.

* 1 Available settings depend on the product. Please check your product specifications.
* 2 Only displayed when RS485 is selected as the type.

Communication destination (LAN) UDP/IP

To set the communication to LAN communication (UDP/IP), select "LAN" as the communication destination and set the mode to "UDP/IP".

Item	Description
Address	Set the address of the communication destination device.
Port	Specify the standby port of the communication destination device.

Communication destination (LAN) TCP/IP Client

To set the communication to LAN communication (TCP/IP client), select "LAN" as the communication destination and set the mode to "TCP/IP Client".

Item	Description
Address	Set the address of the communication destination device.
Port	Specify the standby port of the communication destination device.
TCP connection establishment request interval(s)	Sets the interval of connection request during TCP / IP communication.(Seconds)

Communication destination (LAN) TCP/IP Server

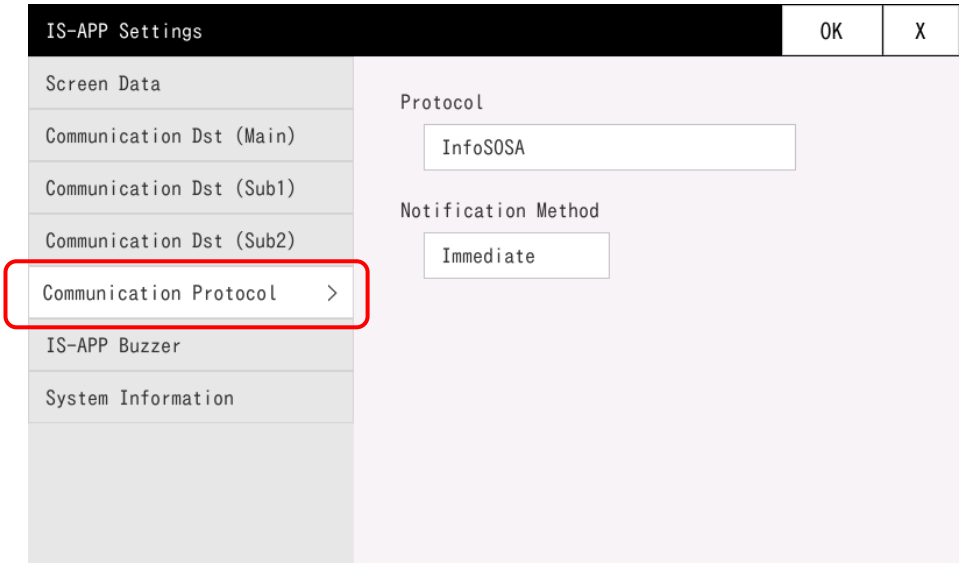
To set the communication to LAN communication (TCP/IP server), select "LAN" as the communication destination and set the mode to "TCP/IP Sever".

Item	Description
Port (with notification)	Specify the standby port number. Maximum number of connections: 1 Event notification is sent only to devices connected to this port. *It is necessary to set it as the communication destination (Main). When set to the communication destination (Sub), event notification will not be performed even for ports with event notification
Port (without notification)	Specify the standby port number. Maximum number of connections: 3 *Even if it is set as the communication destination (Main), event notification will not be sent to the device connected to this port.

Communication Protocol

To set the communication protocol, select “Communication Protocol” from the menu.

【Note】
-The setting of this argument is applied to all communication destinations.
-If the communication destination includes IS-API*, it is necessary to set "InfoSOSA protocol/immediate notification".
*Including the case of connecting to IS-API with TCP/IP server



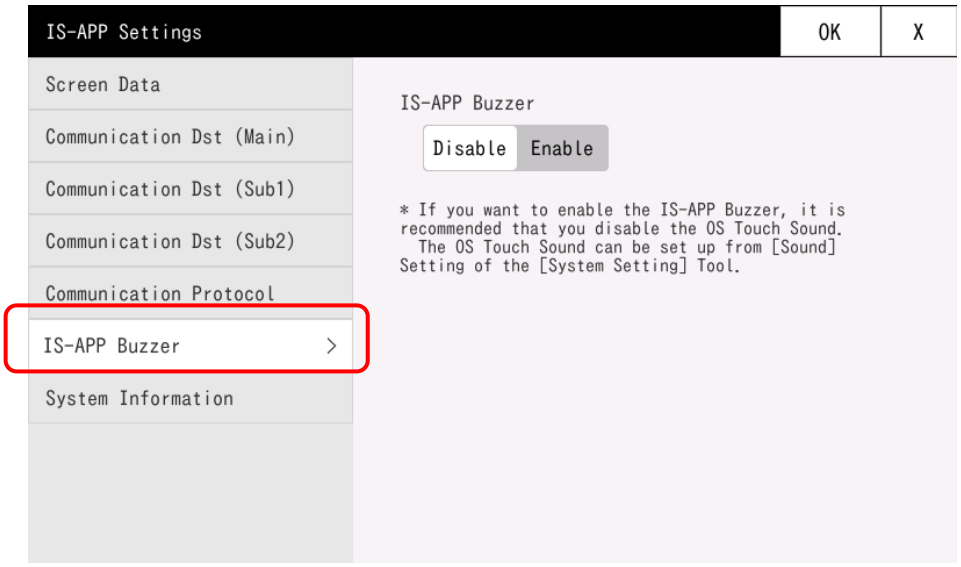
Item	Description
Protocol	Select from InfoSOSA/Standard.
Notification Method	Select from Immediate/Polling.
Monitoring Time (ms) *1	Specify the time until retransmission of standard protocol.
Retry count *1	Specify the retransmission number of standard protocol.

*1 This is displayed only when selected the [Standard] for [Protocol].

For the Protocol and Notification Method, please refer to the separate "InfoSOSA Reference Manual".

Buzzer

To set the buzzer, select “IS-APP Buzzer” from the menu.

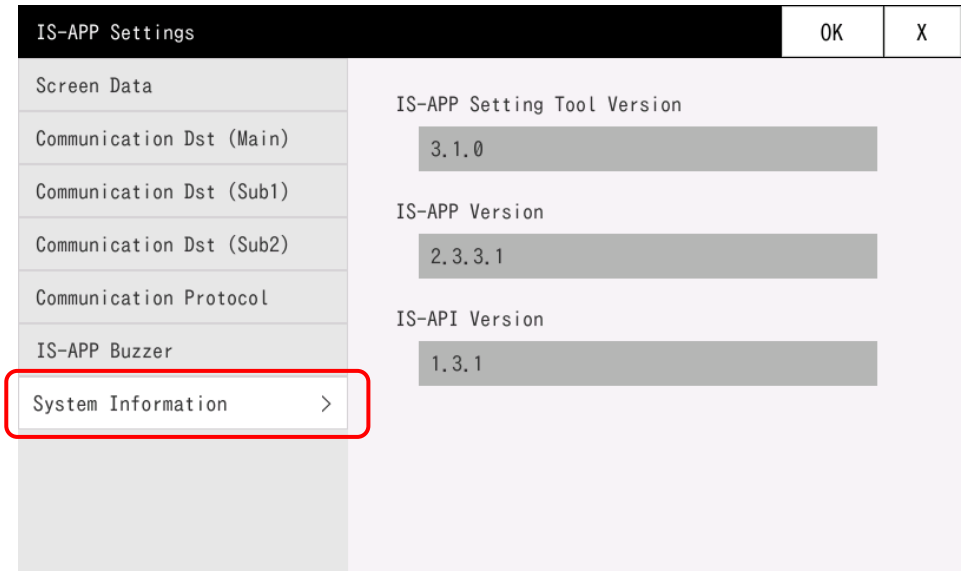


Item	Description
IS-APP Buzzer	Select from Disable/Enable.

*If you want to enable the IS-APP Buzzer, it is recommened that you disable the OS Touch Sound.
*The OS Touch Sound can be set up from [Sound] Setting of the [System Setting] Tool.

System Information

Display the version.



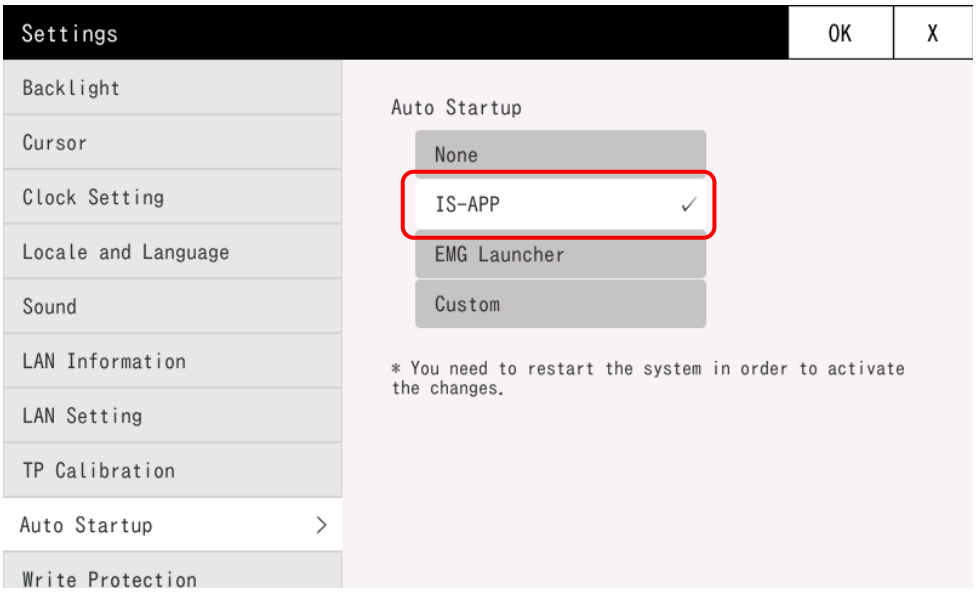
Item	Description
IS-APP Setting Tool Version	The version of this tool is displayed.
IS-APP Version	The version of /usr/bin/is_app is displayed.
IS-API Version	The version of /usr/lib/libisapi.so is displayed.

IS-APP automatic startup settings

You can set up IS-APP to start up automatically when the power is turned on.

To start up automatically, set up so the following script is run at startup.

```
/mnt/user/isapp_run.sh
```

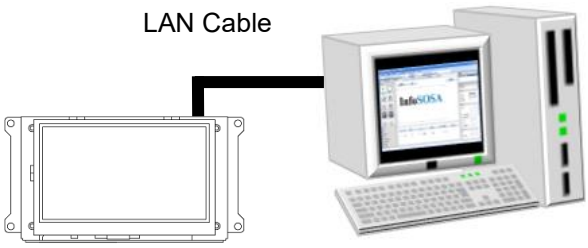


For more information, refer to the separate document “EM Series Tool Manual”.

Update screen data after automatic startup settings

To update IS-APP screen data, follow the steps below.
To change the communication settings, refer to the next item (Change communication settings).

Network settings are required in advance.
Follow the procedure in "1.5 Connection between EM series and PC ".



While the EM Series unit is on, connect to the PC with a LAN cable.

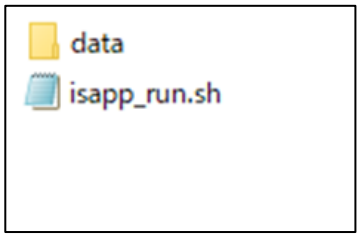
Access the following address with Explorer etc.

EM series user folder path	\\192.168.0.130\em\user

*The IP address is the default value when shipped from the factory.

Log in as the following user.

user	root
password	(none)



Delete screen data folder (default name **data**) saved on the EM Series unit.

Use Explorer or other tool to copy the updated screen data to the EM Series unit which was recognized as a storage device.

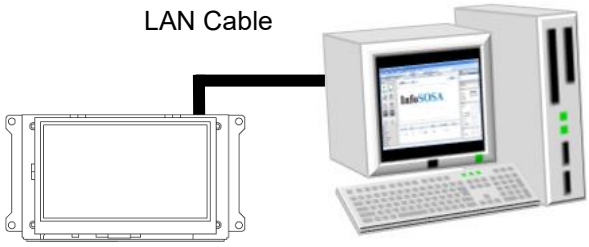
CAUTION
To update the screen data, delete the entire [data] folder first, and then transfer the updated screen data. (If previous files are in the folder, it could cause an operation failure.)

Turn on the EM Series unit and the updated screen is displayed.

Change communication settings after automatic startup settings

To change IS-APP communication settings, automatic startup must be temporarily disabled.

Network settings are required in advance.
Follow the procedure in "1.5 Connection between EM series and PC ".



While the EM Series unit is on, connect to the PC with a LAN cable.

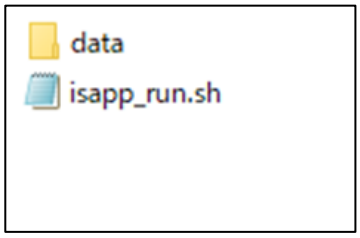
Access the following address with Explorer etc.

EM series user folder path	\\192.168.0.130\em\user

*The IP address is the default value when shipped from the factory.

Log in as the following user.

user	root
password	(none)



Delete the IS-APP execution script (**isapp_run.sh**) saved on the EM Series unit.

Power cycle the EM series unit.
With IS-APP automatic startup temporarily disabled, start the [ISAPP SETTING] tool and set up.

[If you cannot access the EM series user folder]

- Communication is not possible if the IP address for the LAN cable is "192.168.10.*". Please change it using the system setting tool.
- Please check whether there is another device with the same IP address "192.168.0.130" in your PC environment.

3. Others

Chapter Contents

3.1 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/

12th Edition December 2024
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/