InfoSOSA™ シリーズ

# IS-APP スタートアップガイド

株式会社ディ・エム・シー
https://www.dush.co.jp/

# はじめに

この度は、ディ・エム・シーの製品をお買い上げ頂き、誠にありがとうございます。
本マニュアルは、InfoSOSA シリーズ IS-APP の特徴のご紹介、チュートリアル、InfoSOSA シリーズ IS-APP 固有の機能について記載しています。
以降、InfoSOSA シリーズ IS-APP は、InfoSOSA または IS-APP と表記します。

**対象ユーザ**

- ✓ IS-APP をご検討の方
- ✓ 初めて IS-APP をご使用される方
- ✓ IS-APP 固有の機能／仕様について確認されたい方

**対象バージョン**

本マニュアルは以下のバージョンの InfoSOSA について記載しております。

バージョンにより一部動作が異なる場合がございます。

詳細は、別紙「InfoSOSA リリースノート」を参照ください。

| | |
|---|---|
| InfoSOSA Builder | 2.7.1 |
| IS-APP | 2.4.1 |
| IS-API | 1.3.2 |

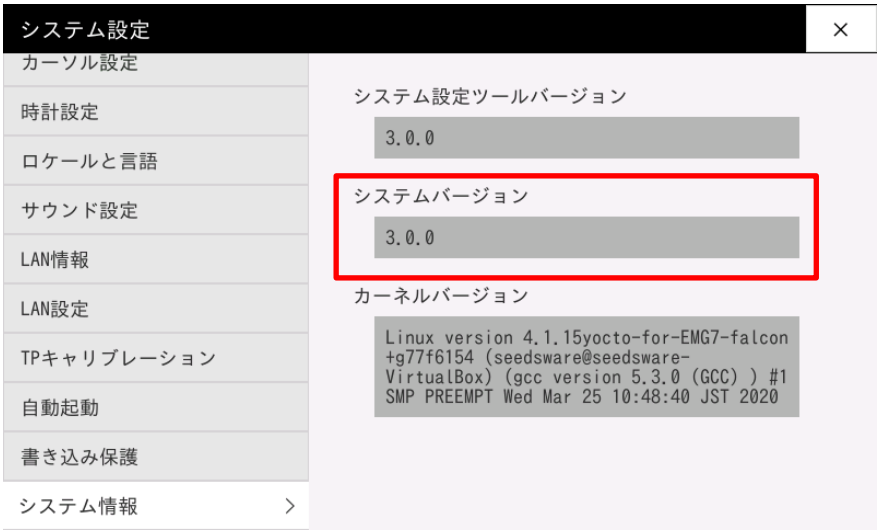
**EM シリーズ本体のシステムバージョン**

本マニュアルは、EM シリーズ本体が下記のシステムバージョンの場合を想定して記載しております。

EM シリーズ本体のシステムバージョンは、システム設定ツールから確認ください。

システム設定ツールについては、別紙「EM シリーズ ツールマニュアル」を参照ください。

| | |
|---|---|
| 対象システムバージョン | 3.0.0 ～ |



# 目 次

# 1章　IS-APP について

**章目次**

# 1.1　IS-APP概要

IS-APP は、アプリケーション版 InfoSOSA です。
作画ソフト「InfoSOSA ビルダ」で作成した画面をディ・エム・シー製パネルコンピュータ「EM シリーズ」上で動作させることができるアプリケーションです。
IS-APP を使用することで、パネルコンピュータに表示する HMI を簡単に作成することが可能になります。

**MEMO**

◆InfoSOSA とは
ディ・エム・シー製マイコン用タッチパネル表示機です。作画ソフト「InfoSOSA ビルダ」が付属し、簡単に HMI を作成することが可能な製品です。

また、IS-APP へのアクセス用 API、「IS-API」が付属しています。
「IS-API」を使用すると以下のようなことが可能です。
・お客様で作成された C/C++アプリケーションなどから IS-APP の画面を変更する
・IS-APP 内のボタンが押されたときに C/C++アプリケーション内の関数を実行したりする

これにより、上位機器との通信や制御部分は、C/C++アプリケーションで行い、
HMI 部分は「InfoSOSA ビルダ」で作成した画面を IS-APP で表示するような使い方も可能です。

# 1.2　関連ドキュメント

本書に関連するドキュメントは以下になります。目的に合わせて参照ください。

## InfoSOSA 開発キット

以下は InfoSOSA 開発キットに同梱されております。

### IS-APP スタートアップガイド（本書）

このドキュメントです。
IS-APP の特徴のご紹介、チュートリアル、IS-APP 固有の機能／仕様について記載しています。

**対象ユーザ**

- ✓ IS-APP をご検討の方
- ✓ 初めて IS-APP をご使用される方
- ✓ IS-APP 固有の機能／仕様について確認されたい方

### InfoSOSA リファレンスマニュアル

InfoSOSA の機能／仕様について記載しています。

**対象ユーザ**

- ✓ InfoSOSA の機能や仕様の詳細を調べたい方
- ✓ InfoSOSA と上位機器の通信仕様を調べたい方

### InfoSOSA ビルダ操作マニュアル

InfoSOSA ビルダの操作方法について記載しています。

**対象ユーザ**

- ✓ InfoSOSA ビルダをご使用中に設定／操作の詳細を調べたい方
- ✓ InfoSOSA ビルダの便利な使い方を知りたい方

## 上位通信テスタ取扱説明書

上位通信テスタの操作方法について記載しています。

※ 上位通信テスタは、上位機器の代わりに PC で InfoSOSA と通信させるためのソフトウェアです。

**対象ユーザ**

- ✓ 上位機器を使わずに、InfoSOSA と通信テストを行う方
- ✓ 上位機器のデバッグを行う時に、通信コマンドの確認を行いたい方
- ✓ 上位通信テスタをご使用中に設定／操作の詳細を調べたい方

## InfoSOSA Ver2.1 からの移行ガイド

InfoSOSA Version2.5 の新機能と Version2.0/Version2.1 プロジェクトを Version2.5 以降で使用する方法について記載しています。

**対象ユーザ**

- ✓ InfoSOSA Version2.0/Version2.1 をご使用で Version2.5 以降への移行をご検討の方

## InfoSOSA リリースノート

InfoSOSA のバージョンによる違いを記載しています。

**対象ユーザ**

- ✓ InfoSOSA をご使用でバージョンアップをご検討の方

## EM シリーズ開発環境

以下は、InfoSOSA 開発キットには含まれておりません。
EM シリーズ開発環境（DVD）に同梱されております。

弊社ホームページからもダウンロード可能です。

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

### EM シリーズ ソフトウェア開発マニュアル

EM シリーズで動作するソフトウェアの開発方法について記載しています。

**対象ユーザ**

- ✓ EM シリーズでソフトウェアを開発される方

### EM シリーズ ツールマニュアル

EM シリーズに搭載されているツールについて記載しています。

**対象ユーザ**

- ✓ EM シリーズでソフトウェアを開発される方
- ✓ EM シリーズをご使用される方

### EM シリーズ Smart e-Studio 取扱説明書

EM シリーズを使った製品の開発や生産を補助するリモートメンテナンスツールです。

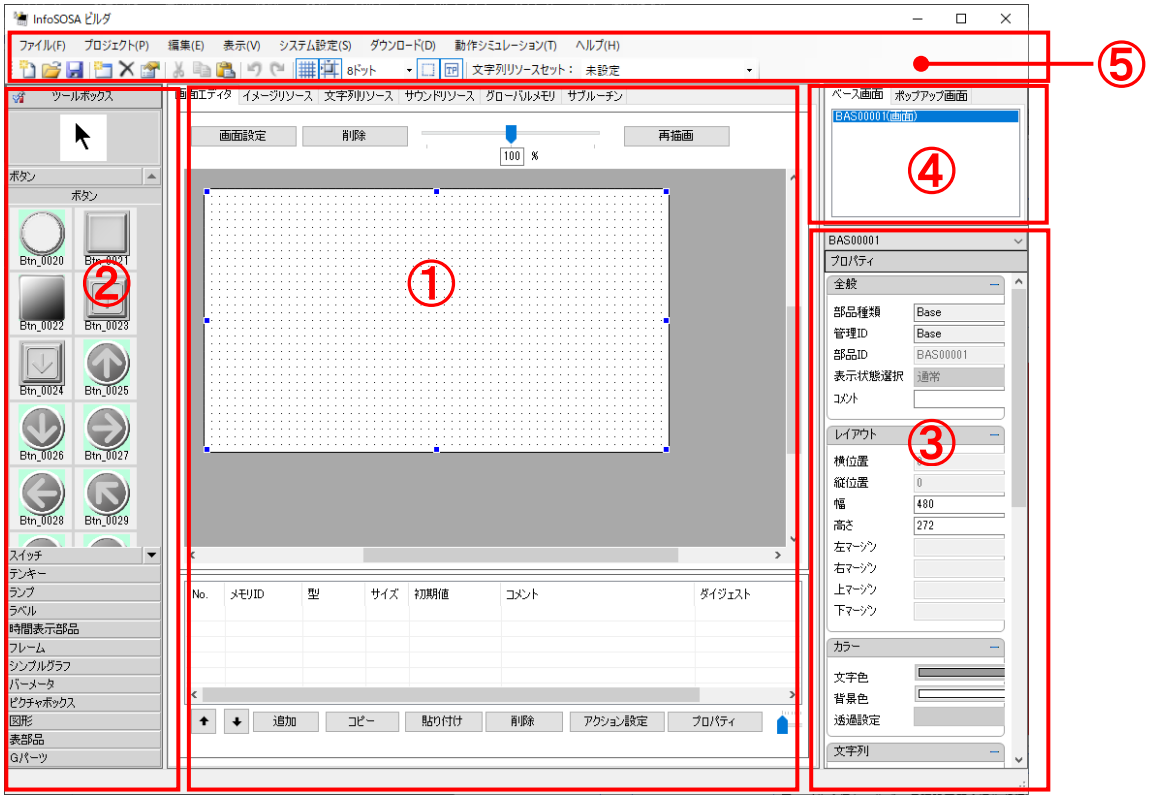
PC 上で動作します。

**対象ユーザ**

- ✓ EM シリーズでソフトウェアを開発される方
- ✓ EM シリーズにデータをインストールされる方

# 1.3　InfoSOSAビルダとは

InfoSOSA ビルダとは、InfoSOSA で表示する画面を作成するためのソフトウェアです。
部品の配置や動作設定は、ドラッグ＆ドロップ、プルダウンメニューからの選択など、マウスのみを使用した簡単な操作で行います。複雑な操作や、設定のためにプログラムソースを記述する必要はありません。
InfoSOSA ビルダは以下の画面から構成されています。




**①作画エリア**　主に作画を行うエリアで、複数のタブから構成されています。
今回のデモでは「画面エディタ」タブを使用します。
他のタブの詳細や操作方法については、リファレンスマニュアル又はビルダ操作マニュアルを参照してください。

**②ツールボックスエリア**　作画に使用できる部品が種類ごとに格納されています。
部品は作画エリアにドラッグ＆ドロップするだけで配置できます。

**③プロパティエリア**　「画面エディタ」内で選択した部品のプロパティが表示されます。
部品の大きさや色、レイアウト等の多彩な設定が可能です。

**④画面リストエリア**　作成中の画面の名前がリストで表示されます。
画面の名前をクリックすると、その画面を編集できます。

**⑤メニュー・ツールバーエリア**　設定画面の呼び出しや、ファイルの保存・作成といった操作を行います。

# 1.4　InfoSOSAビルダのインストール

InfoSOSA ビルダのインストールについて解説します。

## 1.4.1　動作環境

InfoSOSA ビルダは以下の環境を満たす PC で動作します。
インストール前にお使いの PC が以下の動作環境を満たしているかご確認ください。

| 項目 | 内容 |
|---|---|
| 対応 OS | Microsoft® Windows® 10 日本語版(64bit 版)<br>または<br>Microsoft® Windows® 11 日本語版(64bit 版) |
| 必要なフレームワーク<br>ランタイム | Microsoft® .NET Framework 3.5<br>Microsoft® .NET Framework 4.7<br>Microsoft Visual C++ 2015 ランタイム |
| プロセッサ(相当) | 1GHz 以上 |
| メモリ | 4GB 以上（推奨：8GB 以上） |
| ハードディスク | 850MB 以上の空き（推奨：1GB 以上） |
| ディスプレイ | 1024×768 ドット以上　True Color (32bit)を推奨 |

※ 仮想環境下での動作は、動作保証外とさせて頂きます。

## 1.4.2　インストールの開始

注意
インストールを行うPCのハードディスクに十分な空き容量(850MB以上)があることを確認してインストールを行ってください。

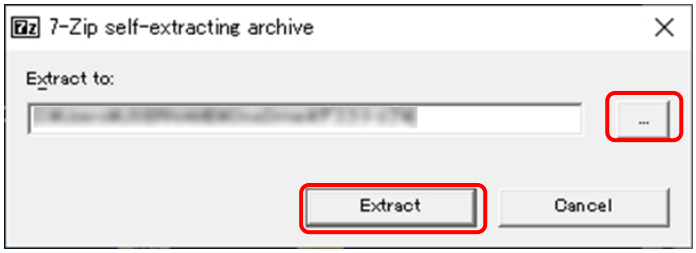
注意
インストールは Administrator（管理者）権限で行ってください。

注意
インストールは、全てのユーザに対して行なわれます。

注意
バージョンアップの場合は、インストール済みのビルダを事前にアンインストールしてください。

1. InfoSOSA 開発キットデータ内の「software」-「builder」フォルダの「InfoSOSABuilder*.*.*.*.exe」を実行してください。

2. 展開先を指定するダイアログが表示されます。[...]ボタンをクリックし、展開先(デスクトップ等)を指定して、[Extract]ボタンをクリックして下さい。



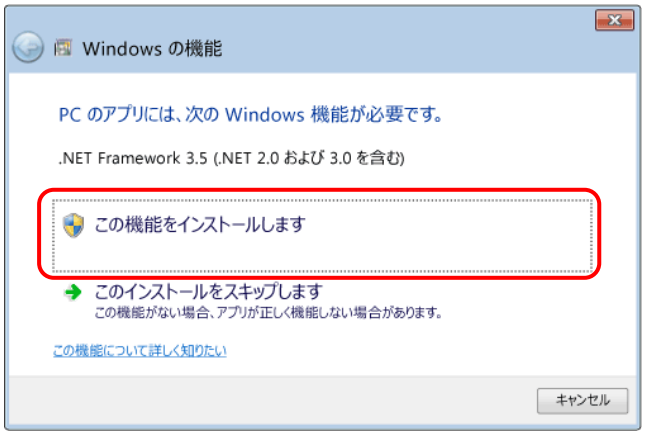

3. 上記 2.で指定した展開先に「InfoSOSABuilder*.*.*.*」フォルダが生成されます。

4. 生成されたフォルダ「InfoSOSABuilder*.*.*.*」内の「Setup.exe」を実行して下さい。

## 1.4.3　.NET Framework のインストール

InfoSOSA ビルダの実行には Microsoft® .NET Framework 3.5/.NET Framework 4.7 が必要です。
お使いの PC にインストールされていない場合は、ビルダのインストールの開始前に「.NET Framework」のインストールを行ってください。
InfoSOSA ビルダのインストール画面が表示された場合は、そのまま InfoSOSA ビルダのインストールを行ってください。

「.NET Framework 4.7」は標準でインストールされています。「.NET Framework 3.5」がインストールされていない場合は、ビルダのインストーラ起動時に以下のようなダイアログが表示されます。「この機能をインストールします」を選択して、「.NET Framework3.5」のインストールを行ってください。

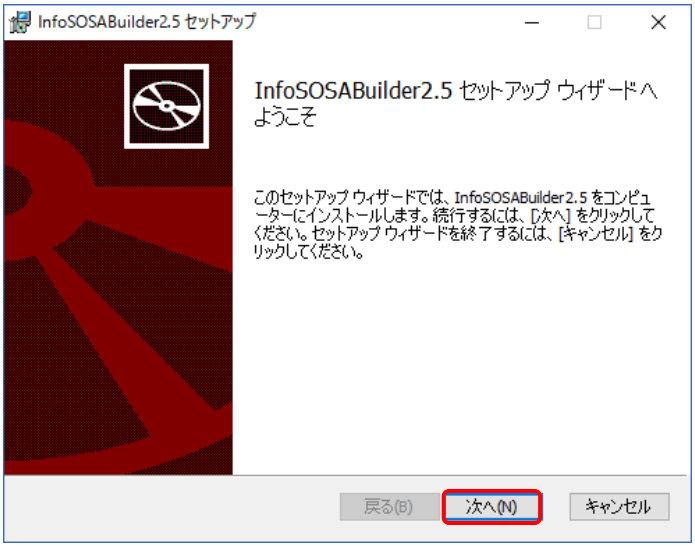



※ 「.NET Framework」のインストールは、Windows Update を使用して行なわれるため、インターネット接続が必要です。

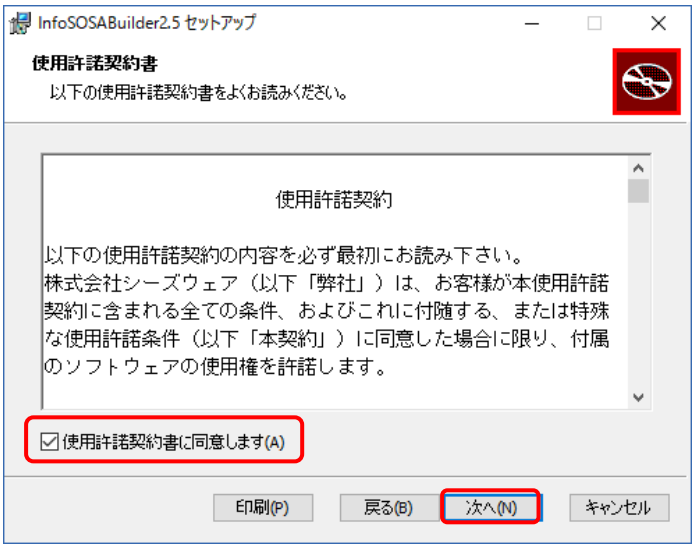
※ オフラインでインストールする場合は、Windows のインストールメディアをご使用頂く必要がございます。

## 1.4.4　InfoSOSA ビルダ のインストール

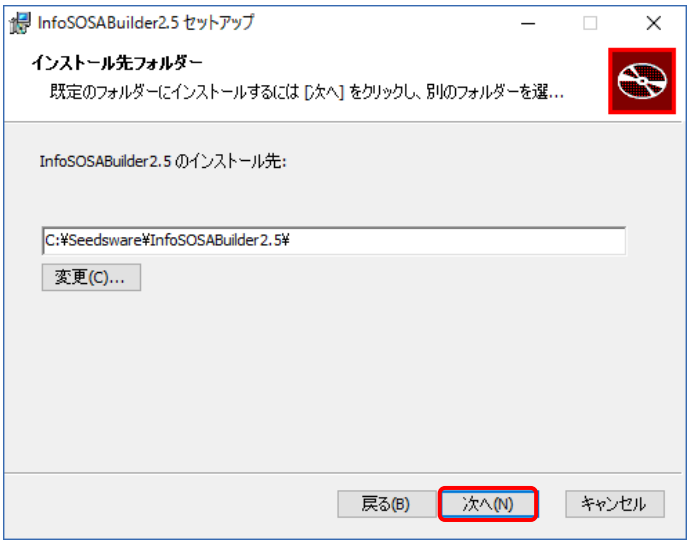
1. インストールを開始すると以下のようなダイアログが表示されます。
   ［次へ］ボタンをクリックします。



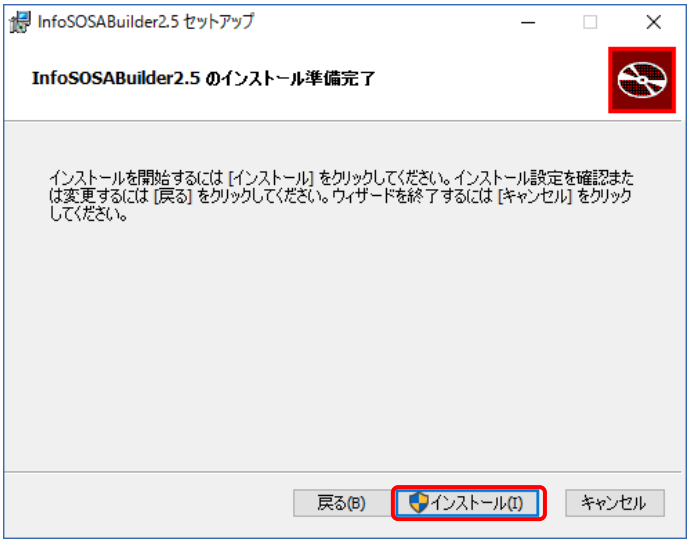

2. 使用許諾契約書をご確認頂き、［同意する］にチェックを入れて、［次へ］ボタンをクリックします。



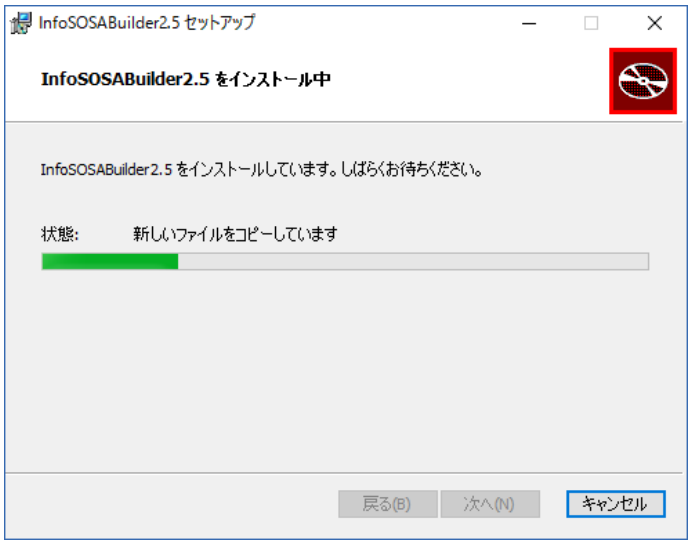
3. 保存先の指定を行い［次へ］ボタンをクリックします。




4. ［インストール］ボタンをクリックします。




5. インストールが開始されます。



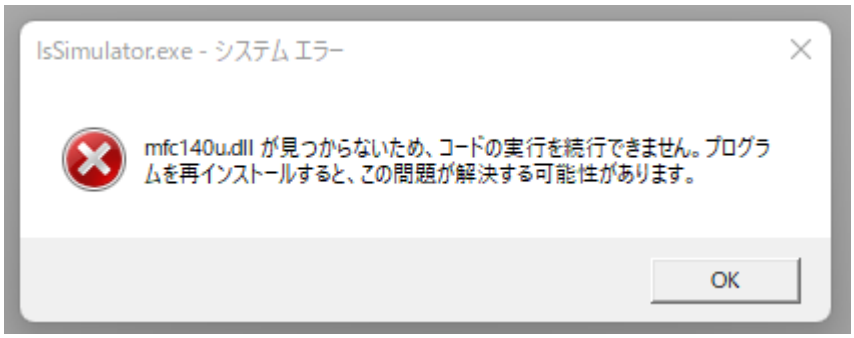
6. インストール完了です。


インストール後、 

ショートカットがデスクトップに登録されます。

## 1.4.5　Microsoft Visual C++ 2015 ランタイムのインストール

以下のようなエラーが表示される場合は、以下の手順で「Microsoft Visual C++ 2015 再頒布可能パッケージ」をインストールしてください。




1. Microsoft 社のダウンロードページにアクセスします。

| |
|---|
| https://learn.microsoft.com/ja-jp/cpp/windows/latest-supported-vc-redist?view=msvc-170 |

2. Microsoft Visual C++ 2015 ランタイムの「X86」「X64」を両方ダウンロードします。

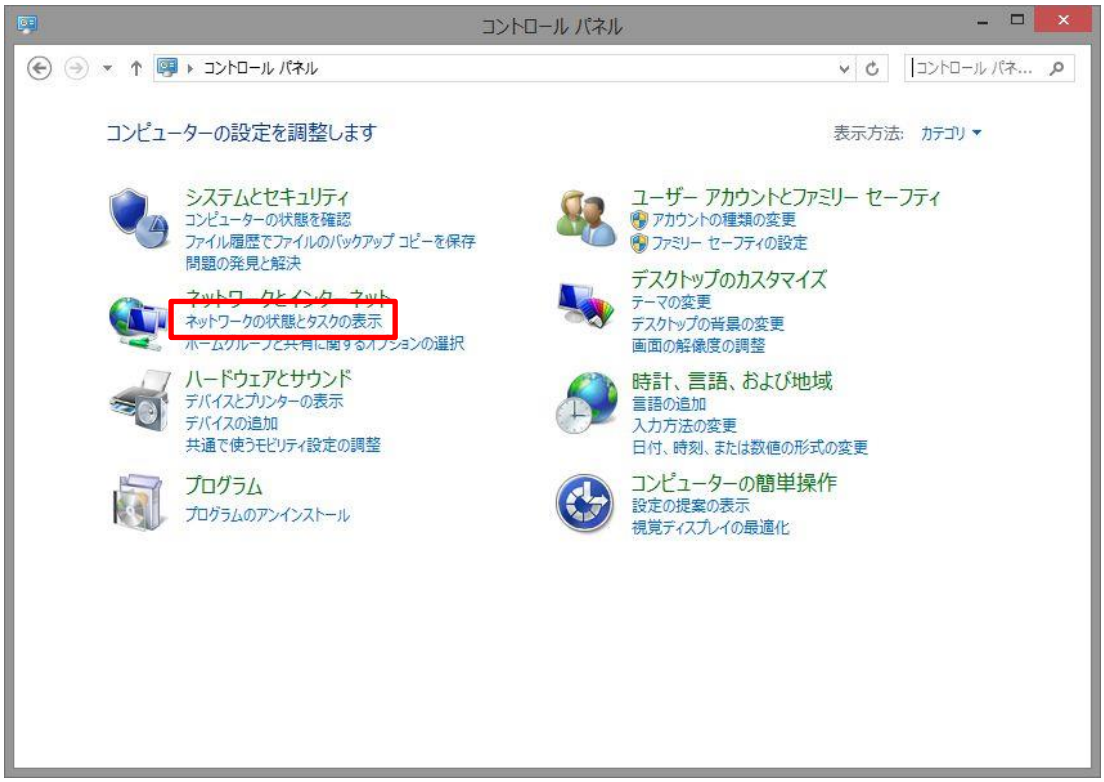
| | |
|---|---|
| X86 | https://aka.ms/vs/17/release/vc_redist.x86.exe |
| X64 | https://aka.ms/vs/17/release/vc_redist.x64.exe |

3. ダウンロードした「VC_redist.x64.exe」と「VC_redist.x86.exe」をそれぞれ実行してインストールしてください。

# 1.5　EMシリーズ本体とPCとの接続

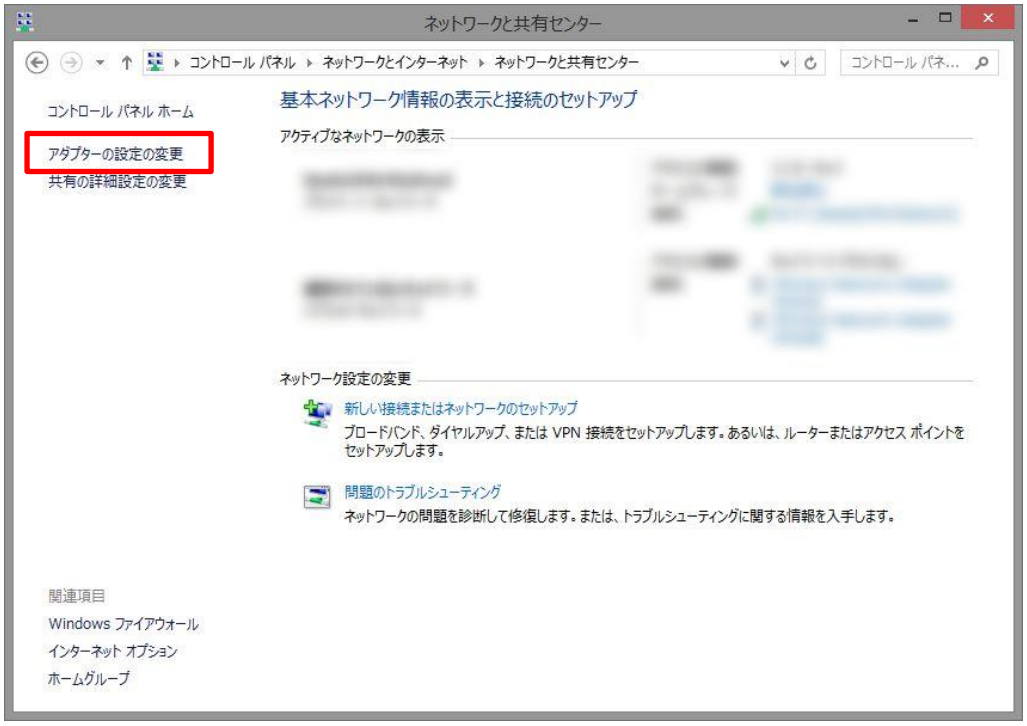
プロジェクト（画面データ）の転送を行うには、EM シリーズ本体を PC と接続する為の設定が必要です。
PC を操作して、EM と接続できるようにネットワーク設定を変更します。
ここでは LAN ケーブルを使用して接続しています。
USB ケーブルでの接続方法は、別紙「EM シリーズ　ソフトウェア開発マニュアル」を参照ください。

1．PC のコントロールパネルを開いて、「ネットワークの状態とタスクの表示」をクリックしてください。



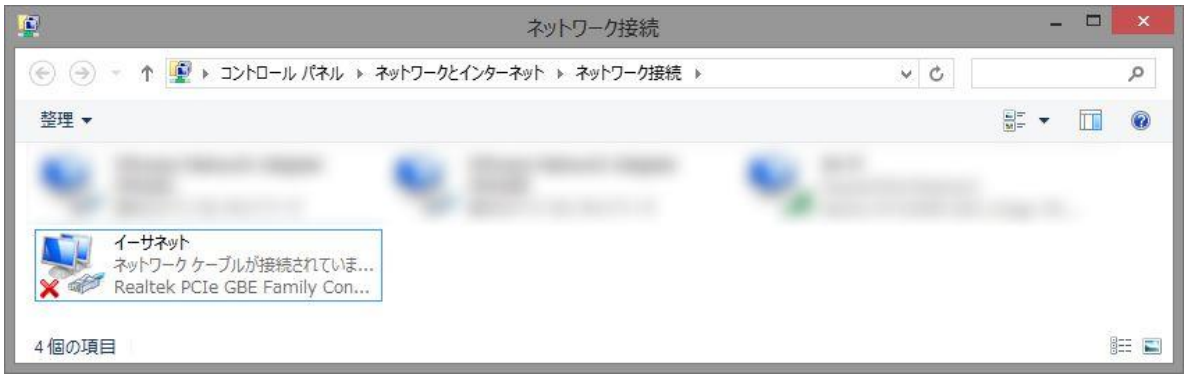

※表示方法が異なる場合は、右上の「表示方法」を「カテゴリ」に変更してください。

2.「アダプター設定の変更」をクリックしてください。

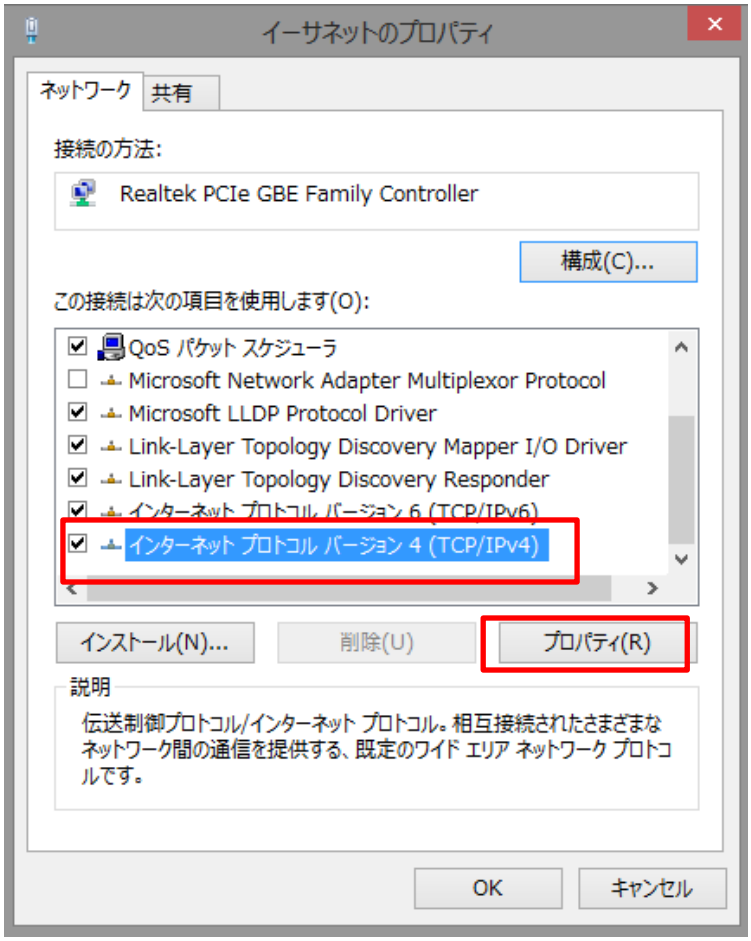



3.「イーサネット」を右クリックして、「プロパティ」をクリックして下さい。

※　環境によっては、「ローカルエリア接続」など名称が異なる場合があります。有線 LAN ポートのアダプターを選択して下さい。



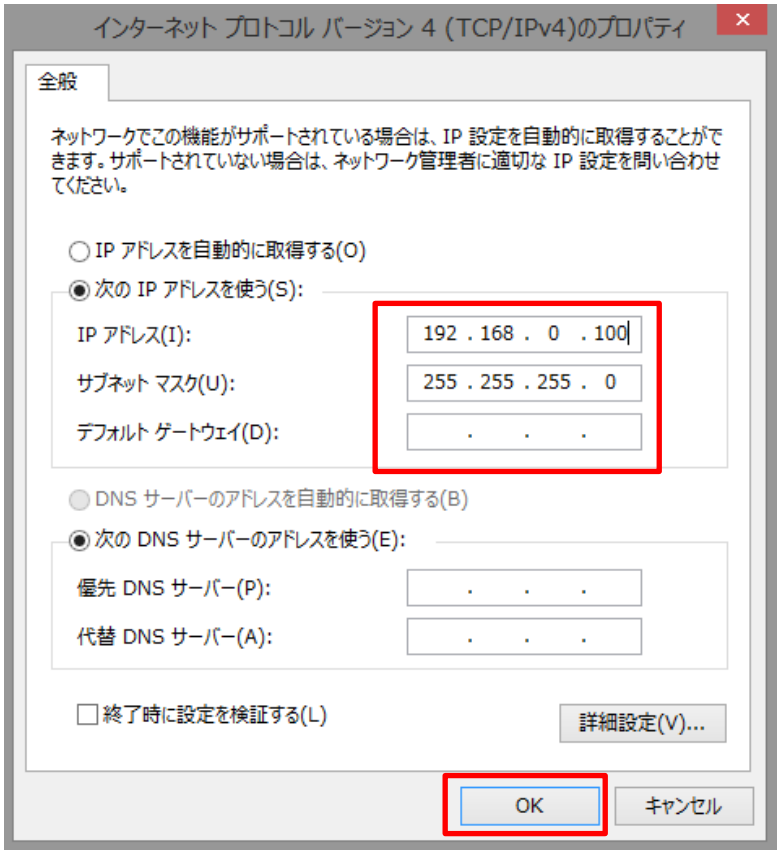
**4.**「インターネットプロトコルバージョン 4(TCP/IPv4)」を選択して、「プロパティ」をクリックしてください。



5. IP アドレス、サブネットマスク、デフォルトゲートウェイを以下の値に設定して OK ボタンをクリックしてください。

| 項目 | 値 |
|---|---|
| IP アドレス | 192.168.0.100 |
| サブネットマスク | 255.255.255.0 |
| デフォルトゲートウェイ | 未設定 |

※　EM 本体の IP アドレスを変更している場合は、PC が同じネットワークになるように設定してください。
※　EM 本体と IP アドレスが重複しない値にしてください。
※　「192.168.10.*」は USB-Ether(usb0)で使用している為、ご使用になれません。



# 1.6　IS-APP/IS-API/IS-APP SETTINGの更新

工場出荷状態で EM シリーズにインストールされているソフトウェアは最新ではありません。
EM シリーズに InfoSOSA 開発キットに同梱されている InfoSOSA アプリケーション「IS-APP」、IS-APP 用通信ライブラリ「IS-API」、IS-APP 設定補助ツール「IS-APP SETTING（IS-APP 設定ツール）」をインストールしてください。

インストールには Smart e-Studio を使用します。
「Smart e-Studio」は、Windows® PC 上で動作する EM シリーズ用リモートメンテナンスツールです。
詳しくは別紙「Smart e-Studio 取扱説明書」を参照ください。
Smart e-Studio は弊社ホームページからダウンロード可能です。

https://www.dush.co.jp/download/driver-app/

## 1.6.1　更新パッケージを使用

Smart e-Studio 用の更新パッケージを使用することで IS-APP、IS-API、IS-APP SETTING を一括で更新することが可能です。

1. EM シリーズ本体と PC を LAN ケーブルで接続し、SSH プロトコルで接続できるようにネットワークの設定を行ってください。

   ネットワークの設定は、ネットワーク管理者にご確認お願い致します。

| 項目 | 内容 |
|---|---|
| プロトコル | SSH プロトコル（TCP、ポート番号 22） |
| 実行アプリケーション | C:¥Seedsware¥Smart e-Studio¥Smart e-Studio.exe<br>※デフォルトインストール先 |

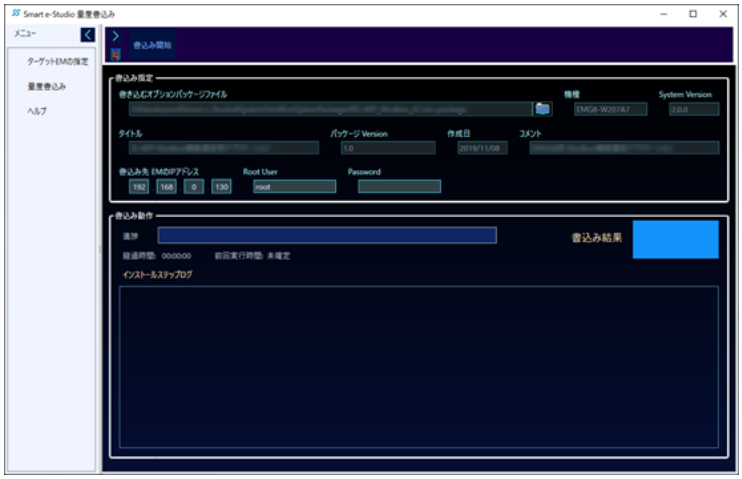
2. PC に「Smart e-Studio」をインストールしてください。

3. 更新パッケージファイルをダブルクリックしてください。

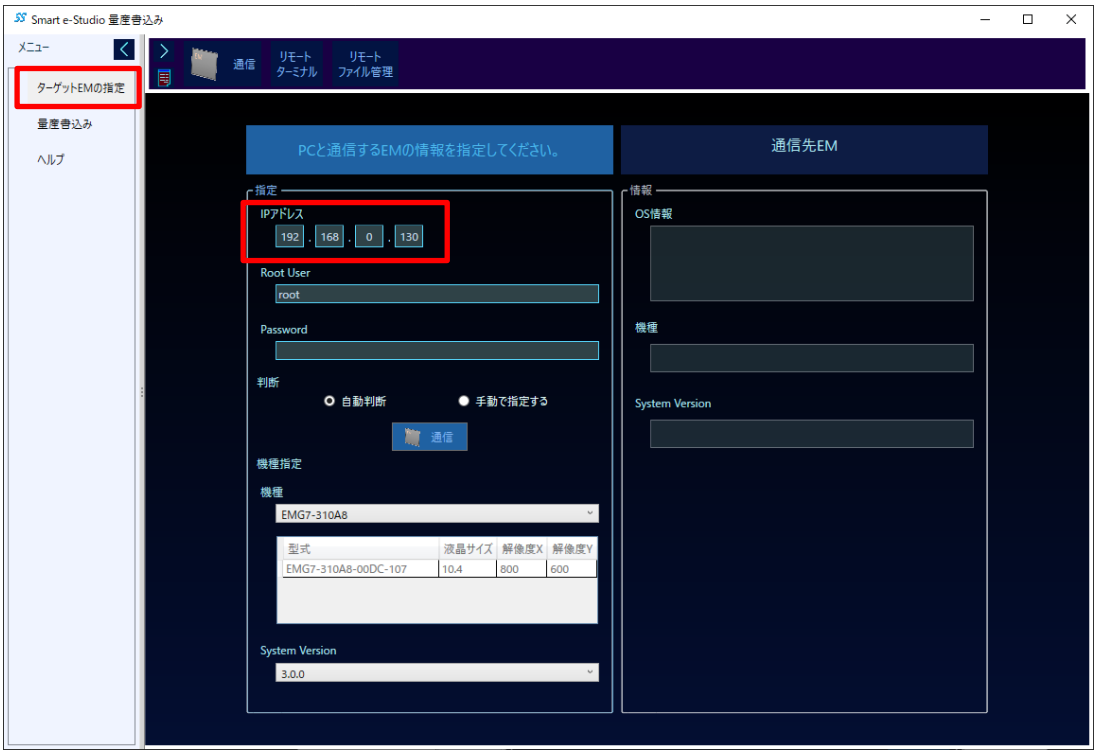
機種によって使用するファイルが異なります。

| 型式 | 更新パッケージファイル |
|---|---|
| EMG7-***A8-****-**7 | [InfoSOSA 開発キットデータ]¥software¥ Smart e-Studio 用更新パッケージ<br>¥IS-APP_update_A8_*****.em-package |
| EM8-***A7-****-**7 | [InfoSOSA 開発キットデータ]¥software¥ Smart e-Studio 用更新パッケージ<br>¥IS-APP_update_A7_*****.em-package |
| EMG8-***A7-****-**7 | |
| EMP-***A7-****-**7 | |

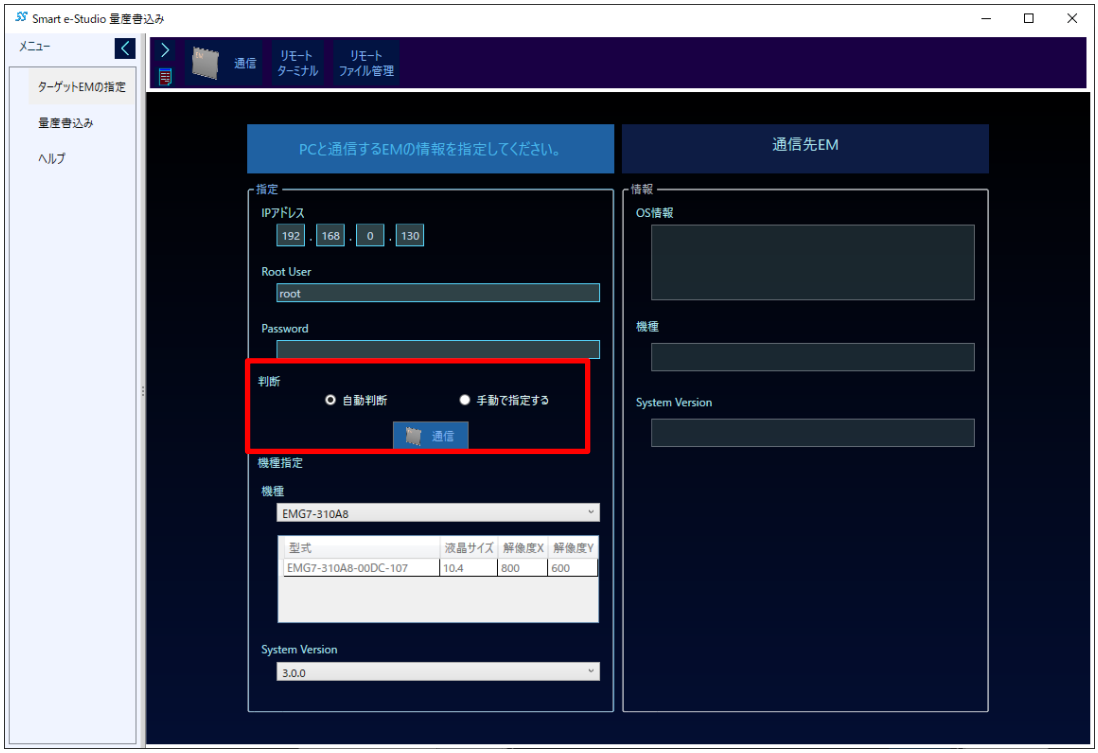
Smart e-Studio が起動します。



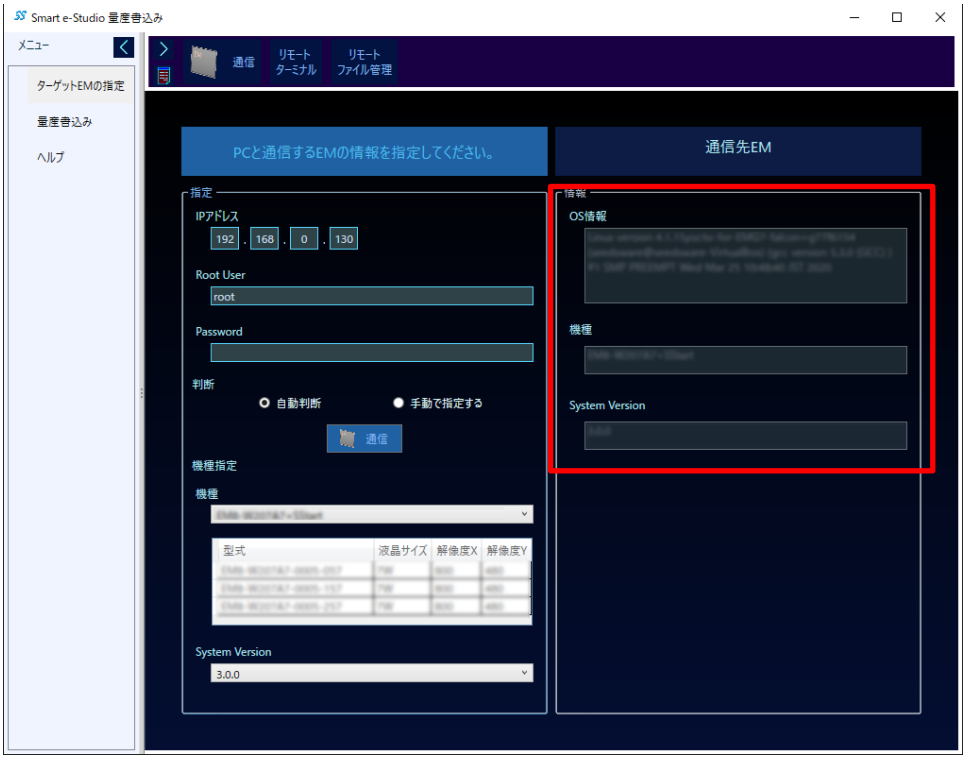
4. 「ターゲット EM の指定」ページを開き、EM シリーズ本体の IP アドレスを入力してください。



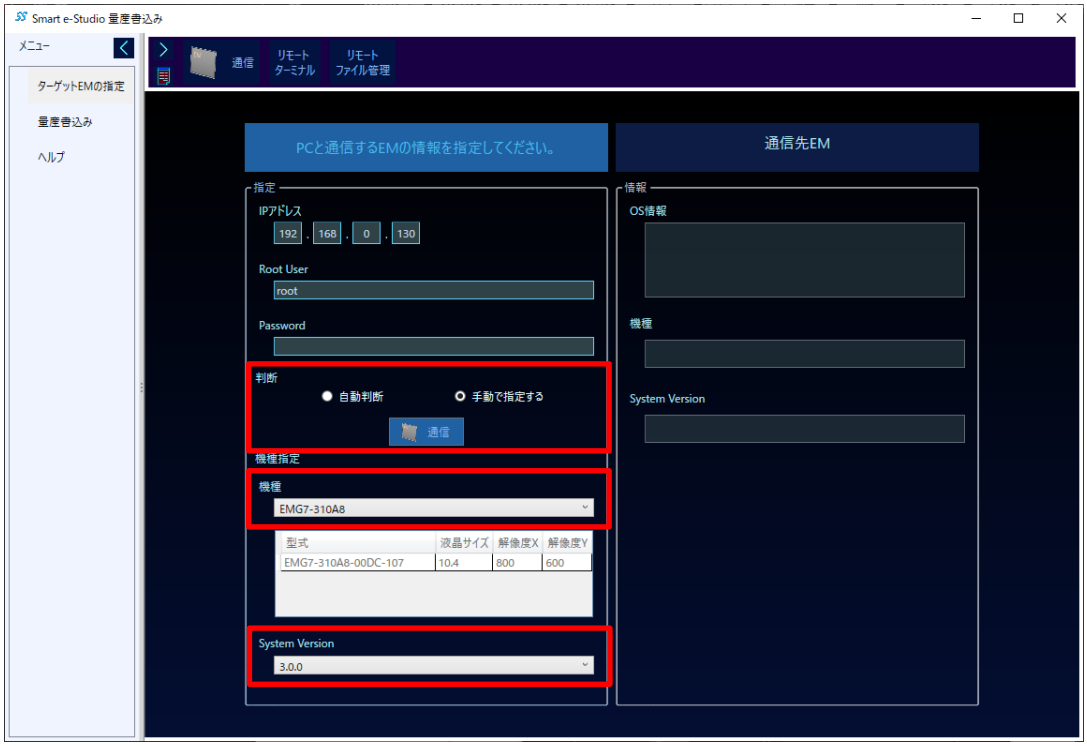

5. 「自動判断」にチェックを入れて、「通信」ボタンを押してください。



右側の欄に EM シリーズ本体の情報が表示されれば、成功です。



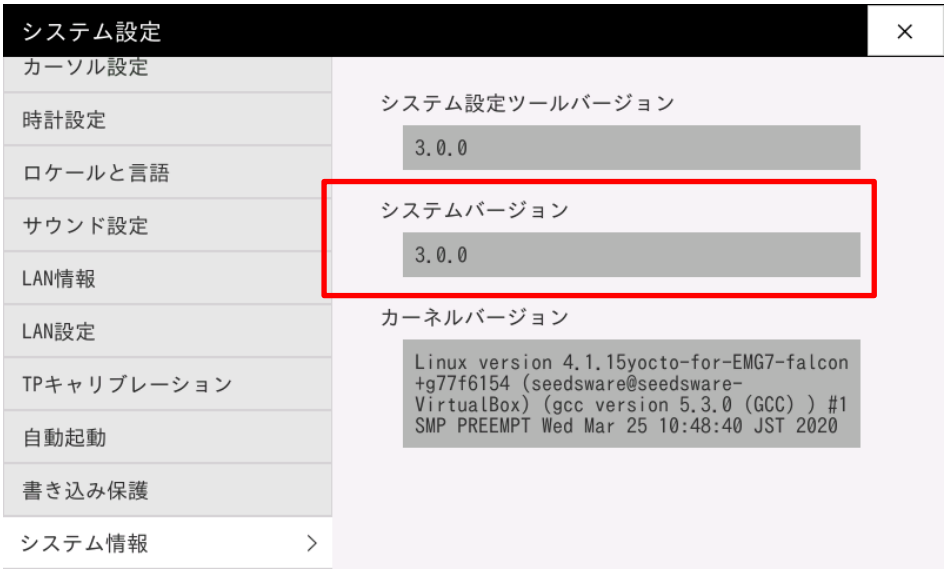

※　自動判断が行えない場合は、「手動で指定する」にチェックを入れて、「機種」と「System Version」を指定後、「通信」ボタンを押してください。



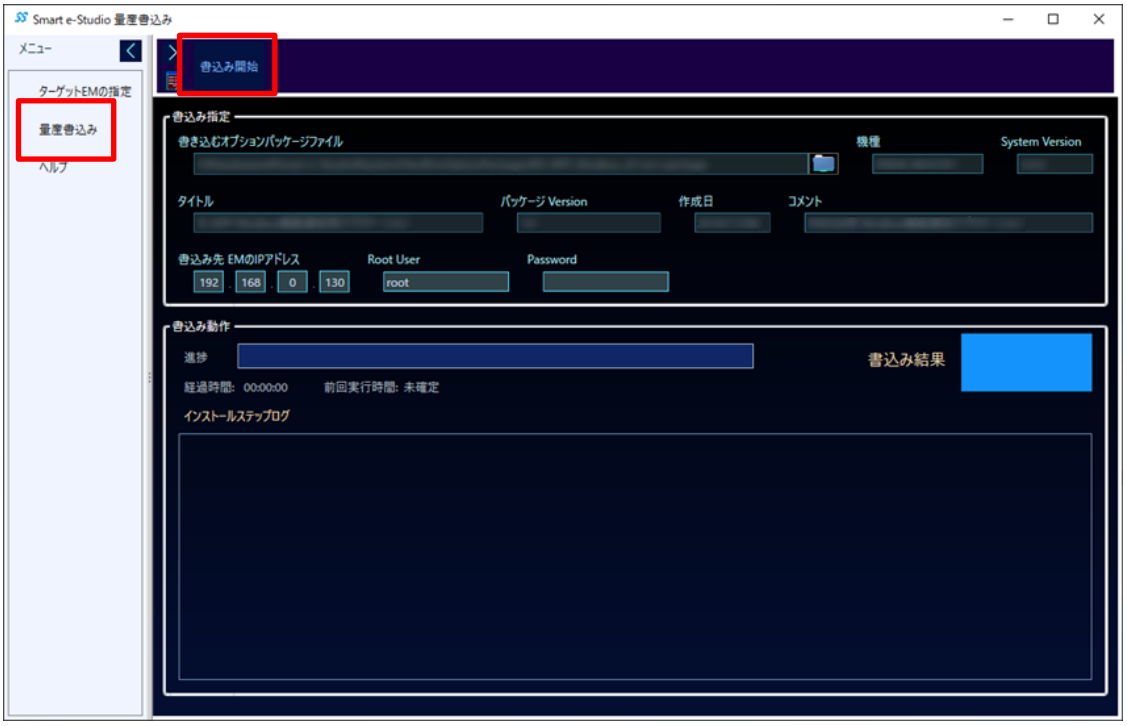
「機種」は以下からお使いの型式に合わせたものを選択してください。

| 機種 | 型式 | 備考 |
|---|---|---|
| EM8-205A7 | EM8-205A7-****-*07 | 5.7 型、オープンフレームタイプ、アナログ抵抗膜モデル |
| EM8-205A7+SStart | EM8-205A7-****-*57 | 5.7 型、オープンフレームタイプ、アナログ抵抗膜モデル、Smart e-Start 対応 |
| EM8-W104A7 | EM8-W104A7-****-*07 | 4.3 型、オープンフレームタイプ、アナログ抵抗膜モデル |
| EM8-W104A7+SStart | EM8-W104A7-****-*57 | 4.3 型、オープンフレームタイプ、アナログ抵抗膜モデル、Smart e-Start 対応 |
| EM8-W207A7 | EM8-W207A7-****-*07 | 7W 型、オープンフレームタイプ、アナログ抵抗膜モデル |
| EM8-W207A7+SStart | EM8-W207A7-****-*57 | 7W 型、オープンフレームタイプ、アナログ抵抗膜モデル、Smart e-Start 対応 |
| EM8-W310A7 | EM8-W310A7-****-*07 | 10.1W 型、オープンフレームタイプ、アナログ抵抗膜モデル |
| EMG7-310A8 | EMG7-310A8-****-*07 | 10.4 型、ベゼルタイプ、投影型静電容量モデル |
| EMG7-312A8 | EMG7-312A8-****-*07 | 12.1 型、ベゼルタイプ、投影型静電容量モデル |
| EMG7-W207A8 | EMG7-W207A8-****-*07 | 7W 型、ベゼルタイプ、投影型静電容量モデル |
| EMG8-205A7 | EMG8-205A7-****-*07 | 5.7 型、オープンフレームタイプ、投影型静電容量モデル |
| EMG8-205A7+SStart | EMG8-205A7-****-*57 | 5.7 型、オープンフレームタイプ、投影型静電容量モデル、Smart e-Start 対応 |
| EMG8-W104A7 | EMG8-W104A7-****-*07 | 4.3 型、オープンフレームタイプ、投影型静電容量モデル |
| EMG8-W104A7+SStart | EMG8-W104A7-****-*57 | 4.3 型、オープンフレームタイプ、投影型静電容量モデル、Smart e-Start 対応 |
| EMG8-W207A7 | EMG8-W207A7-****-*07 | 7W 型、オープンフレームタイプ、投影型静電容量モデル |
| EMG8-W207A7+SStart | EMG8-W207A7-****-*57 | 7W 型、オープンフレームタイプ、投影型静電容量モデル、Smart e-Start 対応 |
| EMP-W207A7 | EMP-W207A7-****-*07 | 7W 型、ペンダントモデル |
| EMP-W207A7+SStart | EMP-W207A7-****-*57 | 7W 型、ペンダントモデル Smart e-Start 対応 |

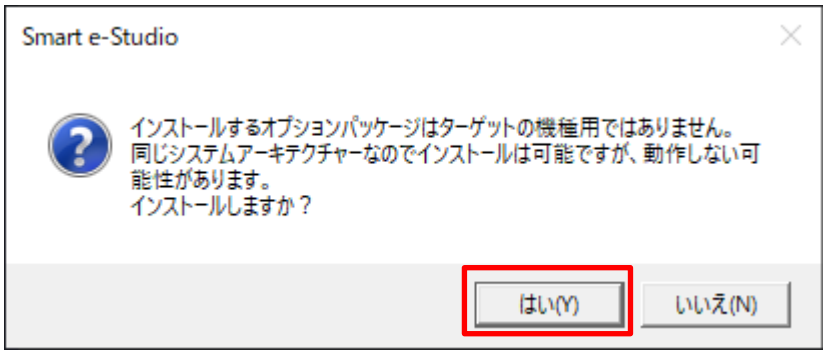
「System Version」は、ターゲット EM 本体にインストールされている「システム設定ツール」を起動し、「システム情報」メニュー「システムバージョン」に表示されている値を選択してください。「システム設定ツール」については、別紙「EM シリーズ　ツールマニュアル」を参照ください。



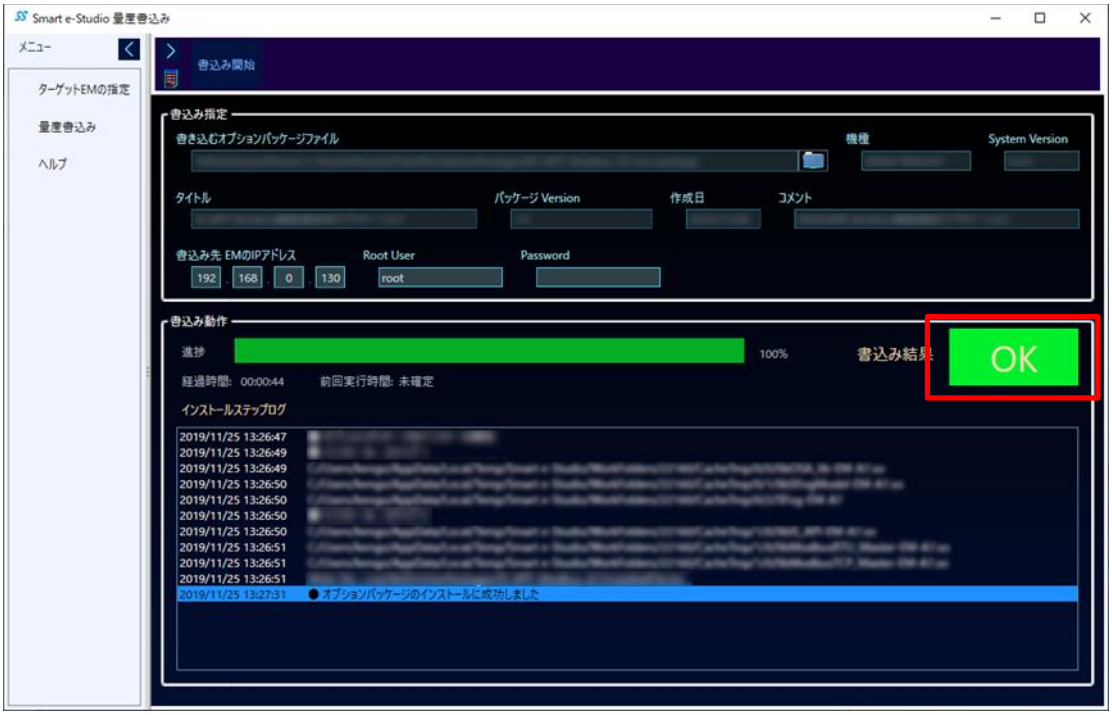

6. 通信成功の確認後、「量産書き込み」ページを開き、「書込み開始」を押してください。



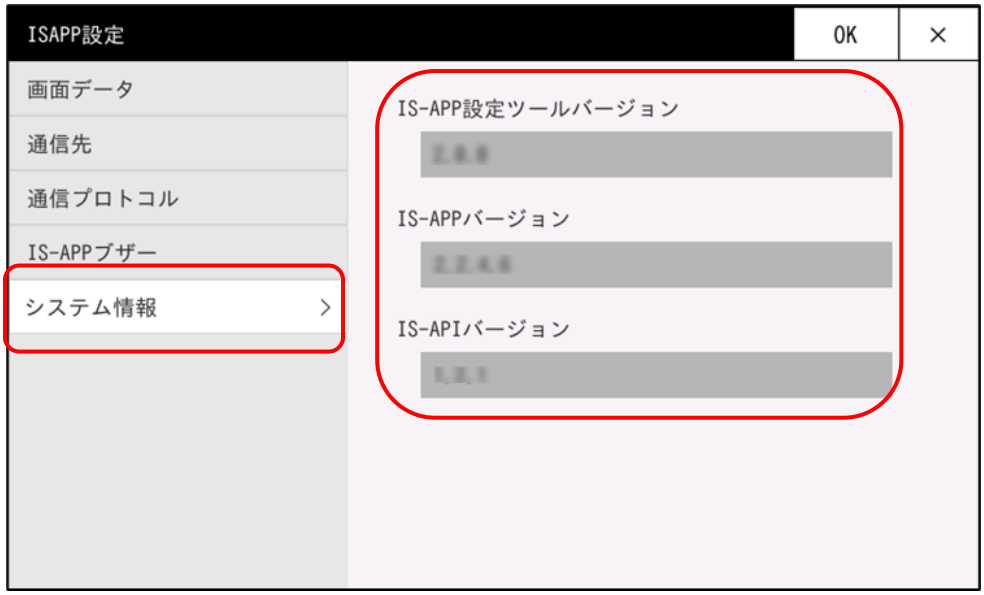
7. 以下のダイアログが表示された場合は「はい」を選択してください。


書込み結果に「OK」が表示されると完了です。



以上で更新完了です。
IS-APP 設定ツールのシステム情報から、IS-APP、IS-API、IS-APP SETTING（IS-APP 設定ツール）が対象のバージョンに更新されていることをご確認ください。




対象のバージョンは、別紙「InfoSOSA_Version2.5-2.7_リリースノート」のモジュールバージョンを参照ください。

## 1.6.2　手動で差し替える

更新パッケージファイルを使用せずに手動で各ファイルを差し替えることも可能です。
コンソールの接続方法、データの転送方法は、別紙「EM シリーズ　ソフトウェア開発マニュアル」を参照ください。

1. EM シリーズにコンソールを接続してください。

2. コンソールで以下の書き込み保護を無効にするコマンドを実行してください。

```
# wprotect_off
```

※再起動すると読み取り専用に戻ります。

3. EM シリーズ本体へ「is_app」「libisapi.so」「isapp_setting」を転送（上書き）してください。

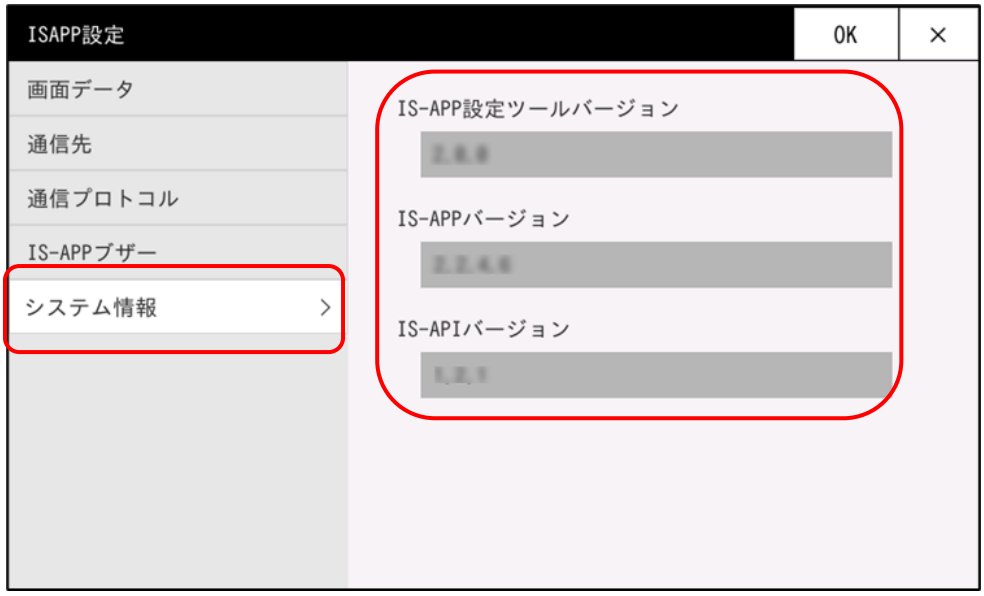
| ファイル名 | 転送先 |
|---|---|
| is_app | /usr/bin/is_app |
| libisapi.so | /usr/lib/libisapi.so |
| isapp_setting | /usr/bin/isapp_setting |

「is_app」「libisapi.so」「isapp_setting」は、InfoSOSA 開発キットデータ内にございます。
お使いの製品ごとに対応ファイルが異なりますのでご注意ください

| 型式 | 対応ファイル |
|---|---|
| EMG7-***A8-****-**7 | [InfoSOSA 開発キットデータ]¥software¥IS-APP¥IS-APP-A8¥is_app<br>[InfoSOSA 開発キットデータ]¥software¥IS-API¥Library¥IS-APP-A8¥libisapi.so<br>[InfoSOSA 開発キットデータ]¥software¥IS-APP_SETTING¥IS-APP-A8¥ isapp_setting |
| EM8-***A7-****-**7<br>EMG8-***A7-****-**7<br>EMP-***A7-****-**7 | [InfoSOSA 開発キットデータ]¥software¥IS-APP¥IS-APP-A7¥is_app<br>[InfoSOSA 開発キットデータ]¥software¥IS-API¥Library¥IS-APP-A7¥libisapi.so<br>[InfoSOSA 開発キットデータ]¥software¥IS-APP_SETTING¥IS-APP-A7¥isapp_setting |

4. EM シリーズ本体を再起動してください。

以上で更新完了です。
IS-APP 設定ツールのシステム情報から、IS-APP、IS-API、IS-APP SETTING（IS-APP 設定ツール）が対象のバージョンに更新されていることをご確認ください。




対象のバージョンは、別紙「InfoSOSA_Version2.5-2.7_リリースノート」のモジュールバージョンを参照ください。

# 1.7　IS-APPを使用したHMIの開発方法

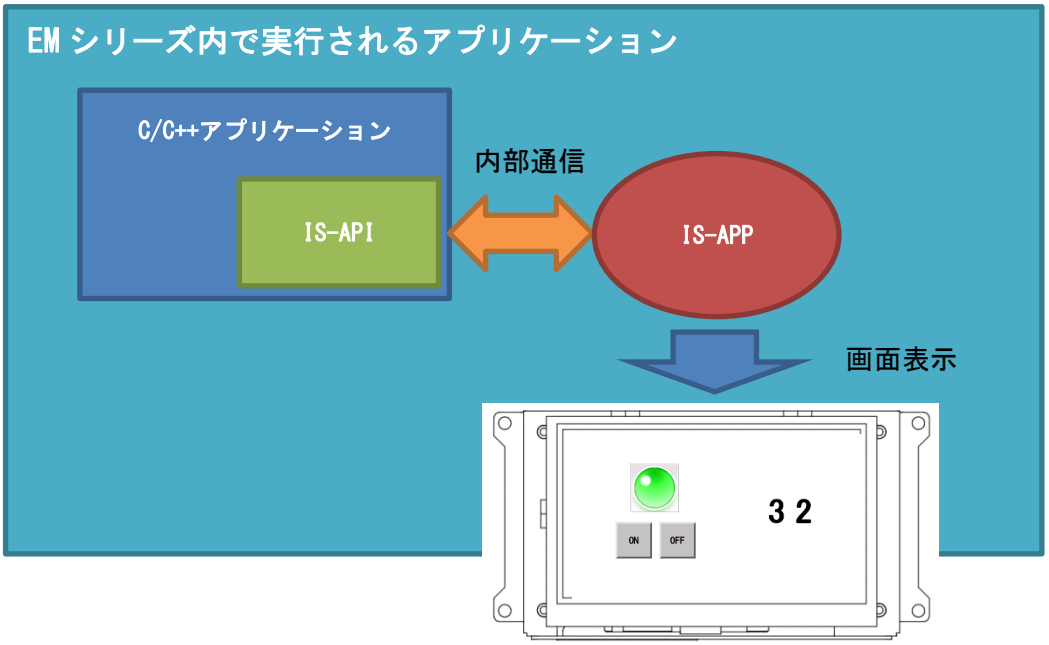
## 概要

IS-APP を使って次のようなシステムを作る手順を説明します。

- 画面表示は IS-APP が行う
- IS-APP で表示する画面は、InfoSOSA ビルダで作成する（プログラミング不要）
- IS-APP の機能では実現が難しい処理をユーザ作成アプリケーションで行う（C/C++などで作成）
- ユーザ作成アプリケーションからは IS-APP へのアクセスは IS-APP アクセス用 API（IS-API）を使用する
- 上位機器（外部）との通信は、ユーザ作成アプリケーションが行う　※本チュートリアルでは省略

本チュートリアルでは、以下のような動作の画面、プログラムを作成します。

✓ 液晶画面上（IS-APP）の ON／OFF ボタンをタッチすると画面上のランプが点灯消灯する
✓ 液晶画面上（IS-APP）の ON／OFF ボタンをタッチするとコンソールアプリケーション（C++プログラム）に状態が通知される
✓ コンソールから数値を入力すると液晶画面上（IS-APP）の数字が変化する




EM シリーズ

## 開発の流れ

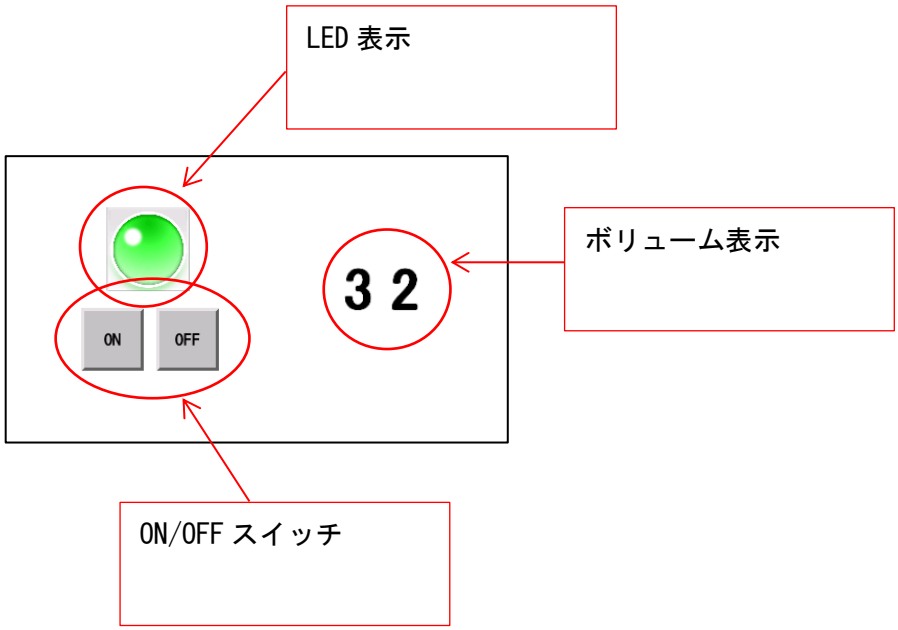
| No. | 項目 | 内容 |
|---|---|---|
| ① | システム仕様作成 | IS-APP で行う表示、C/C++アプリケーションで行う処理、上位機器との通信などシステム全体の仕様を作成します。 |
| ② | InfoSOSA プロジェクトの作成 | InfoSOSA ビルダで IS-APP で表示する画面を保存するファイルを作成します。 |
| ③ | InfoSOSA 画面の作成 | InfoSOSA ビルダでボタンの配置や動作などの設定を行います。 |
| ④ | InfoSOSA プロジェクトの保存 | InfoSOSA ビルダで作成した画面データを保存します。 |
| ⑤ | InfoSOSA シミュレータでの動作確認 | PC 上で画面データの動作確認を行います。 |
| ⑥ | C++アプリケーションの作成 | IS-API を使用して、IS-APP と連携する C++アプリケーションを作成します。 |
| ⑦ | EM シリーズへ転送 | EM シリーズへ IS-APP の実行ファイル、画面データ、IS-API ライブラリ、C++アプリケーションを転送します。 |
| ⑧ | IS-APP 単体テスト | IS-APP を起動し、単体動作を確認します。 |
| ⑨ | IS-APP、C++アプリケーション結合テスト | C++アプリケーションを起動し、結合テストを行います。 |

## 1.7.1　システム仕様作成

システム全体の仕様を作成します。
本チュートリアルでは以下の仕様とします。

### 画面表示(IS-APP で行う処理)

表示する画面は、InfoSOSA ビルダで作成する。
ON/OFF ボタンを押すと画面内のランプが点灯／消灯し、C++アプリケーションにボタンが押されたことを通知する



## C++アプリケーションで行う処理

IS-API を使用して、コンソールで入力した数値に画面のボリューム表示を変える。
IS-APP からの通知（ボタンが押されたこと）をコンソールに表示する

## メモリ定義

以下の２つのメモリを定義

| メモリ ID | 型 | 値範囲 | 内容 |
|---|---|---|---|
| GM_LED01 | ブール | 0、1 | LED の状態 |
| GM_VOL01 | バイト | 0～100 | ボリュームの状態 |

InfoSOSA ではビルダでユーザが自由に InfoSOSA 内にメモリを作成することができます。このメモリの値を上位機器や他のアプリケーションに通知して処理を行ったり、IS-API を使用してメモリの値を変更し InfoSOSA の表示を変えたりすることができます。

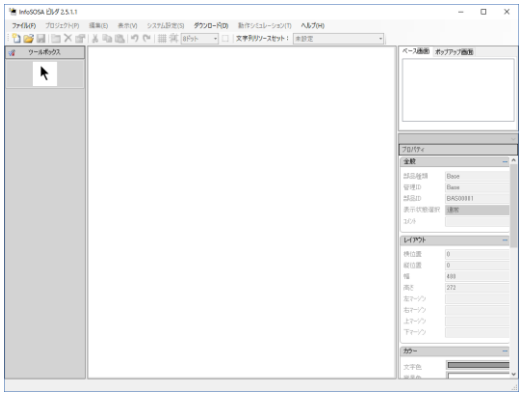
## 1.7.2　InfoSOSA プロジェクトの作成

ここからは InfoSOSA ビルダを使用してプロジェクトの作成を行います。
次に進む前に PC(Windows)に InfoSOSA ビルダのインストールを行ってください。
インストール方法は「1.4　InfoSOSA ビルダのインストール」を参照ください。

**MEMO**

◆プロジェクトとは？
InfoSOSA ビルダで作成した画面や、上位との通信設定などが保存されたファイルの事です。

### InfoSOSA ビルダの起動

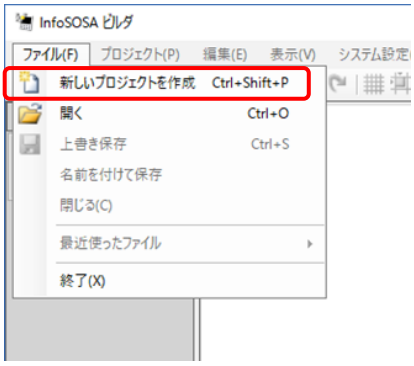


InfoSOSA ビルダはデスクトップの 

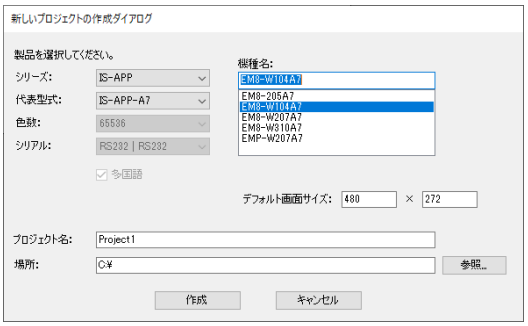
アイコン、またはスタートメニューの「Seedsware」→「InfoSOSA Builder*.*」から起動します。

ビルダを起動すると左のような画面が表示されます。

### 新規プロジェクトの作成




ビルダ左上の「ファイル」メニューから「新しいプロジェクトを作成」をクリックします。




左のようなダイアログが表示されます。
ご使用の製品に対応した機種名を選択します。
選択したら、プロジェクト名を入力し、「作成」ボタンを押してプロジェクトを作成します。
※プロジェクトは「場所」で示されたところに作成されます。

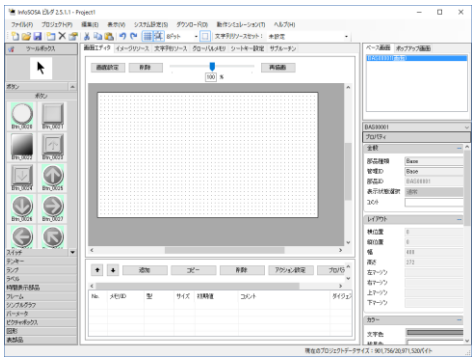
**MEMO**

プロジェクト名と場所は自由に設定できます。

IS-APP では、型式ごとに決められた代表型式、機種名を選択します。
シリーズ名を「IS-APP」に設定し、お使いの型式に対応した代表型式、機種名を選択してください。

| 製品型式 | 代表型式 | 機種名 |
|---|---|---|
| EM8-W104A7-****-1*7 | IS-APP-A7 | EM8-W104A7 |
| EMG8-W104A7-****-1*7 | | |
| EM8-W104A7-****-2*7 | | |
| EMG8-W104A7-****-2*7 | | |
| EM8-205A7-****-1*7 | | EM8-205A7 |
| EMG8-205A7-****-1*7 | | |
| EM8-205A7-****-2*7 | | |
| EMG8-205A7-****-2*7 | | |
| EM8-W207A7-****-1*7 | | EM8-W207A7 |
| EMG8-W207A7-****-1*7 | | |
| EM8-W207A7-****-2*7 | | |
| EMG8-W207A7-****-2*7 | | |
| EM8-W310A7-****-1*7 | | EM8-W310A7 |
| EMG8-W310A7-****-1*7 | | |
| EM8-W310A7-****-2*7 | | |
| EMG8-W310A7-****-2*7 | | |
| EMP-W207A7-****-2*7 | | EMP-W207A7 |
| EMG7-W207A8-****-1*7 | IS-APP-A8 | EMG7-W207A8 |
| EMG7-310A8-****-1*7 | | EMG7-310A8 |
| EMG7-312A8-****-1*7 | | EMG7-312A8 |

【ご注意】
対応してない機種名を選択した場合は、本体に転送しても実行エラーになります。正しい機種名を選択してください。



プロジェクトを作成すると左のような画面が表示されます。

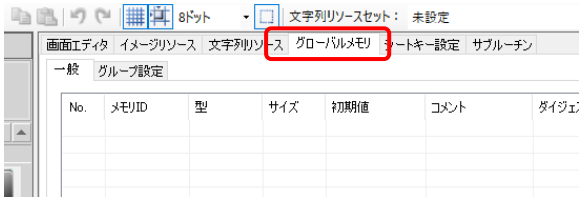
# 1.7.3　InfoSOSA 画面の作成

部品やメモリの設定を行います。

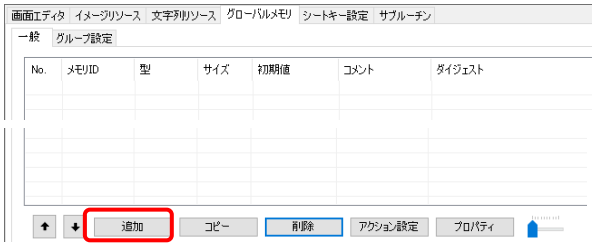
## グローバルメモリの設定

まず、InfoSOSA 内のメモリを作成します。
メモリとは、InfoSOSA 内で値を保持しておく入れ物です。

### メモリの作成




「グローバルメモリ」タブを選択します。



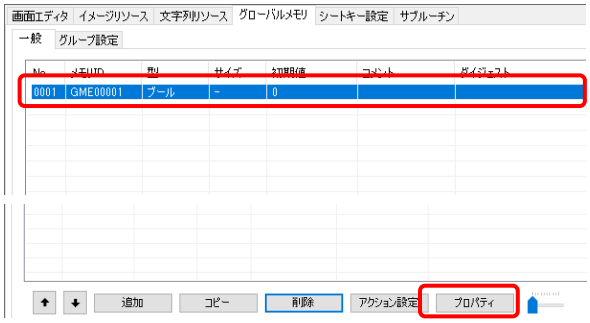

「追加」ボタンをクリックします。


メモリが作成されました。

**MEMO**

◆メモリの種類
アプリケーション実行中は値が保持され常にアクセス可能なグローバルメモリと、特定の画面表示中のみアクセス可能で、別画面を表示すると初期化される画面メモリがあります。

## プロパティ設定




メモリを選択して、「プロパティ」ボタンをクリックしてください。


プロパティを表に合わせて変更してください。

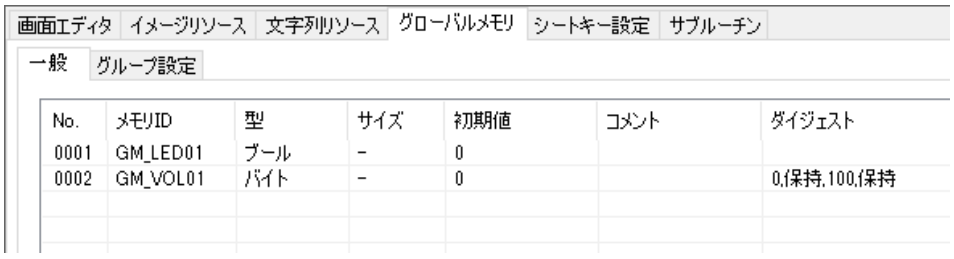
| プロパティ | 値 | プロパティの意味 |
|---|---|---|
| メモリ ID | GM_LED01 | メモリを識別する ID |
| 型 | ブール | メモリの種類 |
| 初期値 | 0 | 電源投入時の初期値 |

### ２つ目のメモリの作成

同様の手順でメモリをもう１つ作成します。２つ目のメモリは以下のようにプロパティを設定してください。

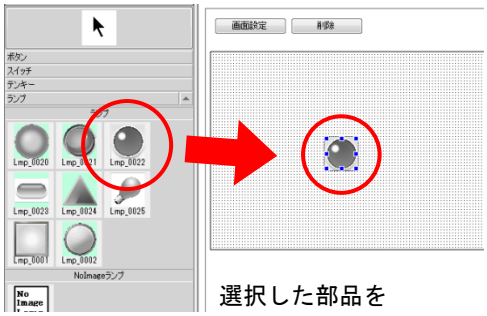
| プロパティ | 値 | プロパティの意味 |
|---|---|---|
| メモリ ID | GM_VOL01 | メモリを識別する ID |
| 型 | バイト | メモリの種類 |
| 初期値 | 0 | 電源投入時の初期値 |
| 最小値 | 0 | メモリの最小値 |
| 最大値 | 100 | メモリの最大値 |
| アンダーフロー | 保持 | メモリの最小値より小さな値が設定された場合の動作 |
| オーバーフロー | 保持 | メモリの最大値より大きな値が設定された場合の動作 |

設定後のリストは以下のようになります。



# ランプ部品の設定

## ランプの配置



選択した部品を
ドラッグ＆ドロップで配置

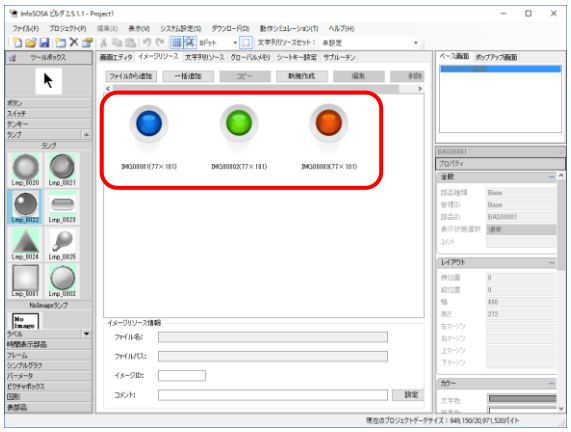
ビルダ左のツールボックスから「ランプ」を選択し、「LMP_00022」を画面にドラッグ＆ドロップしてください。

## ランプの画像変更


配置したランプを選択した状態で、ビルダ右部のプロパティエリアから「イメージ」の項目を設定します。
ここの「動作イメージ」と「通常イメージ」を設定すると、ランプのイメージが変更できます。
「動作イメージ」はランプが ON の時のイメージを、「通常イメージ」はランプが OFF の時のイメージを設定しています。
今回のチュートリアルでは、以下に設定しています。

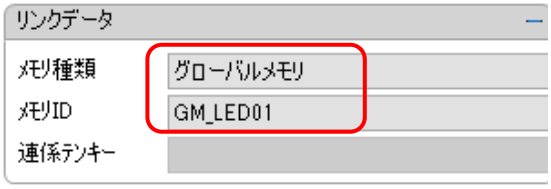
- 動作イメージ：　緑 ON
- 通常イメージ：　緑 OFF

この様にビットマップ部品のイメージを変更することができます。



イメージリソースに画像ファイルを取り込むことで、デフォルトに無い見た目のランプも作成することができます。

### ランプのリンク設定




ランプとグローバルメモリの関連付けを行います。リンク設定を行うと指定したグローバルメモリの値が 0 の時に「通常イメージ」、1 の時に「動作イメージ」が表示されるようになります。

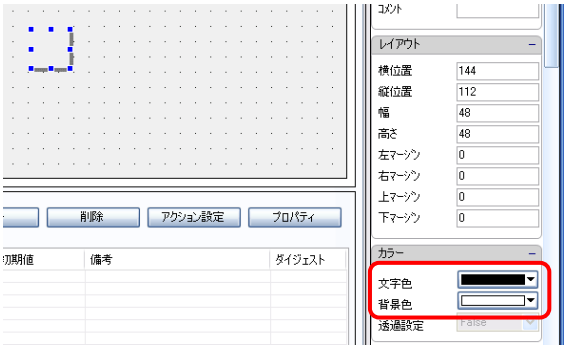
今回のチュートリアルでは、以下に設定しています。
・メモリ種類：　グローバルメモリ
・メモリ ID　：　GM_LED01

## ボタン部品の設定

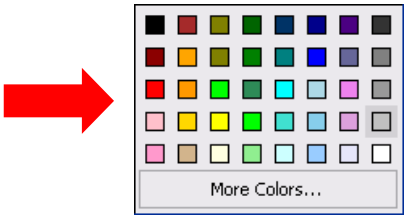
### ボタンの配置


ビルダ左のツールボックスから「ボタン」を選択し「Btn_0099」を、画面にドラッグ＆ドロップしてください。表示されていない場合は、▼でスクロールします。

### ボタンの色を変更

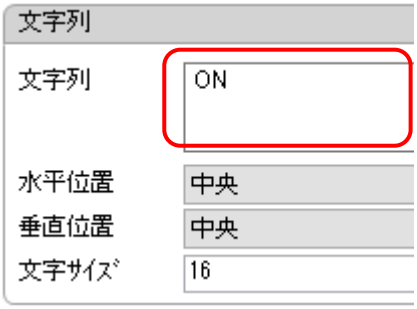



配置したボタンを選択した状態で、ビルダ右のプロパティから「カラー」の項目を設定します。
ここの「背景色」をクリックすると、以下のようなパレットが出現し、ボタンの色を変更できます。




この様に NoImage の部品は色を変更することができます。

## ボタンの文字を変更



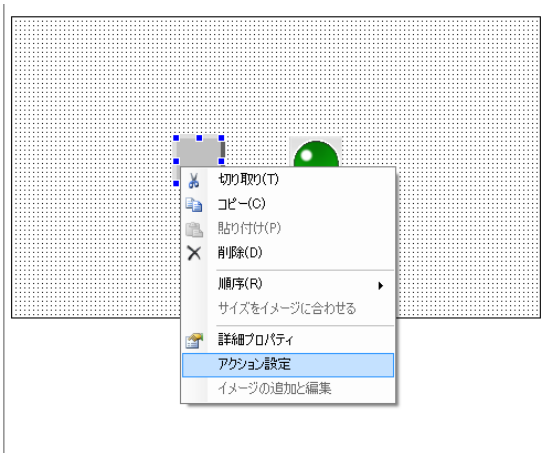

文字列プロパティを変更することで、ボタン上に文字を表示することができます。

今回のチュートリアルでは、以下に設定しています。

・文字列：　ON

## アクション設定①

ボタンを押した時に画面内のランプが点灯するようにボタンにアクションを設定します。

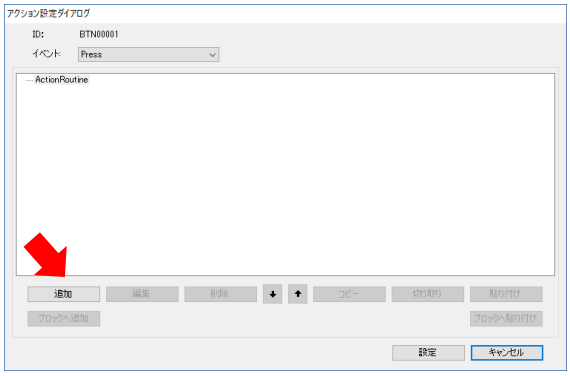



アクションを設定するには、画面に配置したボタンを右クリックし、「アクション設定」をクリックします。

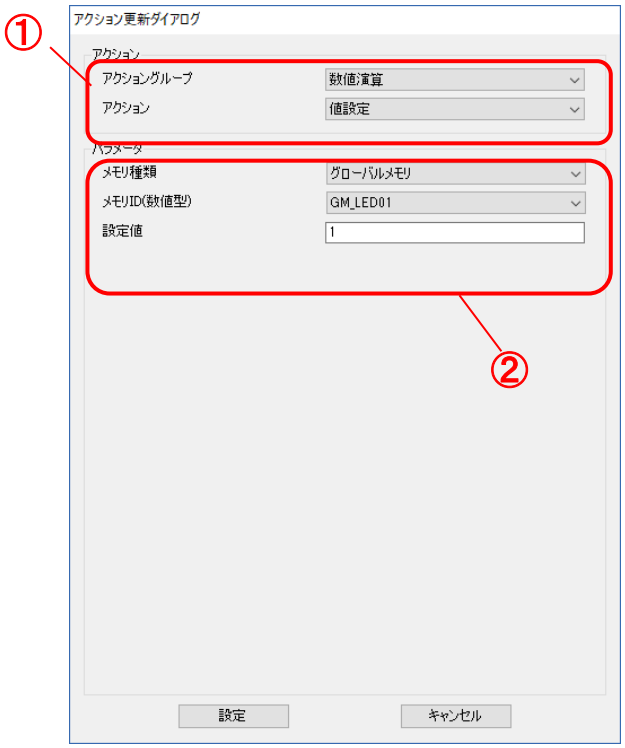
**MEMO**

◆アクションとは？

画面に配置した部品を動作させるための設定のことをいいます。




アクション設定ダイアログが開いたら、左下の「追加」を押してアクション追加ダイアログを開きます。




アクション追加ダイアログには以下の項目があります。

①アクション

・アクショングループ

⇒使用したいアクションのカテゴリを選択します。

・アクション

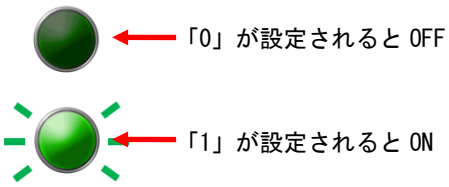
⇒使用したいアクションを選択します。

②パラメータ

⇒どの画面の、どの部品に対してアクションを設定するか、どのような値を設定するか等を設定します。

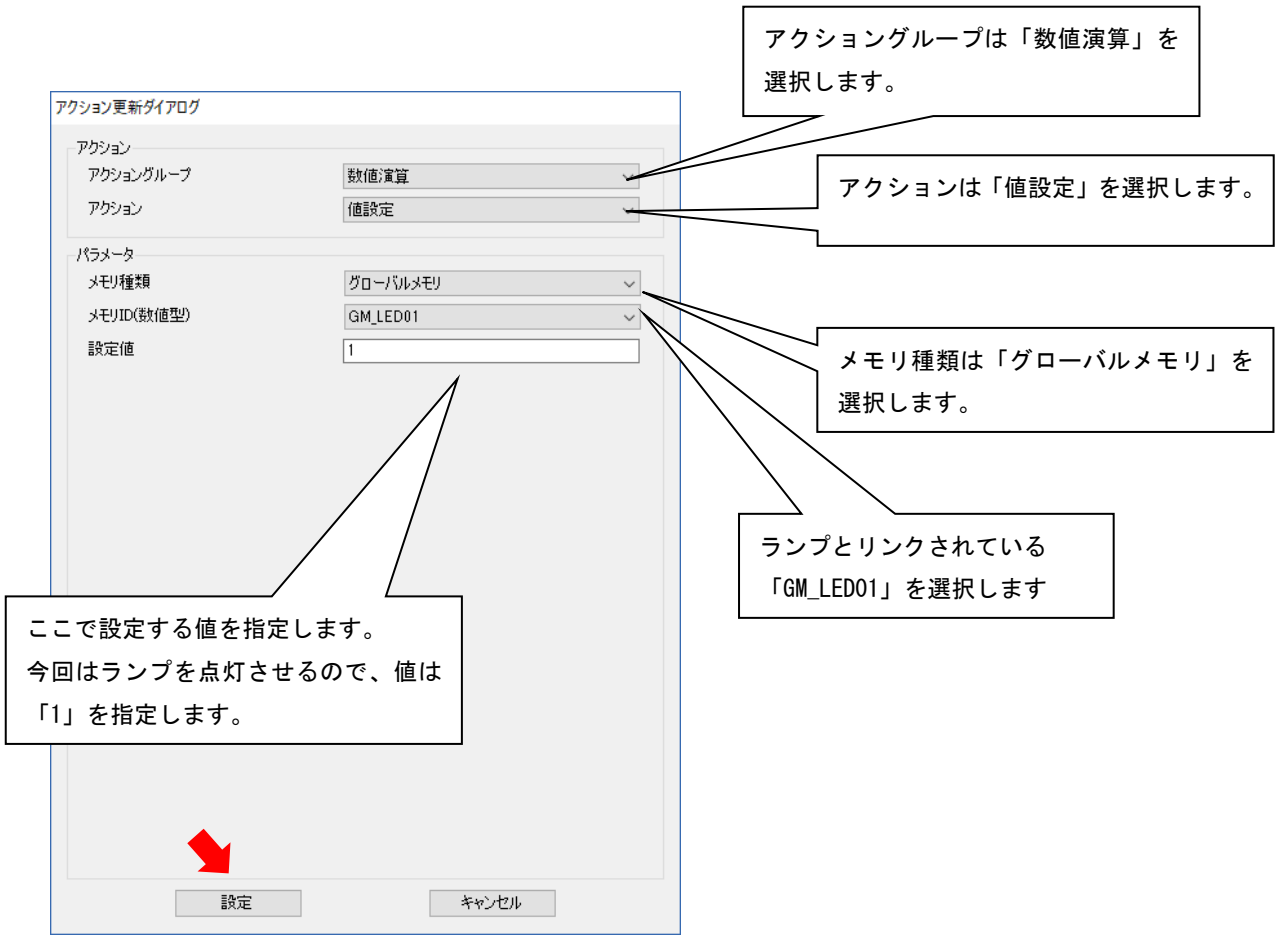
設定できる項目は、画面 ID や部品 ID、値等があります。

ただし、設定できる項目は使用するアクションによって変わります。



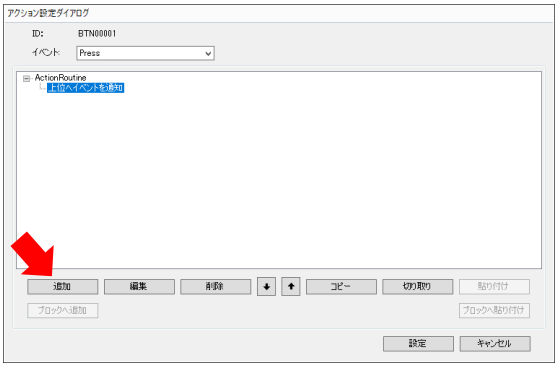

今回は「ランプを点灯させる」というアクションを設定します。ランプはリンク設定されているメモリに「0」が設定されると OFF、「1」が設定されると ON になります。

ランプを点灯させるアクションの設定は以下の通りです。



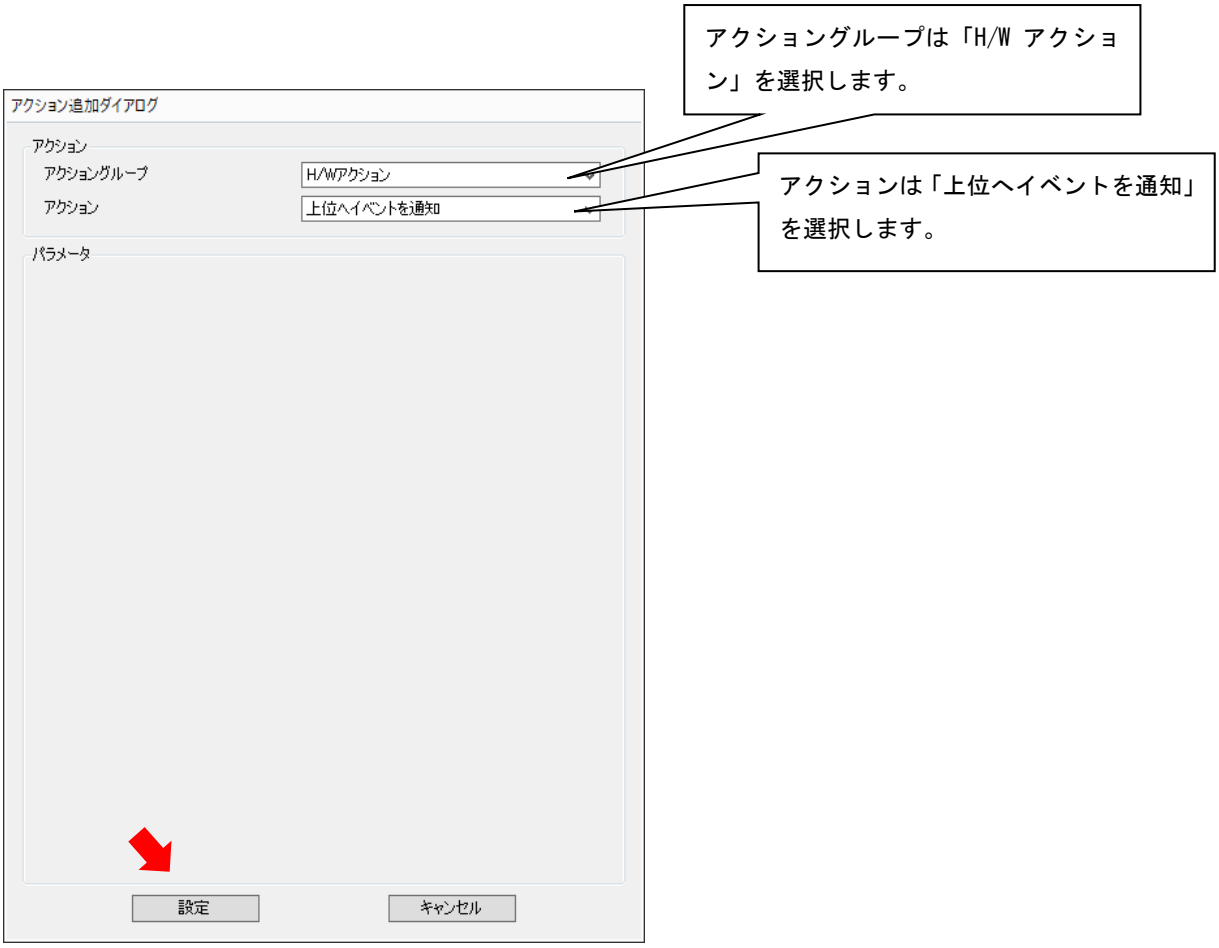
## アクション設定②

続けて、ボタンを押したこと（イベント発生）を通信先に通知する設定を行います。



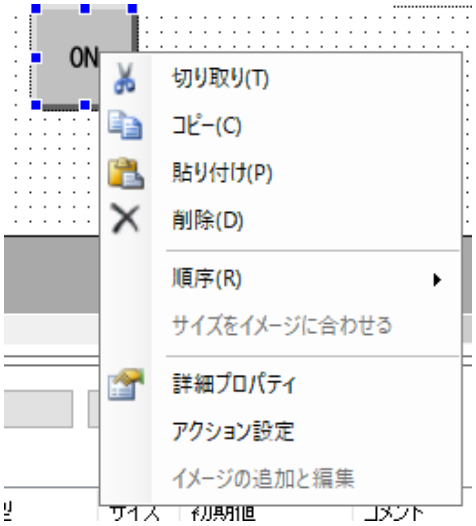

左下の「追加」を押してアクション追加ダイアログを開きます。

イベントの発生を通信先に通知するアクションの設定は以下の通りです。



## ２つ目のボタンの作成

ランプを OFF にするボタンを作成します。




ON ボタンを右クリックして、「コピー」をクリックした後、再度右クリックを行い「貼り付け」をクリックしてください。

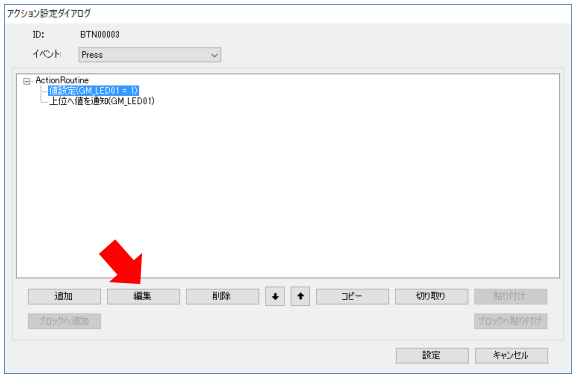
ON ボタンがコピーされるので、変更箇所のみを設定してください。

表示文字を「OFF」に変更

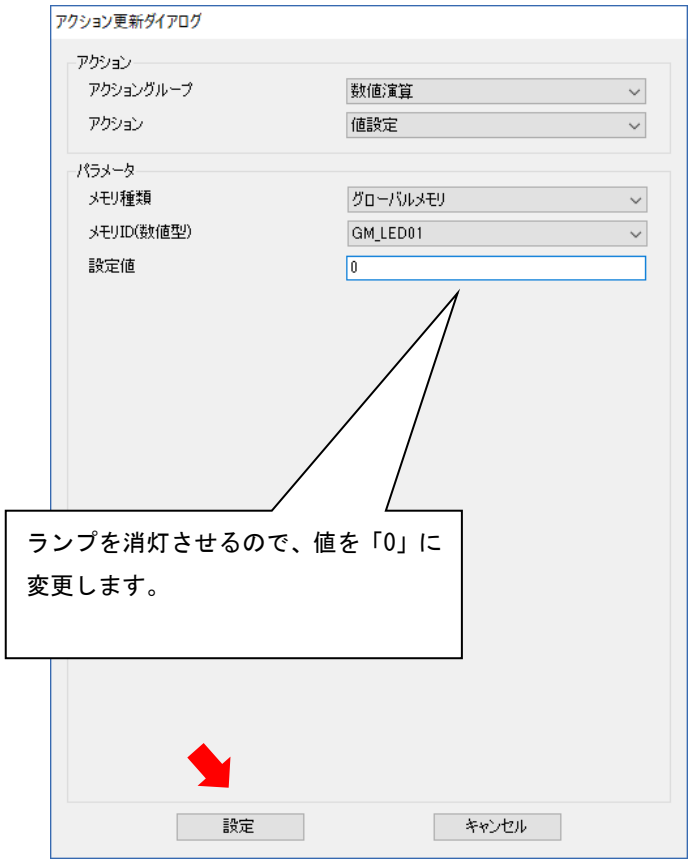

文字列プロパティを OFF

・文字列：　OFF

ランプを消灯させるアクションに変更

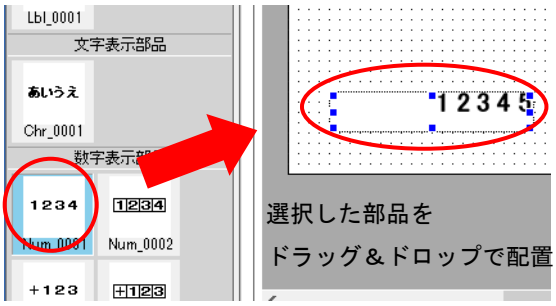



ランプを点灯させるアクションを選択した状態で左下の「編集」を押してアクション更新ダイアログを開きます。



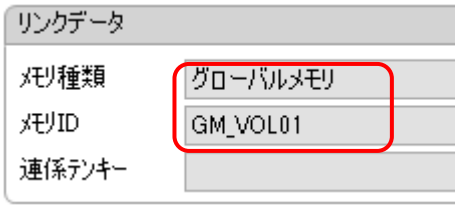
# 数字表示部品の設定

## 数字表示の配置




ビルダ左のツールボックスから「ラベル」を選択し「Num_0001」を、画面にドラッグ＆ドロップしてください。

## 数字表示のリンク設定



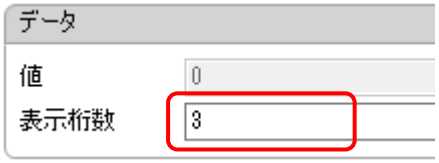

数字表示部品とグローバルメモリの関連付けを行います。リンク設定を行うと指定したグローバルメモリの値を表示します。

今回のチュートリアルでは、以下に設定しています。

・メモリ種類：　グローバルメモリ

・メモリ ID　：　GM_VOL01

## 数字表示の表示桁数を変更

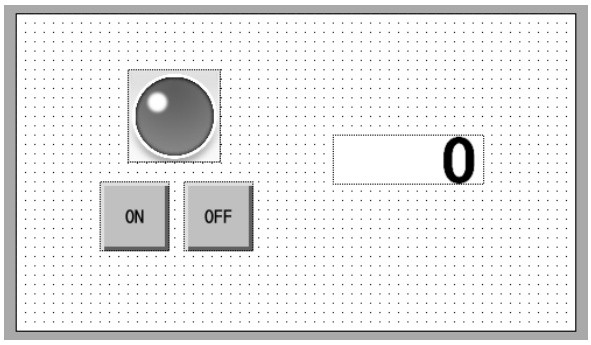



表示する数字の桁数を設定します。

今回のチュートリアルでは、以下に設定しています。

・表示桁数：　3

### 各部品の位置とサイズの調整

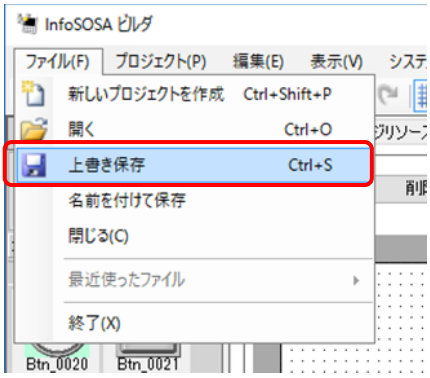



マウスで部品の位置とサイズを調整します。
サイズは部品の端をドラッグすると変更することができます。

## 1.7.4　InfoSOSA プロジェクトの保存

作成したプロジェクトを保存します。

### プロジェクトの保存



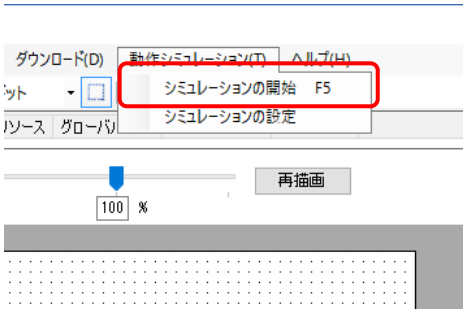

作成したプロジェクトを保存します。
ビルダ左上の「ファイル」メニューを開き、「上書き保存」をクリックすると、プロジェクトの保存ができます。又は、ツールバーの保存ボタンを押しても、上書き保存が可能です。

※ 別名でプロジェクトを保存する場合は「名前を付けて保存」をクリックします。

# 1.7.5　InfoSOSA シミュレータでの動作確認

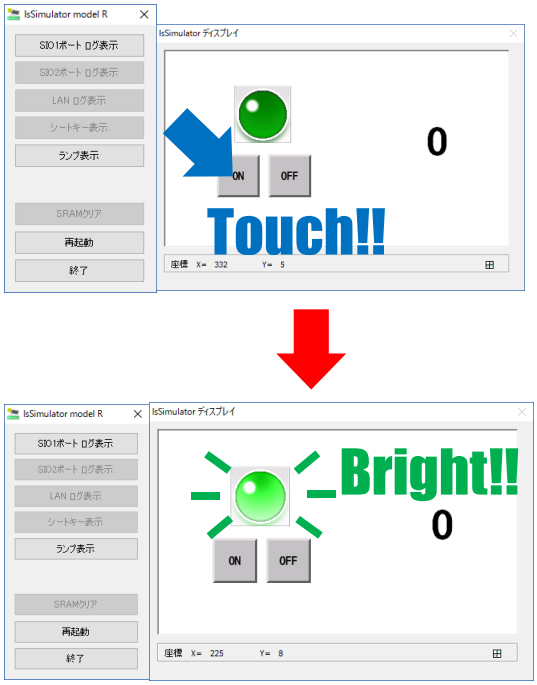
作成したプロジェクトをシミュレータで動作確認します。

## シミュレータの起動




作成したプロジェクトの動作を PC 上で確認します。
シミュレータの起動は、「動作シミュレーション」メニューの「シミュレーションの開始」から行います。

## シミュレータでの動作確認




シミュレータには作成した画面が表示されています。

設定した通りの動作をするか、シミュレータ上に表示されているボタンをクリックして確認します。

アクションが正しく設定されていると、ランプが点灯します。
ランプの点灯を確認できたらシミュレーションは完了です。

## 1.7.6　C++アプリケーションの作成

IS-API ライブラリを使用して、IS-APP と連携するアプリケーションを作成します。
ここからは、EM シリーズのソフトウェア開発環境(Linux 仮想マシン)上で、ソースファイルを作成します。事前にソフトウェア開発環境の作成が必要です。
ソフトウェア開発環境の作成方法については、別紙「EM シリーズ ソフトウェア開発マニュアル」を参照ください。

### 関連ファイルのコピー

IS-API を使用するには、ヘッダファイルとライブラリファイルが必要です。
保存ディレクトリに以下のファイルをコピーしてください。

| 保存ディレクトリ※ | /home/em/test/ |
|---|---|

※保存ディレクトリが無い場合は作成してください。保存ディレクトリは変更できます。

| ライブラリファイル |
|---|
| libisapi.so |

| ヘッダファイル |
|---|
| CIsApi.h |
| CIsComString.h |
| CIsComVariant.h |
| CIsComVariantList.h |

ライブラリファイル／ヘッダファイルは、InfoSOSA 開発キットデータ内の「software¥IS-API」フォルダに収録しています。

ライブラリファイルは、型式によりファイルが異なります。
対応フォルダ内のファイルをご使用ください。

| 型式 | 対応フォルダ |
|---|---|
| EMG7-***A8-****-**7 | IS-APP-A8 |
| EM8-***A7-****-**7 | IS-APP-A7 |
| EMG8-***A7-****-**7 | IS-APP-A7 |
| EMP-***A7-****-**7 | IS-APP-A7 |

## ソースファイル作成

テキストエディタを起動して、ソースファイルを作成します。ここでは、以下のように入力してください。ソースファイルの文字コードは UTF8 で作成してください。

作成したソースファイルは以下のように保存してください。

| | |
|---|---|
| 保存ディレクトリ※ | /home/em/test/ |
| ファイル名 | tutorial.cpp |
| 文字コード | UTF8 |

※保存ディレクトリが無い場合は作成してください。
　保存ディレクトリ、ファイル名は変更できます。

```
// [1] ヘッダファイルのインクルード
#include "CIsApi.h"
#include "CIsComString.h"
#include "CIsComVariant.h"
#include "CIsComVariantList.h"
#include <iostream>
#include <string>

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

CIsApi* pIsApi = NULL;

int main()
{
    // [2] CIsApi クラスオブジェクトの作成
    pIsApi = new CIsApi(51111);
    std::cout << "Connecting..." << std::flush;

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

    // [4] 動作状態確認
    if(!pIsApi->IsRunning()){
        return false;
    }
```

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

    // [6] コールバック関数登録
    std::string str01("BAS00001.BTN00001.PRESS");  //ON ボタンイベント ID
    std::string str02("BAS00001.BTN00002.PRESS");  //OFF ボタンイベント ID
    pIsApi->registerCallback(&str01, event_callback01);  //ON ボタン
    pIsApi->registerCallback(&str02, event_callback02);  //OFF ボタン

    // [7] IS-APP のメモリの値を変更する
    CIsComVariant var;
    std::string  propertyID="@GLBMEM.GM_VOL01.VALUE";
    while(true)
    {
        int vol;
        std::cout << "数値を入力し、[ENTER]キーを押してください。" << std::endl;
        for ( std::cin >> vol ; !std::cin ; std::cin >> vol){
            std::cin.clear();
            std::cin.ignore();
            std::cout << "入力が数値ではありません。" << std::endl;
        }
        var.SetValue(vol);
        pIsApi->setProperty(&propertyID, var);
        std::cout << "setProperty(" << var.GetValue() << ")" << std::endl;
    }

    // [8] 終了処理
    if(pIsApi){
        pIsApi->Stop();
        delete pIsApi;
    }
    return 0;
}

// [9] コールバック関数 1
void event_callback01(CIsComVariantList &list)
{
    std::cout << "LED ON!" << std::endl;
}

// [9] コールバック関数 2
void event_callback02(CIsComVariantList &list)
{
    std::cout << "LED OFF!" << std::endl;
}
```

## ソースファイル解説

ソースファイルの各部分を解説します。コメント部分の番号と対応しています。

### [1] ClsApi クラスのヘッダファイルのインクルード

IS-API のヘッダファイルをインクルードしています。

| | |
|---|---|
| インクルードファイル | CIsApi.h |
| | CIsComString.h |
| | CIsComVariant.h |
| | CIsComVariantList.h |

### [2] ClsApi クラスオブジェクトの作成

IS-API を使用するため、まず初めに ClsApi クラスオブジェクトを作成します。
IS-API はこのオブジェクトを通して使用します。

**■書式**

```
CIsApi(int hPort)
```

**■パラメータ**

| パラメータ | 説明 |
|---|---|
| int hPort | IS-API が使用するポート番号 |

### [3] IS-API の起動

IS-API は TCP/IP サーバーとして動作します。
待ち受けを開始し、IS-APP からの接続を待ちます。

**■書式**

```
bool Start()
```

**■戻り値**

true:成功
false:失敗

## [4] 動作状態確認

IS-API の動作状態を取得します。IS-API の各機能は動作中（本関数の戻り値が True 時）のみ使用することができます。（動作状態確認は必須ではありません）

**■書式**

```
bool IsRunning()
```

**■戻り値**

true:動作中
false:停止中

## [5] テスト通信

実際に IS-APP に対してテスト通信を行います。（テスト通信は必須ではありません）

**■書式**

```
bool Test()
```

**■戻り値**

true:成功
false:失敗

## [6] コールバック関数登録

コールバック関数を登録すると InfoSOSA に表示されたボタンが押された時※に、登録した関数を実行することができます。
※　InfoSOSA ビルダで、ボタンにアクション「イベントを上位に通知する」の設定が必要です。

**■書式**

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

**■パラメータ**

| パラメータ | 説明 |
|---|---|
| std::string* _eventID | 対象のイベント ID<br>※イベント ID は［所属 ID］.［部品/メモリ ID］.［イベント ID］の書式で設定します。詳細はリファレンスマニュアルを参照してください。 |
| EventCallbackFunc _func | 登録する関数 |

**■戻り値**

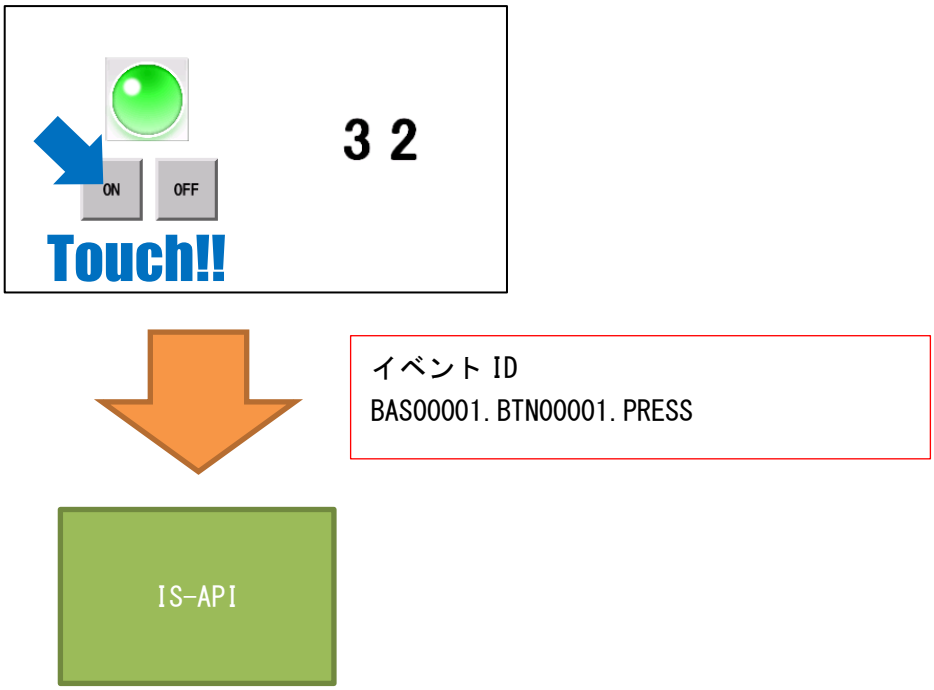
true:成功
false:失敗

コールバック関数の登録は「どのイベント発生時」に「どの関数を実行するか」を指定します。

InfoSOSA では、動作設定（アクション）を実行するタイミングを「イベント」と呼んでいます。例えば、ボタンを押した時には、そのボタンの「PRESS」イベントが発生し、ボタンを離した時には「RELEASE」イベントが発生します。

InfoSOSA ビルダでは、このイベントごとにアクションを設定することができ、ボタンを押した時にランプを点灯させて、離した時に消灯させるようなことが可能です。

イベントに上位通知アクション(上位へイベントを通知など)を設定すると、固有の ID が上位(IS-API)へ通知されます。




以下のようにボタンを押した時のイベント ID の文字列を作成し、登録用の関数に渡してください。

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

イベントの発生仕様、イベント ID の仕様については、別紙「InfoSOSA リファレンスマニュアル」を参照ください。

## [7] IS-APP のメモリの値を変更する

コンソールからの入力を待ち、入力された値を InfoSOSA のメモリにセットします。

**■書式**

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

**■パラメータ**

| パラメータ | 説明 |
|---|---|
| std::string* _propertyID | 変更する対象のプロパティ ID<br>※ID は［所属 ID］.［部品/メモリ ID］.［プロパティ ID］の書式で設定します。詳細は別紙「InfoSOSA リファレンスマニュアル」を参照ください。 |
| CIsComVariant& _var | セットする値、文字列<br>※文字列の文字コードは必ず UTF8 で設定してください。 |

**■戻り値**

true:成功
false:失敗

InfoSOSA のメモリを変更するためには、「どのプロパティ」を「どの値」に変更するかを指定します。

InfoSOSA の部品／メモリには、それぞれユニークな ID が設定されており、複数のプロパティを持っています。
ID は［所属 ID］.［部品/メモリ ID］.［プロパティ ID］の書式で設定します。詳細は別紙「InfoSOSA リファレンスマニュアル」を参照ください。

以下のように文字列を作成し、設定用の関数に渡してください。

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

| 項目 | 説明 |
|---|---|
| 所属 ID | @GLBMEM |
| 部品／メモリ ID | GM_VOL01 |
| プロパティ ID | VALUE |

値は、ClsComVariant 型で作成します。
ClsComVariant 型には以下のように SetValue メソッドで値を設定してください。

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

**■書式**

bool SetValue(int value)

**■パラメータ**

| パラメータ | 説明 |
|---|---|
| int value | 値 |

**■戻り値**

true:成功
false:失敗

## [8] 終了処理

IS-API の TCP/IP サーバーを停止し、IS-APP との接続を切断します。

**■書式**

bool Stop()

**■戻り値**

true:成功
false:失敗

## [9] コールバック関数

[6]コールバック関数登録で登録した関数です。InfoSOSA のボタンが押された時に、コンソールに LED の ON／OFF を表示します。

## ビルド

作成したソースファイルをビルドします。引き続きソフトウェア開発環境(Linux 仮想マシン)を操作します。

### 端末を起動する

ツールバー上の  をクリックして「端末」を起動します。

### ソースファイルを保存したディレクトリに移動

起動した「端末」に以下のようにコマンドを入力してください。

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

※保存ディレクトリを変更している場合は、変更先に移動してください。

### ビルド環境のセットアップ

環境変数などの設定を行います。端末の起動ごとに実行する必要があります。
型式ごとに対応スクリプトが変わります。
事前に開発環境にツールチェインのインストールが必要です。ツールチェインのインストール方法については、別紙「EM シリーズ ソフトウェア開発マニュアル」を参照ください。

| 型式 | 対応スクリプト |
|---|---|
| EMG7-***A8-****-**7 | IS-APP-A8 用スクリプト　※1 |
| EM8-***A7-****-**7 | IS-APP-A7 用スクリプト　※2 |
| EMG8-***A7-****-**7 | IS-APP-A7 用スクリプト　※2 |
| EMP-***A7-****-**7 | IS-APP-A7 用スクリプト　※2 |

※1　IS-APP-A8 用スクリプト

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

※2　IS-APP-A7 用スクリプト

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

ツールチェインのインストール先を/opt/poky/2.1.2 から変更している場合は、インストール先に合わせて変更してください。

### ビルドの実行

以下のコマンドでビルドを実行します。

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

| 項目 | 説明 |
|---|---|
| $CXX | 環境変数「$CXX」<br>環境変数はビルド環境のセットアップ時に設定されます。適切なコンパイラ、コンパイルオプションが指定されます。 |
| tutorial.cpp | ビルド対象のソースファイル |
| -L./ | ライブラリ検索パス<br>カレントディレクトリを指定しています。 |
| -lisapi | 使用するライブラリ（libisapi.so） |
| -o tutorial | 出力ファイル名 |

カレントディレクトリに以下の実行ファイルが生成されます。

| | |
|---|---|
| 実行ファイル | tutorial |

## 1.7.7　EM シリーズへ転送

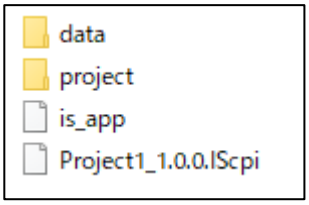
作成した各データを EM シリーズへ転送します。
事前に EM シリーズにインストールされているソフトウェアの更新が必要です。
「1.6 IS-APP/IS-API/IS-APP SETTING の更新」を参照ください。

### IS-APP 転送用データ作成

PC(Windows)で InfoSOSA ビルダを操作します。InfoSOSA ビルダで作成したプロジェクトは、以下の手順で転送用データに変換します。


「ダウンロード」メニューの「データ作成(USB/IS-APP)」から行います。


転送用データを作成します。「データ作成後に保存先フォルダを開く」をチェックした状態で「作成」ボタンを押してください。ダウンロードデータの保存場所を変える場合は、「参照」ボタンを押して保存先を変更してください。




保存先フォルダが開きます。

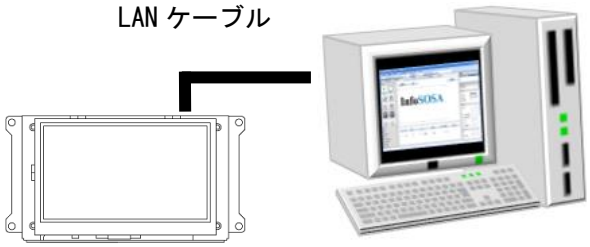
| フォルダ／ファイル | 説明 |
|---|---|
| data | 転送用に変換された画面データ |
| project | 変換前の画面データ（編集可） |
| is_app | IS-APP の実行ファイル |
| Project1_1.0.0.IScpi | 転送データの内容を記述したシステムファイル。ファイル名は、[プロジェクト名]_[ユーザバージョン].IScpi |

## 転送方法

### PC の操作

PC(Windows)を操作します。
EM シリーズには SambaServer が搭載されており、EM シリーズ本体のユーザフォルダに Windows からアクセス可能です。

転送にはネットワーク設定が必要です。
「1.5　EM シリーズ本体と PC との接続」の手順を行ってください。

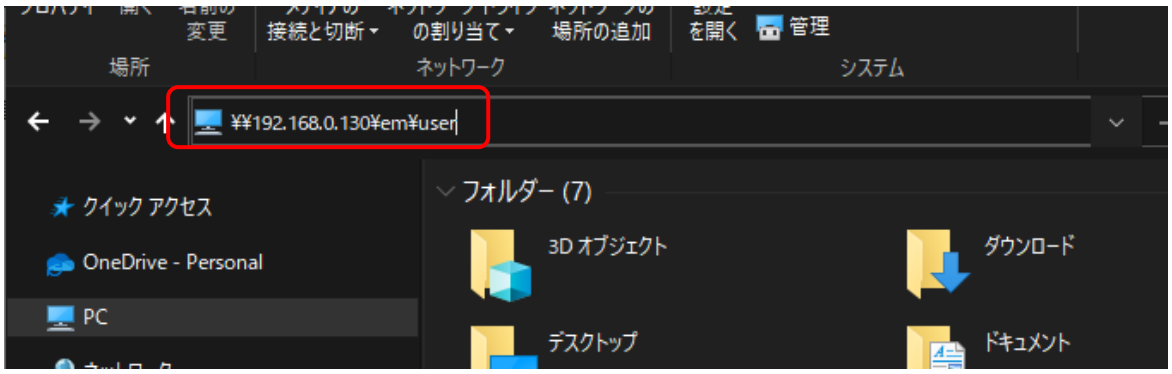



EM シリーズ本体と PC を LAN ケーブルで接続します。

エクスプローラーなどで、以下のアドレスにアクセスします。

| | |
|---|---|
| EM シリーズ ユーザフォルダパス | ¥¥192.168.0.130¥em¥user |

※IP アドレスは工場出荷時のデフォルト値です。



以下のユーザでログインします。

| | |
|---|---|
| ユーザ | root |
| パスワード | 無し |

EM シリーズのユーザフォルダにアクセスできますので、エクスプローラーなどで EM シリーズにファイルをコピーします。

■IS-APP

「IS-APP 転送用データ作成」で作成したデータをコピーしてください。

※下記以外のフォルダ／ファイルはコピー不要です。

| フォルダ／ファイル | 説明 |
|---|---|
| data | 転送用に変換された画面データ<br>※フォルダごとコピーしてください |

**【ご注意】**

**画面データを更新する場合は、必ず転送済みの画面データを一度フォルダごと削除してから転送してください。（フォルダ内に古いファイルがあると誤作動の原因になります）**

■IS-API

ソフトウェア開発環境(Linux 仮想マシン)から取り出してコピーしてください。

| フォルダ／ファイル | 説明 |
|---|---|
| tutorial | 「1.7.6 C++アプリケーションの作成」で作成した実行ファイル |

**【EM シリーズのユーザフォルダにアクセスできない場合】**

- LAN ケーブル用の IP アドレスが「192.168.10.*」の場合は通信できません。システム設定ツールより変更してください。
- お客様の PC 環境上に IP アドレスが同じ「192.168.0.130」になっている別のデバイスが存在していないかご確認ください。

## 1.7.8　IS-APP 単体テスト

IS-APP を起動してください。電源 ON で自動的に起動する設定も可能です。

### 起動方法(手動)

PC(Windows)を操作して、EM シリーズにコンソールを接続してください。コンソールの接続方法は、別紙「EM シリーズ ソフトウェア開発マニュアル」を参照ください。

データを転送したフォルダ（ユーザフォルダ）に移動します。

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

実行ファイル「is_app」を実行します。
IS-APP は、実行時のコマンドライン引数で、表示する画面データのフォルダや通信設定を行います。

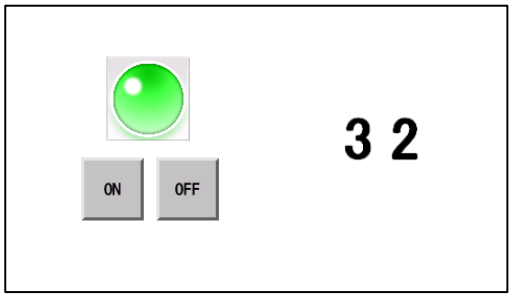
```
$ export DISPLAY=:0
$ is_app -r /mnt/user/data/ -a 51111
```

※1 行目はプログラム実行時の表示先を設定しています。
（環境変数「DISPLAY」にディスプレイ番号 0 を設定）
※画面データはフォルダを指定します。

| 引数 | -r, --rscdir |
|---|---|
| 書式 | -r 〈画面データフォルダのパス〉 |
| 説明 | 画面データのフォルダを指定します。 |

| 引数 | -a, --api |
|---|---|
| 書式 | -a 〈接続先ポート番号〉 |
| 説明 | 接続先に IS-API を指定します。 |

以下のような画面が表示されます。




その他のコマンドライン引数については、「2.1.7 コマンドライン引数」を参照ください。

## 起動方法(自動)

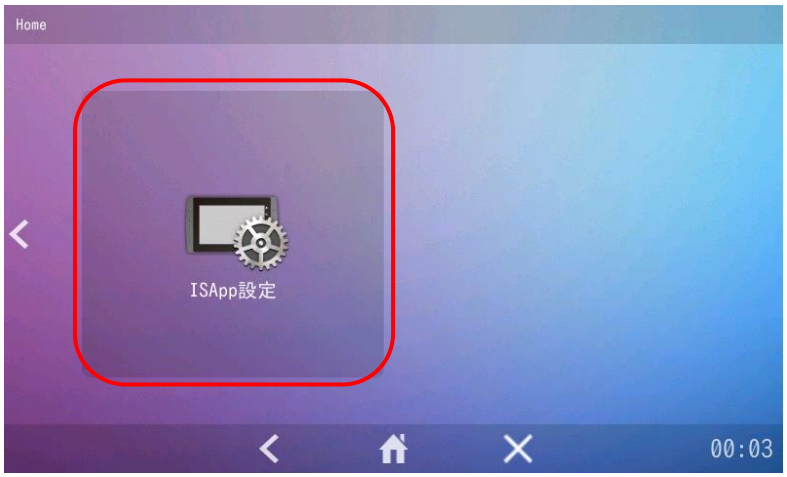
電源 ON で自動的に IS-APP を起動する方法を記載します。

### 1. 起動スクリプト作成

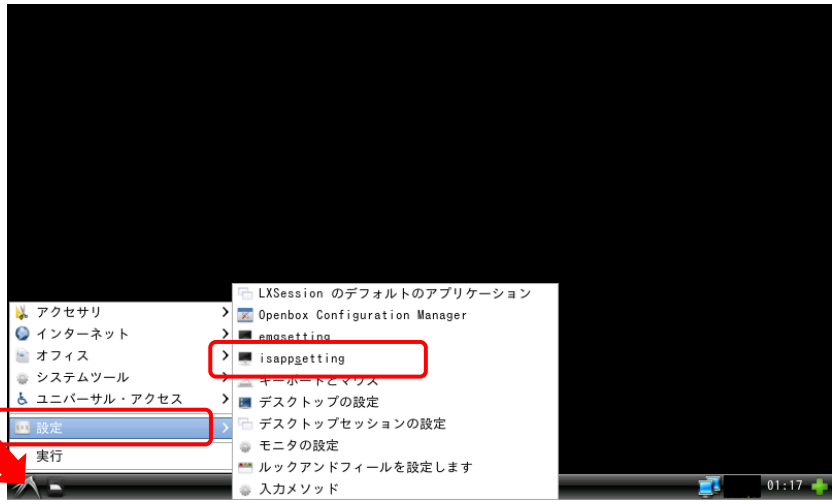
IS-APP は、通信設定などを起動時のコマンドライン引数として指定します。
ISAPP SETTING を使用すると GUI でコマンドライン引数の設定を行うことができます。
（コマンドライン引数が設定された状態の起動スクリプトを作成します）

ISAPP SETTING は EMG ランチャーまたは、スタートメニューから起動します。

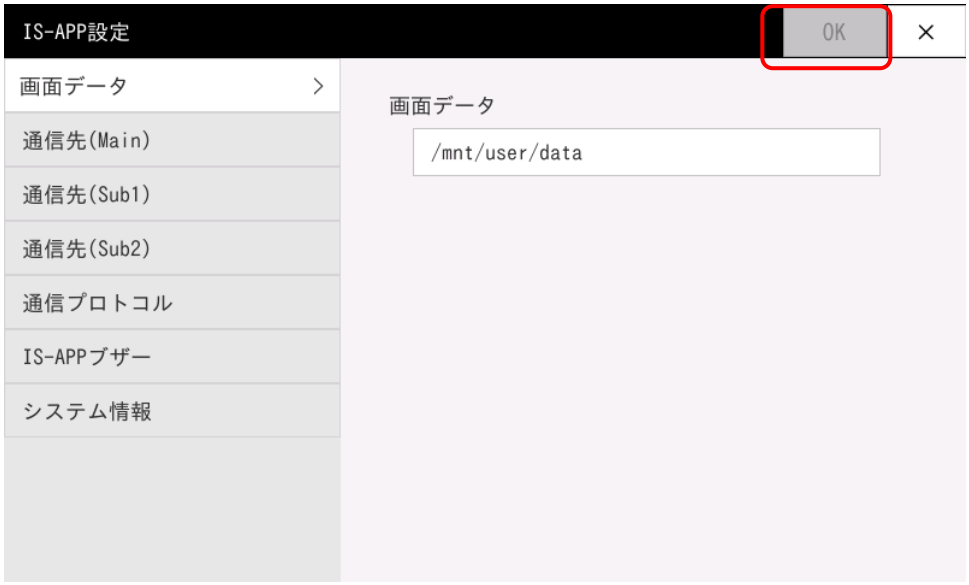
[EMG ランチャー] - [ISApp 設定]



[スタートメニュー] - [設定] - [isappsetting]




設定完了後、右上の OK ボタンをタッチすると起動スクリプトファイルが更新されます。
ここでは、デフォルト設定のまま×ボタンで閉じてください。




起動スクリプトは以下の場所に作成されます。

| /mnt/user/isapp_run.sh |
|---|

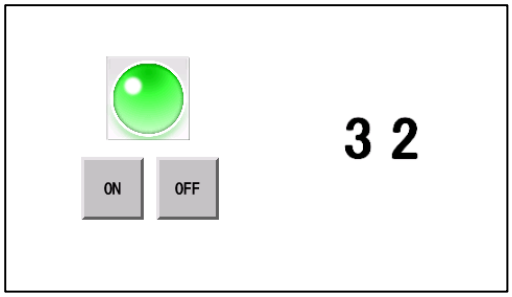
各設定については「2.3 ISAPP SETTING」を参照ください。

## 2．自動起動設定

電源 ON 時に自動的に起動スクリプトを実行して、IS-APP を起動する設定は、システム設定ツールから行えます。

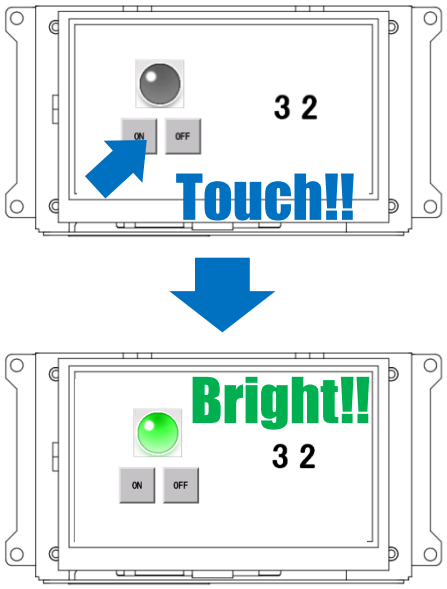

詳しくは、別紙「EM シリーズ ツールマニュアル」を参照ください。

電源を入れ直すと以下のような画面が表示されます。



## 動作確認

ON／OFF ボタンをタッチすると画面上のランプが点灯消灯します。
これは、通信は行なわずに InfoSOSA がアクションで自身（InfoSOSA 内）のメモリ「GM_LED01」の値を書き換えています。メモリ「GM_LED01」の値が変わることでリンク設定が行われているランプが点灯します。



## 1.7.9　結合テスト

C++アプリケーションを起動し、IS-APP と接続します。

### 起動方法

IS-APP を起動したコンソールと別のコンソールを起動して接続します。

実行ファイル「tutorial」を実行します。

例：

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

※1 行目はプログラム実行時の表示先を設定しています。
（環境変数「DISPLAY」にディスプレイ番号 0 を設定）
※2 行目はユーザフォルダに移動しています。
※3 行目は実行ファイル「tutorial」に実行権限を設定しています。

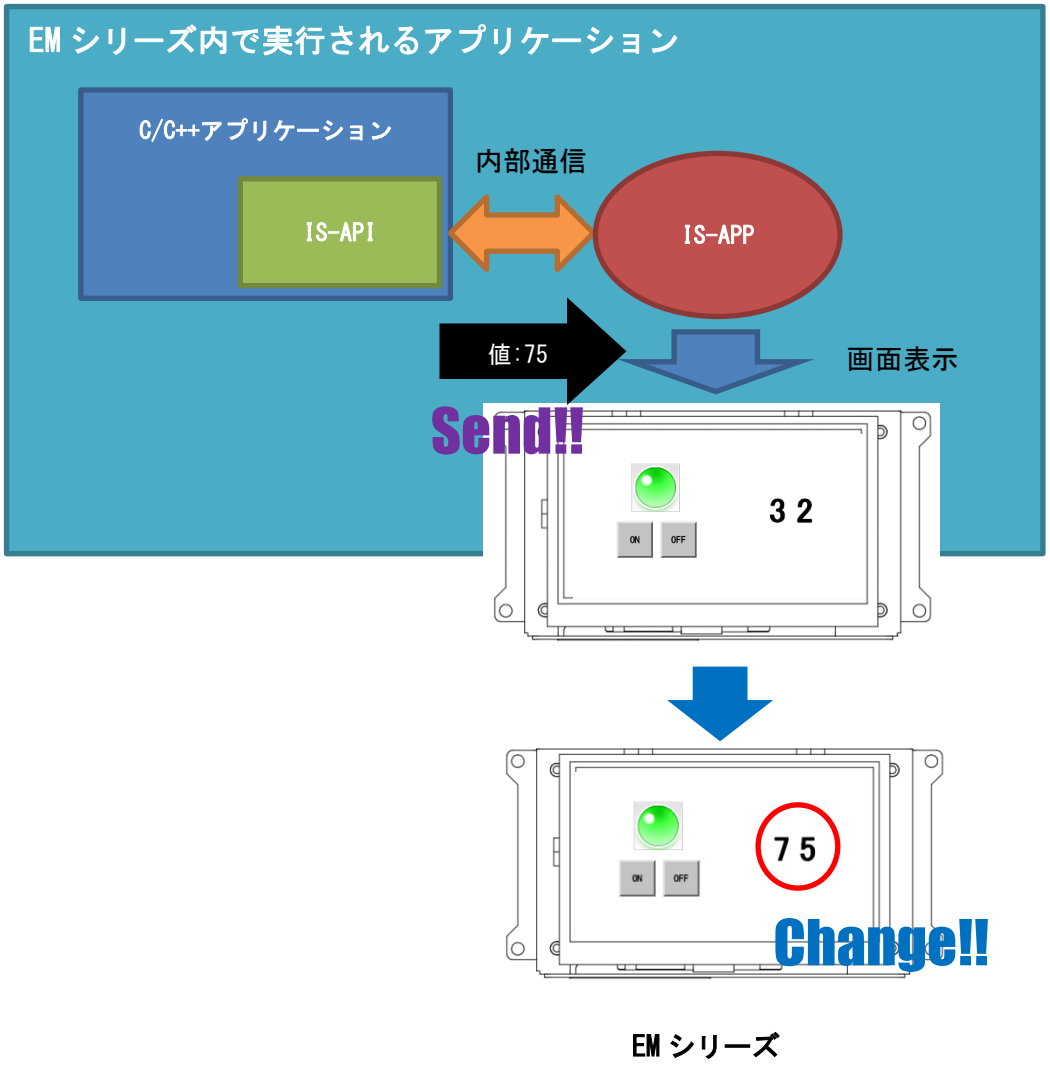
IS-APP と接続が成功するとコンソール画面上に以下のように表示されます。

```
Connecting...OK
数値を入力し、[ENTER]キーを押してください。
```

## 動作確認

### コンソールに数字を入力すると画面上の数字が変化する

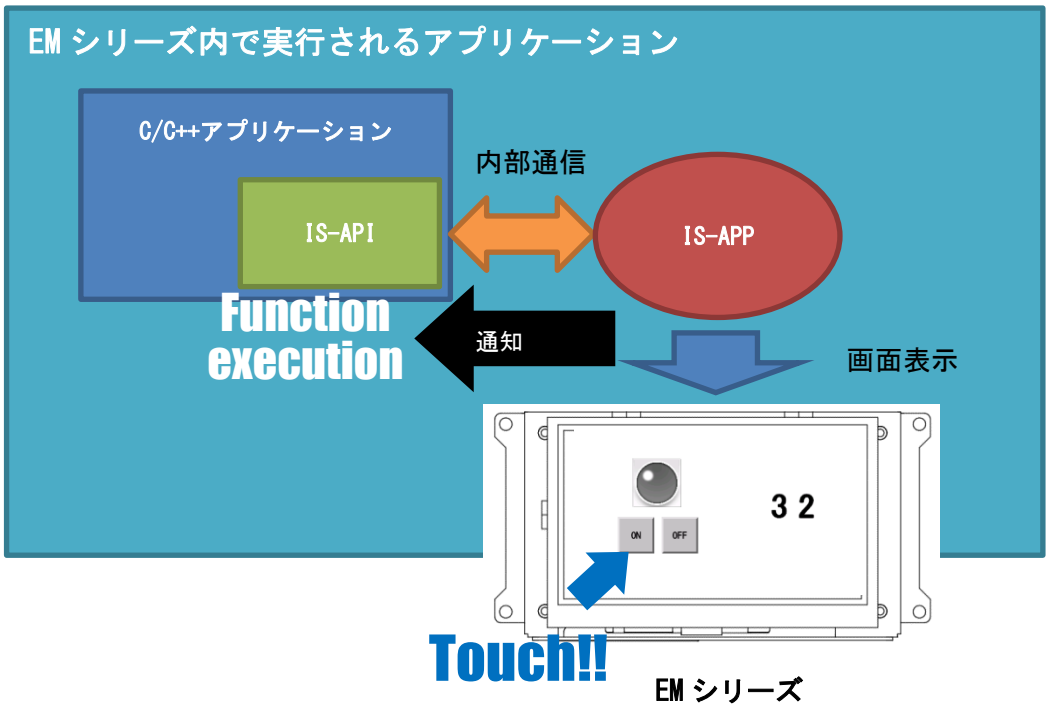
コンソールに数字を入力して ENTER キーを押すと液晶画面上（IS-APP）の数字が変わります。これは、C++アプリケーションが IS-API を使用して IS-APP のメモリ「GM_VOLO1」の値を書換えています。メモリ「GM_VOLO1」の値が変わることでリンク設定が行われている数字表示部品が変換します。




EM シリーズ

### 画面上のボタンを押すとコンソールに状態が表示される

液晶画面上（IS-APP）の ON／OFF ボタンをタッチするとコンソールに状態が表示されます。
これは、ボタンを押した時 IS-APP が状態を送信し、IS-API が登録された関数を実行しています。
この機能により、IS-APP のボタンを押したときに任意の処理を実行することができます。



## 1.7.10　その他の機能

以上でチュートリアルは完了になります。
チュートリアルで使用していない機能については以下を参照ください。

### InfoSOSA ビルダ

InfoSOSA ビルダは、他にも部品やアクションの種類があり、プログラミングを行なわずとも簡単な条件分岐処理等は設定することが可能です。
詳しくは、別紙「InfoSOSA リファレンスマニュアル」「InfoSOSA ビルダ操作マニュアル」を参照ください。

### IS-APP

IS-APP は、実行時のコマンドライン引数で以下の設定を行うことができます。
・ウィンドウ表示
・通信設定（外部機器と通信）
・ブザー機能
・サウンド機能
詳しくは、「2.1　IS-APP 詳細仕様」を参照ください。

### IS-API

IS-API は、他にも表示画面を切り替えたり、複数のメモリをまとめて書き換える機能があります。
詳しくは、「2.2　IS-API 詳細仕様」を参照ください。

### ISAPP SETTING

EM シリーズには、IS-APP をコンソールを使わずに自動起動するように設定が行える補助ツール「ISAPP SETTING」がインストールされています。
C/C++アプリケーションを使わない場合は、このツールを使用することでコンソール接続を行なわずに IS-APP を起動することが可能です。
詳しくは、「2.3　ISAPP SETTING」を参照ください。

# 2章　リファレンス

**章目次**

# 2.1　IS-APP詳細仕様

## 2.1.1　概要

IS-APP は、InfoSOSA ビルダで作成した画面を表示するアプリケーションです。
InfoSOSA ビルダで画面データを作成し、EM シリーズに転送してください。
（InfoSOSA ビルダは Windows®上で動作する作画ソフトウェアです）
実行時にコマンドライン引数で、表示する画面データが格納されたフォルダ、使用するデバイスなどを指定します。
具体的な使用方法は、「1.7　IS-APP を使用した HMI の開発方法」を参照ください。

## 2.1.2　実行ファイル名

| 実行ファイル |
| --- |
| is_app |

EM シリーズ本体にインストールが必要

インストール方法は、「1.6 IS-APP/IS-API/IS-APP SETTING の更新」を参照ください。

## 2.1.3　実行可能数

同時に実行可能な IS-APP は１つです。
２つ以上のプロジェクトを同時に表示することはできません。

## 2.1.4　対応ハードウェア

本アプリケーションは、以下のハードウェア以外では動作しません。

| 型式 |
|---|
| EMG7-***A8-****-**7 |
| EM8-***A7-****-**7 |
| EMG8-***A7-****-**7 |
| EMP-***A7-****-**7 |

※末尾が 0*7 の型式は除く

## 2.1.5　使用可能システムフォント数

製品型式により、システムフォントの使用可能な種類数と簡体字フォントの使用可否が異なります。実行する画面データに使用されたフォントの種類数が超過していたり、簡体字フォントが使えない製品で使用していたりする場合は実行できません。

| 型式 | 選択可能なフォント種類数 | 簡体字使用可否 |
|---|---|---|
| EMG7-***A8-****-1*7 | 5 | ○ |
| EM8-***A7-****-1*7 | 5 | ○ |
| EMG8-***A7-****-1*7 | 5 | ○ |
| EM8-***A7-****-2*7 | 1 | × |
| EMG8-***A7-****-2*7 | 1 | × |
| EMP-***A7-****-2*7 | 1 | × |

## 2.1.6　起動モード

### フルスクリーンモード

全画面に表示するモードです。ウィンドウモード指定オプション(-w)を指定しない、またはウィンドウサイズが液晶解像度と同じ場合はこのモードで起動します。タスクバーや他のウィンドウより前面に表示されます。
起動中に他の全画面に表示するアプリケーションが起動した場合は、後に起動したアプリケーションが全面に表示されます。

### ウィンドウモード

ウィンドウで表示されるモードです。ウィンドウモード指定オプション(-w)を付けて起動するとこのモードで起動します。タスクバーが手前に表示されます。表示順序を指定することで常に他のアプリケーションの手前や背後に表示することが可能です。

※ウィンドウモード指定オプション(-w)を付けた場合でもウィンドウサイズと液晶解像度が同じ場合はフルスクリーンモードになります。

## 2.1.7　コマンドライン引数

### 画面データフォルダ

表示する画面データのフォルダを指定します。

#### 引数名

-r　または　--rscdir

#### 書式

-r <PATH>

| パラメータ | 内容 |
|---|---|
| <PATH> | 画面データへのパスを指定します。 |

例：

```
is_app -r /mnt/user/data
```

#### デフォルト値

本引数は必須です。省略することはできません。

## ウィンドウ

IS-APP のウィンドウサイズや位置を指定します。

### 引数名

-w　または　--window

### 書式

-w 〈X 座標〉〈Y 座標〉［横サイズ(省略可)］［縦サイズ(省略可)］〈タイトルバー〉〈表示順序〉

| パラメータ | 内容 |
|---|---|
| 〈X 座標〉〈Y 座標〉 | 起動時のウィンドウ位置を指定します。<br>0～液晶解像度（横）<br>0～液晶解像度（縦） |
| 〈横サイズ〉<br>〈縦サイズ〉 | 起動時のウィンドウサイズを指定します。<br>省略すると、自動（ビルダで指定したベース画面サイズ）になります。<br>1～液晶解像度（横）<br>1～液晶解像度（縦） |
| 〈タイトルバー〉 | yes： タイトルバーを表示する　※移動、リサイズ可能<br>no： タイトルバーを表示しない　※移動、リサイズ不可 |
| 〈表示順序〉 | none： 指定しない<br>bottom： 最背面に表示する<br>top: 最前面に表示する |

例：

```
is_app -r /mnt/user/data -w 100 100 640 320 yes top
```

### デフォルト値

省略した場合は以下の設定になります。

| パラメータ | 内容 |
|---|---|
| 〈X 座標〉〈Y 座標〉 | 0,0 |
| 〈横サイズ〉<br>〈縦サイズ〉 | 自動（ビルダで指定したベース画面サイズ） |
| 〈タイトルバー〉 | no： タイトルバーを表示しない　※移動、リサイズ不可 |
| 〈表示順序〉 | none： 指定しない |

## 通信設定

IS-APP の通信設定を指定します。

通信インタフェースは、最大３つまで指定することができます。
指定可能な組み合わせは以下になります。

| Main 引数 | Sub 引数 | Sub 引数 |
|---|---|---|
| シリアル | 指定無し | 指定無し |
| LAN | 指定無し | 指定無し |
| IS-API | 指定無し | 指定無し |
| シリアル | LAN | 指定無し |
| シリアル | IS-API | 指定無し |
| LAN | シリアル | 指定無し |
| IS-API | シリアル | 指定無し |
| シリアル | シリアル | 指定無し |
| シリアル | シリアル | LAN |
| シリアル | シリアル | IS-API |
| LAN | シリアル | シリアル |
| IS-API | シリアル | シリアル |

**【ご注意】**

・　Sub 引数は順不同です。

・　LAN、IS-API はどちらかしか指定できません。LAN（IS-API 含む）で複数接続する場合は、LAN の接続プロトコルに TCP/IP サーバーを指定してください。

・　引数に不足があった場合、不足項目はデフォルト値が設定されます。

・　シリアルインタフェースを指定する場合は、同じデバイスファイルは指定できません。

・　アクション「イベントを上位へ通知する」「値を上位へ通知する」実行時のイベント通知先は、Main 引数に指定したインタフェースになります。Sub 引数には通知されません。
例：シリアル、IS-API と通信する場合（イベントはシリアルのみへ通知）

```
is_app -r /mnt/user/data -s /dev/com1 rs232c 115200 none none -a 51111
```

・　起動伝文は全てのインタフェースに送信されます。※TCP/IP サーバーを除く
［起動伝文(シリアル)］
{STX}00s00000008000010000003DC{ETX}
［起動伝文(LAN)］
s000000800010000

・　IS-API と接続した場合、文字コード設定(SIO3)が自動的に「Unicode （UTF-16LE）」に設定されます。IS-API をご使用される場合は、「Shift JIS」に設定しないでください。IS-API が正常に動作しなくなります。

・　文字コード設定は、全ての通信インタフェースに適用されます。IS-API を使用される場合は、シリアルや LAN 通信も「Unicode （UTF-16LE）」で通信してください。

## 引数名

シリアル：-s　または　--sio
LAN：-l　または　--lan
IS-API：-a　または　--api　※デフォルト

## 書式(シリアル)

RS232C の場合：※デフォルト
-s〈デバイスファイル〉rs232c〈通信速度〉〈パリティ〉〈フロー制御〉

RS422 の場合：
-s〈デバイスファイル〉rs422〈通信速度〉〈パリティ〉

RS485 の場合：
-s〈デバイスファイル〉rs485〈通信速度〉〈パリティ〉〈アドレス〉〈再送回数〉〈再送間隔〉〈送信権タイムアウト〉

| パラメータ | 内容 |
|---|---|
| 〈デバイスファイル〉 | シリアルポートのデバイスファイルを指定します。<br>デフォルト：SIO1 のデバイスファイルが自動設定されます<br>SIO1：/dev/com1<br>SIO2：/dev/com2 |
| 〈通信速度〉 | 通信速度を指定します。<br>4800：4800bps<br>9600：9600bps<br>19200：19200bps<br>38400：38400bps<br>57600：57600bps<br>115200：115200bps　※デフォルト |

| パラメータ | 内容 |
|---|---|
| <パリティ> | パリティを指定します。<br>none： パリティ無し　※デフォルト<br>odd： 奇数パリティ<br>even： 偶数パリティ |
| <フロー制御> | RS232C 通信時のフロー制御を指定します。<br>none： フロー制御無し　※デフォルト |
| <アドレス> | RS485 通信時の自局アドレスを指定します。<br>1～31<br>デフォルト：1 |
| <再送回数> | RS485 通信時、衝突で送信が失敗した場合の再送回数を指定します。<br>0～10<br>デフォルト：3 |
| <再送間隔> | RS485 通信時、衝突で送信が失敗した場合の再送の間隔を指定します。<br>0～1000 (単位 10ms)<br>デフォルト：100 |
| <送信権タイムアウト> | RS485 通信時、他の機器が送信している場合の最大待機時間を指定します。<br>0～3000 (単位 10ms)<br>デフォルト：500 |

例：

RS232C の場合：

```
is_app -r /mnt/user/data -s /dev/com1 rs232c 115200 none none
```

RS422 の場合

```
is_app -r /mnt/user/data -s /dev/com2 rs422 115200 none
```

RS485 の場合

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

### 書式(LAN)

UDP/IP の場合：※デフォルト

```
-l udp <通信先アドレス><通信先ポート>
```

TCP/IP クライアントの場合：

```
-l tcp <通信先アドレス><通信先ポート><接続間隔>
```

TCP/IP サーバーの場合：

```
-l tcps <イベント通知用ポート><通常ポート>
```

| パラメータ | 内容 |
|---|---|
| <通信先アドレス> | UDP/IP または TCP/IP クライアントの場合の通信先の IP アドレスを指定します。<br>デフォルト：192.168.0.100 |
| <通信先ポート> | UDP/IP または TCP/IP クライアントの場合の通信先のポート番号を指定します。<br>0～65535<br>デフォルト値：51111 |
| <接続間隔> | TCP/IP クライアントの場合の通信先 TCP/IP サーバーへの接続間隔を指定します。<br>0～99 (単位 秒)<br>デフォルト：5 |
| <イベント通知用ポート> | TCP/IP サーバーの場合の TCP/IP クライアントからの接続を受け付けるポート番号を指定します。<br>このポートに接続可能な TCP/IP クライアントは最大「1」です。<br>このポートに接続したクライアントにのみイベントを通知します※。<br>0～65535<br>デフォルト値：51111<br>※LAN インタフェースを Sub 引数に設定している場合は、本ポートにも通知は行われません。 |
| <通常ポート> | TCP/IP サーバーの場合の TCP/IP クライアントからの接続を受け付けるポート番号を指定します。<br>このポートに接続可能な TCP/IP クライアントは最大「3」です。<br>このポートに接続したクライアントにはイベントの通知は行いません。<br>0～65535<br>デフォルト値：51115 |

例：

**UDP/IP の場合**：

```
is_app -r /mnt/user/data -l udp 192.168.0.100 51111
```

**TCP/IP(client)の場合**

```
is_app -r /mnt/user/data -l tcp 192.168.0.100 51111 5
```

**TCP/IP(server)の場合**

```
is_app -r /mnt/user/data -l tcps 51111 51115
```

### 書式(IS-API)

-a〈通信先ポート〉

| パラメータ | 内容 |
|---|---|
| 〈通信先ポート〉 | 通信先のポート番号を指定します。<br>0～65535<br>デフォルト値：51111 |

例：

```
is_app -r /mnt/user/data -a 51111
```

### デフォルト値

全て省略した場合は以下の設定になります。

| Main 引数 | Sub 引数 | Sub 引数 |
|---|---|---|
| IS-API | 指定無し | 指定無し |

Main 引数

| パラメータ | 内容 |
|---|---|
| 通信タイプ | IS-API |
| 通信先ポート | 51111 |

## 通信プロトコル

IS-APP の通信プロトコルを指定します。
本設定は、通信設定がシリアルまたは LAN の時のみ有効です。
引数に不足があった場合、不足項目はデフォルト値が設定されます。

**【ご注意】**

- 本引数の設定は全ての通信先に適用されます。
- IS-API と接続する場合は、本引数は使用できません。「InfoSOSA 標準プロトコル／即時通知」設定になります。

### 引数名

-p　または　--protocol

### 書式

InfoSOSA 標準プロトコルの場合：※デフォルト
-p infososa〈通知方法〉

通常プロトコルの場合：
-p standard〈通知方法〉〈監視時間〉〈再送回数〉

| パラメータ | 内容 |
|---|---|
| 〈通知方法〉 | 通知方法を指定します。<br>immediate： 即時通知　※デフォルト<br>polling： 上位から要求 |
| 〈監視時間〉 | 通常プロトコルの再送までの時間を指定します。<br>1～5(秒)<br>デフォルト値：1 |
| 〈再送回数〉 | 通常プロトコルの再送回数を指定します。<br>0～3<br>デフォルト値：3 |

例：
is_app -r /mnt/user/data -s -p standard immediate 3 1

### デフォルト値

省略した場合は以下の設定になります。

| パラメータ | 内容 |
|---|---|
| 通信プロトコル | InfoSOSA プロトコル |
| 通知方法 | 即時通知 |

## ブザー

IS-APP のブザー機能を有効にします。
IS-APP でブザーを使用する場合は、あらかじめシステムタッチ音を無効にしてください。
本引数でブザー機能を有効にしていない場合は、InfoSOSA ビルダで設定したタッチ音やブザー音は無効になります。

### 引数名

-b　または　--buzzer

### 書式

-b〈デバイス〉

| パラメータ | 内容 |
|---|---|
| <デバイスファイル> | ブザーのデバイスファイルを指定します。<br>デフォルト：ブザーデバイスファイル(/dev/buzzer)が自動設定されます。<br>※通常は指定する必要はありません。 |

例：

```
is_app -r /mnt/user/data -b
```

### デフォルト値

省略した場合は以下の設定になります。

| パラメータ | 内容 |
|---|---|
| 有効／無効 | 無効 |

## サウンド

IS-APP のサウンドの出力先を変更します。

### 引数名

-o　または　--sound

### 書式

-o〈デバイス名〉〈ミキサー名〉〈ボリューム名〉

| パラメータ | 内容 |
|---|---|
| 〈デバイス名〉 | サウンド出力に使用するデバイス名を指定します。 |
| 〈ミキサー名〉 | サウンド出力に使用するミキサー名を指定します。 |
| 〈ボリューム名〉 | サウンド出力に使用するボリューム名を指定します。 |

例：

```
is_app -r /mnt/user/data -o default default PCM
```

### デフォルト値

省略した場合は以下の設定になります。
IS-APP 起動時にデバイスが存在しない場合は無効になります。

| パラメータ | 内容 |
|---|---|
| 有効／無効 | 有効 |
| 〈デバイス名〉 | default |
| 〈ミキサー名〉 | default |
| 〈ボリューム名〉 | PCM |

## バージョン表示

コンソールに IS-APP のバージョンを表示します。
本引数を指定した場合は、IS-APP の起動は行いません。

### 引数名

-v　または　--version

### 書式

-v

例：
is_app -v

## ヘルプ表示

コンソールに IS-APP の引数のヘルプを表示します。
本引数を指定した場合は、IS-APP の起動は行いません。

### 引数名

-h　または　--help

### 書式

-h

例：
is_app -h

# 2.2　IS-API詳細仕様

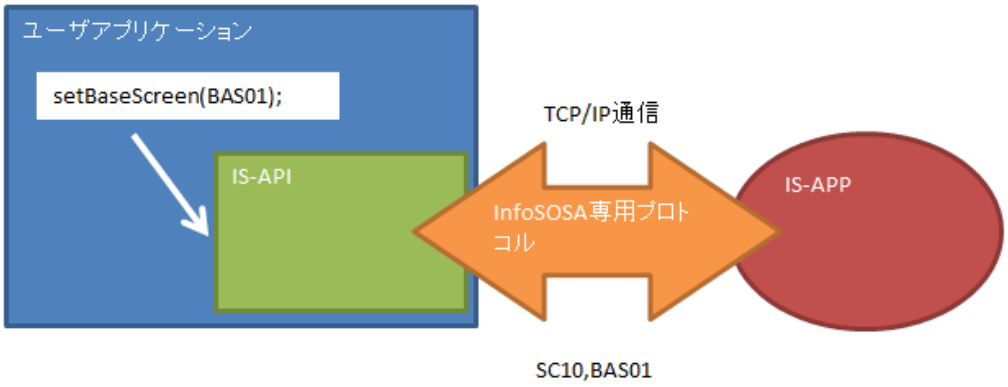
## 2.2.1　概要

IS-APP は、上位機器と InfoSOSA の専用プロトコルで通信を行います。
本 API はこの InfoSOSA の専用プロトコルによる通信部分を担当し、C/C++アプリケーションから簡単に IS-APP の表示画面を切り替えたり、メモリの取得などを行ったりすることが可能になります。

IS-APP と IS-API は、同一機器内の TCP/IP 通信を使用して接続されます。
IS-API を使用するためには、IS-APP の通信設定を IS-API にして起動する必要があります。
IS-APP の通信設定については、「2.1.7 コマンドライン引数　通信設定」を参照ください。

各機能は、それぞれ InfoSOSA の上位通信コマンドに対応しています。上位通信コマンドの詳細仕様については、別紙「InfoSOSA リファレンスマニュアル」を参照ください。

具体的な使用方法は、「1.7　IS-APP を使用した HMI の開発方法」を参照ください。



## 2.2.2　ライブラリ／ヘッダファイル

IS-API を使用した C/C++アプリケーションを作成するためには、ソフトウェア開発環境へライブラリファイルとヘッダファイル、EM シリーズ本体へライブラリファイルのインストールが必要になります。

| ライブラリファイル |
| --- |
| libisapi.so |

EM シリーズ本体へのインストールと開発環境にコピーが必要

| ヘッダファイル |
| --- |
| CIsApi.h |
| CIsComString.h |
| CIsComVariant.h |
| CIsComVariantList.h |

開発環境にコピーが必要

ライブラリファイル／ヘッダファイルは、InfoSOSA 開発キットデータ内の「software¥IS-API」フォルダに収録しています。

ライブラリファイルは、型式によりファイルが異なります。
対応フォルダ内のファイルをご使用ください。

| 型式 | 対応フォルダ |
| --- | --- |
| EMG7-***A8-****-**7 | IS-APP-A8 |
| EM8-***A7-****-**7 | IS-APP-A7 |
| EMG8-***A7-****-**7 | IS-APP-A7 |
| EMP-***A7-****-**7 | IS-APP-A7 |

EM シリーズ本体へのインストール方法は、「1.6 IS-APP/IS-API/IS-APP SETTING の更新」を参照ください。

## 2.2.3　同時接続可能数

１つの IS-API が接続可能な IS-APP は１つになります。

## 2.2.4　文字コード

API の引数に設定する文字列の文字コードは UTF8 になります。

## 2.2.5　リアルタイムシグナル

IS-API は以下のリアルタイムシグナルを使用しています。

| 定義 | ID | 動作 |
|---|---|---|
| SIGRTMIN | 34 | IS-API の各スレッドの停止 |

## 2.2.6　API 一覧

■CIsApi クラス

| 公開機能 | 内容 |
|---|---|
| CIsApi(int hPort) | IS-APP との接続用オブジェクトを作成します。<br>本クラスは、IS-API が TCP/IP サーバーになります。 |
| CIsApi_c(int hPort, unsigned int startTimeout) | IS-APP との接続用オブジェクトを作成します。<br>本クラスは､IS-API が TCP/IP クライアントになります。 |
| bool Start() | IS-API を起動し、IS-APP と通信を確立させます。 |
| bool Stop() | IS-API を停止し、IS-APP との通信を切断します。 |
| bool IsRunning() | IS-API の動作状態を取得します。 |
| bool Test() | IS-APP にテストアクセスを行い正しく動作しているかを確認します。 |
| int getApiVersion() | IS-API のバージョンを取得します。 |
| bool getInfososaVersion(std::string* _isVer, std::string* _userVer) | 接続中の IS-APP のバージョンを取得します。 |
| bool setBaseScreen(std::string* _screenID) | IS-APP の表示画面を切り替えます。 |
| bool getBaseScreen(std::string* _screenID) | IS-APP の現在表示中の画面 ID を取得します。 |
| bool showPopupScreenA(std::string* _screenID, int _x, int _y) | IS-APP の指定した座標にポップアップ画面 A を表示します。 |
| bool showPopupScreenB(std::string* _screenID, int _x, int _y) | IS-APP の指定した座標にポップアップ画面 B を表示します。 |
| bool hidePopupScreenA() | IS-APP のポップアップ画面 A を閉じます。 |
| bool hidePopupScreenB() | IS-APP のポップアップ画面 B を閉じます。 |
| bool getPopupScreen(int* _a_state, int* _b_state) | IS-APP のポップアップ画面の表示状態を取得します。 |
| bool setBuzzer(int _freq, int _msec) | 指定した周波数、時間でブザーを鳴らします。 |
| bool getBuzzer(int* _state) | IS-APP のブザーの鳴動状態を取得します。 |
| bool setProperty(std::string* _propertyID, CIsComVariant& _var) | IS-APP の部品のプロパティ設定を行います。 |
| bool getProperty(std::string* _propertyID, CIsComVariant& _var) | IS-APP の部品のプロパティを取得します。 |

| 公開機能 | 内容 |
|---|---|
| bool setGroupMemory(std::string* _groupID, CIsComVariantList& _varList) | IS-APP のグローバルメモリグループに値を設定します。 |
| bool getGroupMemory(std::string* _groupID, CIsComVariantList& _varList) | IS-APP のグローバルメモリグループの値を取得します。 |
| bool callSubRoutine(std::string* _subroutineID) | IS-APP に設定されている特定のサブルーチンを実行します。 |
| bool registerCallback(std::string* _eventID, EventCallbackFunc _func) | IS-APP の画面に表示されたボタンを押した時などに実行するコールバック関数を登録します。 |
| bool unregisterCallback(std::string* _eventID) | 登録したコールバック関数を解除します。 |

■CIsComVariant クラス

| 公開機能 | 内容 |
|---|---|
| bool SetValue(int value) | CIsComVariant クラスに値を設定します。 |
| int GetValue() | CIsComVariant クラスの値を取得します。 |

■CIsComString クラス

| 公開機能 | 内容 |
|---|---|
| bool SetString(int bytes, char* str) | CIsComString クラスに文字列を設定します。 |
| char* GetStringBuffer() | CIsComString クラスに設定された文字列へのポインタを取得します。 |
| int GetStringLength() | CIsComString クラスに設定された文字列のバイト数を取得します。 |

■CIsComVariantList クラス

| 公開機能 | 内容 |
|---|---|
| bool Add(CIsComVariant& elem) | CIsComVariantList クラスの末尾の要素に指定した CIsComVariant クラスを追加します。 |
| CIsComVariant* GetElem(int index) | CIsComVariantList クラスから指定した要素を取得します。 |
| int Size() | CIsComVariantList クラスに登録されている要素数を取得します。 |
| bool Clear() | CIsComVariantList の全ての登録を削除します。 |
| CIsComVariant* ReplaceElem(int index, CIsComVariant& elem) | CIsComVariantList クラスの指定した位置に指定した CIsComVariant クラスを設定します。 |

## 2.2.7　ClsApi/ClsApi_c

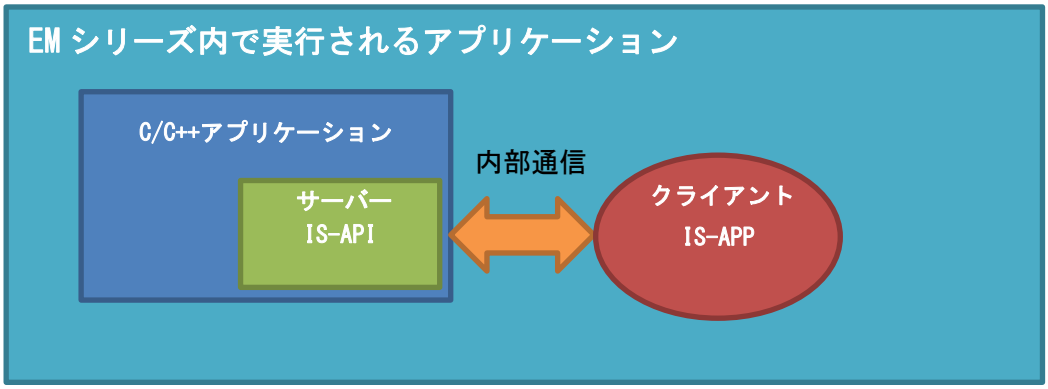
IS-APP へは本クラスを通じてアクセスします。
※本クラスのメソッドはスレッドセーフではございません。１つのスレッドから実行してください。

IS-APP 側が TCP/IP クライアントの場合は、「ClsApi」クラスを使用します。
IS-APP 側が TCP/IP サーバーの場合は、「ClsApi_c」クラスを使用します。

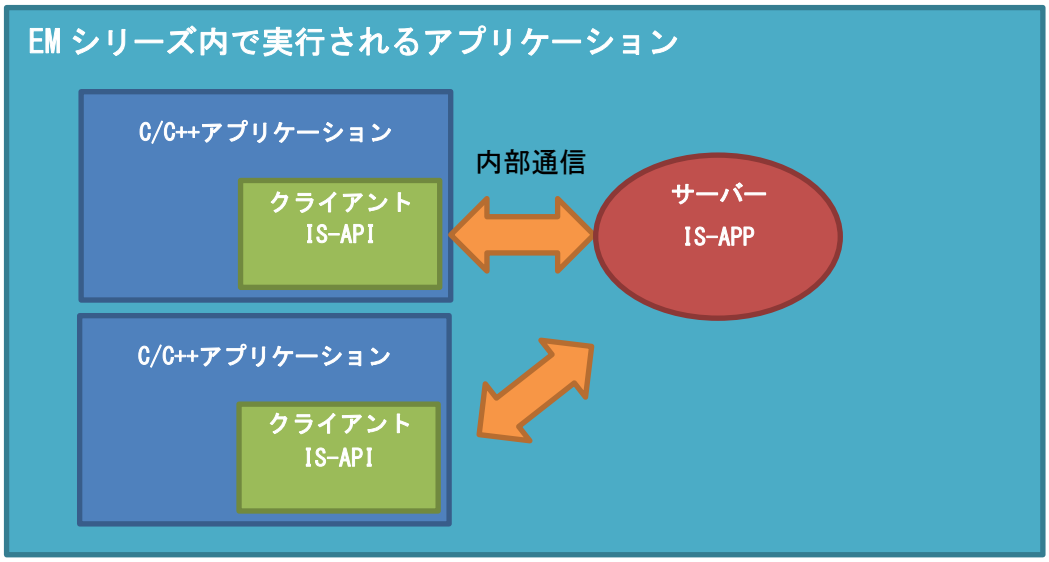
【対応表】

| IS-APP 引数 | IS-API クラス |
|---|---|
| -a | ClsApi |
| -l tcps | ClsApi_c |

IS-APP 側が TCP/IP クライアントの場合は、IS-APP に接続可能な IS-API は１つになります。




IS-APP 側を TCP/IP サーバーにすると、複数の IS-API を接続することが可能です。



## ヘッダファイル

CIsApi.h

## 公開機能

### コンストラクタ

ポート番号、タイムアウト時間を指定してオブジェクトを作成します。

**■書式**

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

**■パラメータ**

| パラメータ | 説明 |
|---|---|
| int hPort | CIsApi：IS-API が使用するポート番号<br>CIsApi_c：IS-API が接続するポート番号<br>※デフォルトは 51111 です。 |
| unsigned int<br>startTimeout | CIsApi：<br>指定できません。Start()を実行後は、IS-APP と接続完了するまで待ち受けを行います。<br>CIsApi_c：<br>Start()を実行した時に IS-APP に対して接続を行う場合のタイムアウト時間を設定します。単位は秒、値範囲は 0～300 です。0 の場合は一度接続を行い失敗すると直ちにタイムアウトします。※デフォルトは 15 秒です。 |

**■使用例(CIsApi)**

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

**■使用例(CIsApi_c)**

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

## Start

IS-API を起動し、IS-APP と通信を確立させます。

**■書式**

```
bool Start()
```

**■パラメータ**

なし

**■戻り値**

true:成功
false:失敗

**■使用例**

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

## IsRunning

IS-API の動作状態を取得します。IS-API の各機能は動作中（本関数の戻り値が True 時）のみ使用することができます。

**■書式**

```
bool IsRunning()
```

**■パラメータ**

なし

**■戻り値**

true:動作中
false:停止中

**■使用例**

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

## Test

IS-APP に対してテストアクセスを行います。

**■書式**

```
bool Test()
```

**■パラメータ**

なし

**■戻り値**

true:成功
false:失敗

**■使用例**

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

## Stop

IS-APP との接続を切断し、IS-API を終了します。

**■書式**

```
bool Stop()
```

**■パラメータ**

なし

**■戻り値**

true:成功
false:失敗

**■使用例**

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

## getApiVersion

IS-API のバージョンを取得します。

**■書式**

int getApiVersion()

**■パラメータ**

なし

**■戻り値**

IS-API バージョン(XYY)
X=メジャーバージョン
YY=マイナーバージョン

例：
1.0.0 の場合は、100 が返ります。

**■使用例**

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

## getInfososaVersion

接続中の IS-APP のバージョンを取得します。

**■対応した InfoSOSA 上位通信コマンド**

SI02

**■書式**

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

**■パラメータ**

| パラメータ | 説明 |
|---|---|
| std::string* _isVer | IS-APP バージョン |
| std::string* _userVer | IS-APP に表示されている画面データに設定されたユーザバージョン<br>※ユーザバージョンは InfoSOSA ビルダで設定します。詳細は別紙「InfoSOSA ビルダ操作マニュアル」を参照ください。 |

**■戻り値**

true:成功
false:失敗

**■使用例**

```
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

IS-APP の表示を指定したベース画面 ID の画面に切り替えます。

**■対応した InfoSOSA 上位通信コマンド**

SC10

**■書式**

```
bool setBaseScreen(std::string* _screenID)
```

**■パラメータ**

| パラメータ | 説明 |
|---|---|
| std::string* _screenID | 切り替え先のベース画面 ID |

**■戻り値**

true:成功
false:失敗

**■使用例**

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

## getBaseScreen

IS-APP の表示中のベース画面 ID を取得します。

**■対応した InfoSOSA 上位通信コマンド**

SC11

**■書式**

```
bool getBaseScreen(std::string* _screenID)
```

**■パラメータ**

| パラメータ | 説明 |
|---|---|
| std::string* _screenID | 現在表示中のベース画面 ID |

**■戻り値**

true:成功

false:失敗

**■使用例**

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

## showPopupScreenA

IS-APP に指定したポップアップ画面 A を表示します。

**■対応した InfoSOSA 上位通信コマンド**

SC13

**■書式**

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

**■パラメータ**

| パラメータ | 説明 |
|---|---|
| std::string* _screenID | 表示するポップアップ画面 ID |
| int _x | 表示するポップアップ画面の表示 X 座標 |
| int _y | 表示するポップアップ画面の表示 Y 座標 |

**■戻り値**

true:成功
false:失敗

**■使用例**

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

## showPopupScreenB

IS-APP に指定したポップアップ画面 B を表示します。

**■対応した InfoSOSA 上位通信コマンド**

SC14

**■書式**

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

**■パラメータ**

| パラメータ | 説明 |
|---|---|
| std::string* _screenID | 表示するポップアップ画面 ID |
| int _x | 表示するポップアップ画面の表示 X 座標 |
| int _y | 表示するポップアップ画面の表示 Y 座標 |

**■戻り値**

true:成功
false:失敗

**■使用例**

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

### hidePopupScreenA

IS-APP のポップアップ画面 A を閉じます。

**■対応した InfoSOSA 上位通信コマンド**

SC15

**■書式**

```
bool hidePopupScreenA()
```

**■パラメータ**

なし

**■戻り値**

true:成功
false:失敗

**■使用例**

```
pIsApi->hidePopupScreenA();
```

### hidePopupScreenB

IS-APP のポップアップ画面 B を閉じます。

**■対応した InfoSOSA 上位通信コマンド**

SC16

**■書式**

```
bool hidePopupScreenB()
```

**■パラメータ**

なし

**■戻り値**

true:成功
false:失敗

**■使用例**

```
pIsApi->hidePopupScreenB();
```

## getPopupScreen

IS-APP のポップアップ画面の表示状態を取得します。

**■対応した InfoSOSA 上位通信コマンド**

SC17

**■書式**

```
bool getPopupScreen(int* _a_state, int* _b_state)
```

**■パラメータ**

| パラメータ | 説明 |
|---|---|
| int* _a_state | ポップアップ画面 A の表示状態(0:OFF/1:ON) |
| int* _b_state | ポップアップ画面 B の表示状態(0:OFF/1:ON) |

**■戻り値**

true:成功
false:失敗

**■使用例**

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

## setBuzzer

IS-APP にブザーを鳴らす指示を送ります。

※IS-APP のブザー機能を無効にしている場合は、戻り値は true になりますがブザーは鳴りません。

**■対応した InfoSOSA 上位通信コマンド**

BZ01

**■書式**

```
bool setBuzzer(int _freq, int _msec)
```

**■パラメータ**

| パラメータ | 説明 |
|---|---|
| int _freq | ブザーを鳴らす周波数(Hz)<br>範囲：500～5000 |
| int _msec | ブザーを鳴らす時間(ms)<br>範囲：100～10000 の 100 刻み |

**■戻り値**

true:成功

false:失敗

**■使用例**

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

## getBuzzer

IS-APP のブザーの鳴動状態を取得します。

※IS-APP のブザー機能を無効にしている場合は、0(OFF)が設定されます。

**■対応した InfoSOSA 上位通信コマンド**

BZ02

**■書式**

```
bool getBuzzer(int* _state)
```

**■パラメータ**

| パラメータ | 説明 |
|---|---|
| int* _state | ブザーの状態(0:OFF/1:ON) |

**■戻り値**

true:成功

false:失敗

**■使用例**

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

## setProperty

IS-APP の部品、メモリのプロパティを変更することができます。メモリの値を変えたり、文字の色を変えたりすることができます。
プロパティ ID の書式、設定値は InfoSOSA の通信仕様に準じます。
InfoSOSA の通信仕様については、別紙「InfoSOSA リファレンスマニュアル」を参照ください。

**■対応した InfoSOSA 上位通信コマンド**

PA01

**■書式**

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

**■パラメータ**

| パラメータ | 説明 |
|---|---|
| std::string* _propertyID | 変更する対象のプロパティ ID<br>※ID は［所属 ID］.［部品/メモリ ID］.［プロパティ ID］の書式で設定します。詳細は別紙「InfoSOSA リファレンスマニュアル」を参照ください。 |
| CIsComVariant& _var | セットする値、文字列<br>※文字列の文字コードは必ず UTF8 で設定してください。 |

**■戻り値**

true:成功
false:失敗

**■使用例（数値）**

```
std::string propertyID;
CIsComVariant var; //数値を設定する場合は CIsComVariant 型の変数を作成
propertyID="@GLBMEM.GME00001.VALUE";
int value = 100;
var.SetValue(value); //CIsComVariant 型に数値を設定
pIsApi->setProperty(&propertyID, var);
```

**■使用例（文字列）**

```
std::string propertyID;
std::string strtmp;
CIsComString var; //文字列を設定する場合は CIsComString 型の変数を作成
propertyID="@GLBMEM.GME00002.TEXT";
strtmp = "ABC¥¥nXYZ";
var.SetString(strtmp.size(),(char*)strtmp.c_str()); //CIsComString 型に文字列を設定
pIsApi->setProperty(&propertyID, (CIsComVariant&)var); // CIsComVariant 型にキャスト
```

※改行は文字列¥n です。（使用例は制御文字列¥n にならないように¥¥n と記述）

## getProperty

IS-APP の部品、メモリのプロパティを取得することができます。
プロパティ ID の書式、設定値は InfoSOSA の通信仕様に準じます。
InfoSOSA の通信仕様については、別紙「InfoSOSA リファレンスマニュアル」を参照ください。

**■対応した InfoSOSA 上位通信コマンド**

PA02

**■書式**

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

**■パラメータ**

| パラメータ | 説明 |
|---|---|
| std::string* _propertyID | プロパティを取得する対象のプロパティ ID<br>※ID は［所属 ID］.［部品/メモリ ID］.［プロパティ ID］の書式で設定します。詳細は別紙「InfoSOSA リファレンスマニュアル」を参照してください。 |
| CIsComVariant& _var | 取得した値、文字列<br>※文字列の文字コードは UTF8 になります。 |

**■戻り値**

true:成功
false:失敗

**■使用例（数値）**

```
std::string propertyID;
CIsComVariant var; //数値を取得する場合は CIsComVariant 型の変数を作成
propertyID="@GLBMEM.GME00001.VALUE";
pIsApi->getProperty(&propertyID, var);
std::cout <<  propertyID << "=" << var.GetValue() << std::endl;
```

**■使用例（文字列）**

```
std::string propertyID;
std::string recv_str;
CIsComString var; //文字列を取得する場合は CIsComString 型の変数を作成
propertyID="@GLBMEM.GME00002.TEXT";
pIsApi->getProperty(&propertyID, (CIsComVariant&)var); // CIsComVariant 型にキャスト
recv_str=std::string(var.GetStringBuffer(),var.GetStringLength());
std::cout << propertyID << "=" << recv_str << std::endl;
```

## setGroupMemory

IS-APP のグローバルメモリグループの値を設定することができます。
グローバルメモリグループは、あらかじめ InfoSOSA ビルダで作成しておく必要があります。
グローバルメモリグループについては、別紙「InfoSOSA リファレンスマニュアル」を参照ください。

**■対応した InfoSOSA 上位通信コマンド**

PA05

**■書式**

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

**■パラメータ**

| パラメータ | 説明 |
|---|---|
| std::string* _groupID | 値を設定する対象のグループ ID |
| CIsComVariantList& _varList | 設定する値、文字列を格納したリストデータ<br>※文字列の文字コードは必ず UTF8 で設定してください。 |

**■戻り値**

true:成功
false:失敗

**■使用例**

```
std::string groupID;
std::string strtmp;
CIsComVariantList varList;
CIsComVariant var1; //数値を設定する場合は CIsComVariant 型の変数を作成
CIsComString var2; //文字列を設定する場合は CIsComString 型の変数を作成

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

groupID = "GRP00001";
varList.Add(var1);
varList.Add((CIsComVariant&)var2); // CIsComVariant 型にキャストしてリストに格納

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

## getGroupMemory

IS-APP のグローバルメモリグループの値を取得することができます。
グローバルメモリグループは、あらかじめ InfoSOSA ビルダで作成しておく必要があります。
グローバルメモリグループについては、別紙「InfoSOSA リファレンスマニュアル」を参照ください。

**■対応した InfoSOSA 上位通信コマンド**

PA06

**■書式**

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

**■パラメータ**

| パラメータ | 説明 |
|---|---|
| std::string* _groupID | 値を取得する対象のグループ ID |
| CIsComVariantList& _varList | 値、文字列を格納するリストデータ<br>※空である必要があります。<br>※文字列の文字コードは UTF8 になります。 |

**■戻り値**

true:成功

false:失敗

**■使用例**

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

groupID = "GRP00001";
varList.Clear(); //getGroupMemory で格納するリストデータは空
pIsApi->getGroupMemory(&groupID,varList);

std::cout << groupID << "=" ;
size = varList.Size();
for(int i = 0; i < size; ++i){
  //リストに格納されているデータは数値か文字列分からないため dynamic_cast を行い判別
  pisstr = dynamic_cast <CIsComString*>(varList.GetElem(i));
  if(pisstr){ //文字列(CIsComString)の場合は dynamic_cast が成功
    recv_str = std::string(pisstr->GetStringBuffer(),pisstr->GetStringLength());
    std::cout << recv_str ;
  }
  else{ //数値の場合は CIsComString への dynamic_cast が行えない
    std::cout << varList.GetElem(i)->GetValue();
  }
  if(i<size-1){
    std::cout << "," ;
  }
}
std::cout << std::endl;
```

## callSubRoutine

IS-APP のサブルーチンを実行します。
サブルーチンは、あらかじめ InfoSOSA ビルダで作成しておく必要があります。
サブルーチンについては、別紙「InfoSOSA リファレンスマニュアル」を参照ください。

**■対応した InfoSOSA 上位通信コマンド**

PA07

**■書式**

```
bool callSubRoutine(std::string* _subroutineID)
```

**■パラメータ**

| パラメータ | 説明 |
|---|---|
| std::string* _subroutineID | 実行するサブルーチン ID |

**■戻り値**

true:成功
false:失敗

**■使用例**

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

## registerCallback

IS-APP のイベント通知／値通知を受信した時に実行するコールバック関数を登録します。コールバック関数を登録することで、IS-APP 画面に表示されたボタンを押した時に任意の関数を実行することができます。値通知の場合、コールバック関数には値のリスト（ClsComVariantList）が渡されます。イベント通知の場合は、同様に値のリストは渡されますが、要素数は 0 になります。
１つのイベントに対して登録可能なコールバック関数は１つのみになります。
イベント通知／値通知を行うためには、あらかじめ InfoSOSA ビルダでイベント通知／値通知アクションを設定する必要があります。
アクションについては、別紙「InfoSOSA リファレンスマニュアル」を参照ください。

**■書式**

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

**■パラメータ**

| パラメータ | 説明 |
|---|---|
| std::string* _eventID | 対象のイベント ID<br>※イベント ID は［所属 ID］.［部品/メモリ ID］.［イベント ID］の書式で設定します。詳細はリファレンスマニュアルを参照してください。 |
| EventCallbackFunc _func | 登録する関数※ |

※ 登録する関数は、すぐに処理が完了するようにしてください。関数内では「処理フラグを立てる」「イベントをキューに追加する」などのみを行い、実際の処理は別スレッド等で行うことを推奨致します。

**■戻り値**

true:成功
false:失敗

**■使用例（イベント通知）**

```
void event_callback(CIsComVariantList &list) //対象のイベント ID を受信した時に実行する関数
{
  //IS-APP のボタンを押されたときに実行する処理
   […略…]
}

int main()
{
   […略…]
  std::string eventID("BAS00001.BTN00001.PRESS"); //対象のイベント ID
  pIsApi->registerCallback(&eventID, event_callback);
   […略…]
}
```

**■使用例（値通知）**

```
void event_callback(CIsComVariantList &list) //対象のイベント ID を受信した時に実行する関数
{
  //IS-APP のボタンを押されたときに実行する処理
   […略…]

  //通知された値を表示
  int size;
  CIsComString* pisstr;
  std::string str;

  size = list.Size();
  std::cout << "RECV=" ;
  for(int i = 0; i < size; ++i){
    //通知されるデータは数値か文字列分からないため dynamic_cast を行い判別
    pisstr = dynamic_cast<CIsComString*>(list.GetElem(i));
    if(pisstr){ //文字列(CIsComString)の場合は dynamic_cast が成功
      str = std::string(pisstr->GetStringBuffer(),pisstr->GetStringLength());
      std::cout << str ;
    }
    else{ //数値の場合は CIsComString への dynamic_cast が行えない
      std::cout << list.GetElem(i)->GetValue() ;
    }
    if(i<size-1){
      std::cout << "," ;
    }
  }
  std::cout << std::endl;
}

int main()
{
   […略…]
  std::string eventID("BAS00001.BTN00001.PRESS"); //対象のイベント ID
  pIsApi->registerCallback(&eventID, event_callback);
   […略…]
}
```

## unregisterCallback

コールバック関数の登録を解除します。

**■書式**

```
bool unregisterCallback(std::string* _eventID)
```

**■パラメータ**

| パラメータ | 説明 |
|---|---|
| std::string* _eventID | 対象のイベント ID |

**■戻り値**

true:成功

false:失敗

**■使用例**

```
pIsApi-> unregisterCallback(&eventID);
```

## 2.2.8　ClsComVariant

IS-API において、値／文字列を格納するクラスです。文字列を設定／取得する場合は、派生クラスの ClsComString を使用します。

### ヘッダファイル

CIsComVariant.h

### 公開機能

#### SetValue

ClsComVariant クラスに値を設定します。

**■書式**

bool SetValue(int value)

**■パラメータ**

| パラメータ | 説明 |
|---|---|
| int value | 値 |

**■戻り値**

true:成功
false:失敗

**■使用例**

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

## GetValue

ClsComVariant クラスの値を取得します。

**■書式**

int GetValue()

**■パラメータ**

なし

**■戻り値**

値

**■使用例**

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

## 2.2.9　ClsComString

IS-API において、文字列を格納するクラスです。ClsComVariant クラスの派生クラスになります。ClsApi::setProperty()の引数に使用する時などは ClsComVariant クラスにキャストして使用します。

### ヘッダファイル

CIsComVariant.h

### 継承クラス

ClsComVariant

### 公開機能

#### SetString

ClsComString クラスに文字列を設定します。

**■書式**

```
bool SetString(int bytes, char* str)
```

**■パラメータ**

| パラメータ | 説明 |
|---|---|
| int bytes | 文字列長(byte) |
| char* str | 文字列へのポインタ<br>※文字列の文字コードは必ず UTF8 で設定してください。 |

**■戻り値**

true:成功
false:失敗

**■使用例**

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

※改行は文字列¥n です。（使用例は制御文字列¥n にならないように¥¥n と記述）

## GetStringBuffer

ClsComString クラスの文字列へのポインタを取得します。
文字列の文字コードは UTF8 になります。

**■書式**

```
char* GetStringBuffer()
```

**■パラメータ**

なし

**■戻り値**

文字列データへのポインタ

## GetStringLength

ClsComString クラスの文字列の文字列長を取得します。

**■書式**

```
int GetStringLength()
```

**■パラメータ**

なし

**■戻り値**

文字列長(byte)

**■使用例**

```
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　ClsComVariantList

ClsComVariant クラスのコンテナクラスです。グループ設定や値通知など複数の ClsComVariant データを扱う場合に使用します。

### ヘッダファイル

CIsComVariantList.h

### 公開機能

#### Add

ClsComVariantList クラスの末尾に ClsComVariant データを追加します。

**■書式**

bool Add(CIsComVariant& elem)

**■パラメータ**

| パラメータ | 説明 |
|---|---|
| CIsComVariant& elem | 追加する CIsComVariant データ |

**■戻り値**

true:成功
false:失敗

#### GetElem

ClsComVariantList クラスから指定した位置の ClsComVariant データを取得します。

**■書式**

CIsComVariant* GetElem(int index)

**■パラメータ**

| パラメータ | 説明 |
|---|---|
| int index | 取り出す CIsComVariant データの位置 |

**■戻り値**

CIsComVariant データへのポインタ

## Size

ClsComVariantList クラスに登録されている ClsComVariant データの数を取得します。

**■書式**

```
int Size()
```

**■パラメータ**

なし

**■戻り値**

ClsComVariantList クラスに登録されている ClsComVariant データ数

## Clear

ClsComVariantList クラスに登録されたデータを全て削除します。

**■書式**

```
bool Clear()
```

**■パラメータ**

なし

**■戻り値**

true:成功
false:失敗

## ReplaceElem

ClsComVariantList クラスの指定した位置に ClsComVariant データを上書きします。上書きされたデータへのポインタが戻り値に設定されます。

**■書式**

```
CIsComVariant* ReplaceElem(int index, CIsComVariant& elem)
```

**■パラメータ**

| パラメータ | 説明 |
|---|---|
| int index | CIsComVariant データの登録位置 |
| CIsComVariant& elem | 登録する CIsComVariant データ |

**■戻り値**

上書きした位置に登録されていた CIsComVariant データへのポインタ

**■使用例**

```
CIsComVariantList varList;
   […略…]
//varList の 3 と 1 の要素を入れ替え
CIsComVariant *vartmp = varList.ReplaceElem(1, (CIsComVariant&)(varList.GetElem(3)));
varList.ReplaceElem(3, (CIsComVariant&)(*vartmp));
```

# 2.3　ISAPP SETTING

## 2.3.1　概要

IS-APP を使用するための補助ツールです。IS-APP の実行スクリプトの作成をサポートします。

## 2.3.2　実行ファイル名

| 実行ファイル |
| --- |
| isapp_setting |

EM シリーズ本体にインストールが必要

インストール方法は、「1.6 IS-APP/IS-API/IS-APP SETTING の更新」を参照ください。

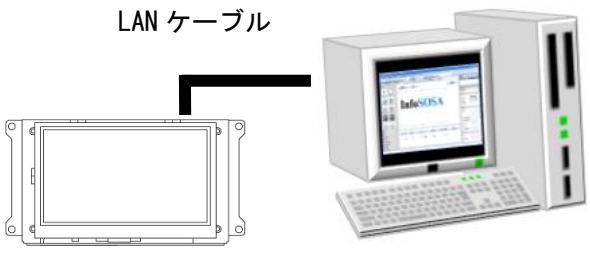
## 2.3.3　使用方法

事前に EM シリーズにインストールされているソフトウェアの更新が必要です。
「1.6 IS-APP/IS-API/IS-APP SETTING の更新」を参照ください。

### LAN ケーブルでのデータ転送

IS-APP で表示する画面データを PC から EM シリーズへ転送します。
PC(Windows)を操作します。
EM シリーズには SambaServer が搭載されており、EM シリーズ本体のユーザフォルダに Windows からアクセス可能です。

転送にはネットワーク設定が必要です。
「1.5　EM シリーズ本体と PC との接続」の手順を行ってください。




EM シリーズ本体と PC を LAN ケーブルで接続します。

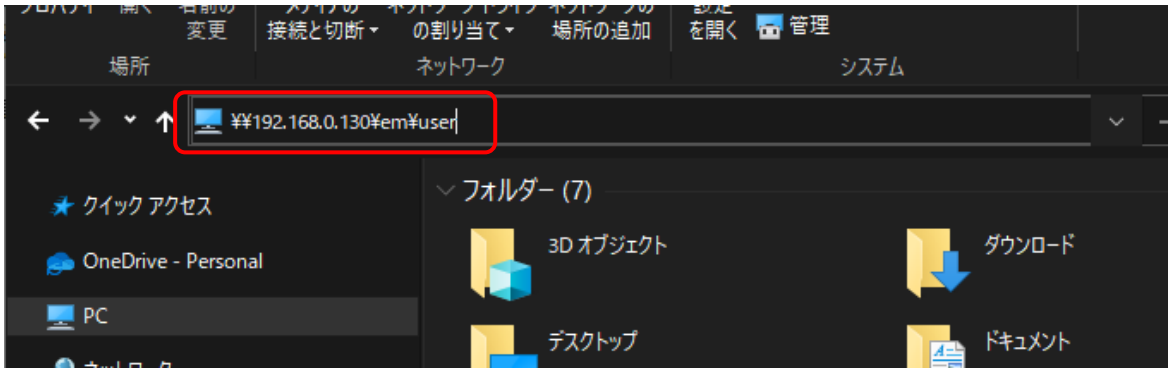
エクスプローラーなどで、以下のアドレスにアクセスします。

| | |
|---|---|
| EM シリーズ ユーザフォルダパス | ¥¥192.168.0.130¥em¥user |

※IP アドレスは工場出荷時のデフォルト値です。

以下のユーザでログインします。

| | |
|---|---|
| ユーザ | root |
| パスワード | 無し |




EM シリーズのユーザフォルダにアクセスできますので、エクスプローラーなどで EM シリーズにファイルをコピーします。

データの作成方法は、別紙「InfoSOSA ビルダ操作マニュアル」または「1.7　IS-APP を使用した HMI の開発方法」を参照ください。

| フォルダ／ファイル | 説明 |
|---|---|
| data | 転送用に変換された画面データ<br>※フォルダごとコピーしてください |

**【ご注意】**
**画面データを更新する場合は、必ず転送済みの画面データを一度フォルダごと削除してから転送してください。（フォルダ内に古いファイルがあると誤作動の原因になります）**

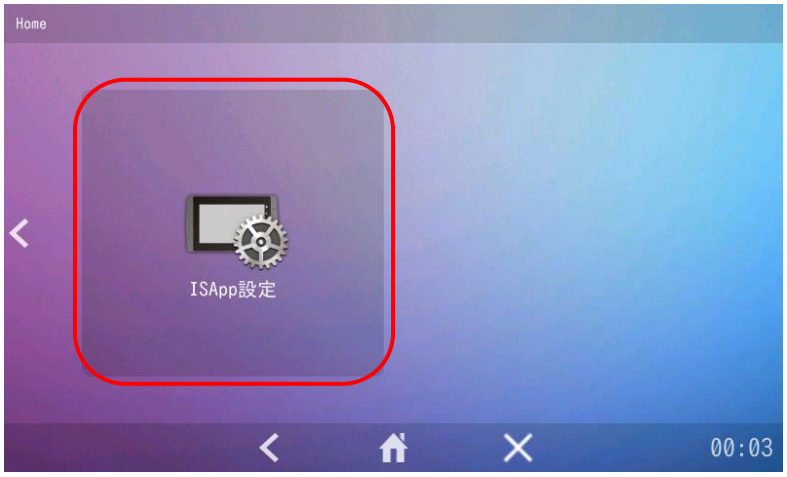
**【EM シリーズのユーザフォルダにアクセスできない場合】**

- LAN ケーブル用の IP アドレスが「192.168.10.*」の場合は通信できません。システム設定ツールより変更してください。
- お客様の PC 環境上に IP アドレスが同じ「192.168.0.130」になっている別のデバイスが存在していないかご確認ください。

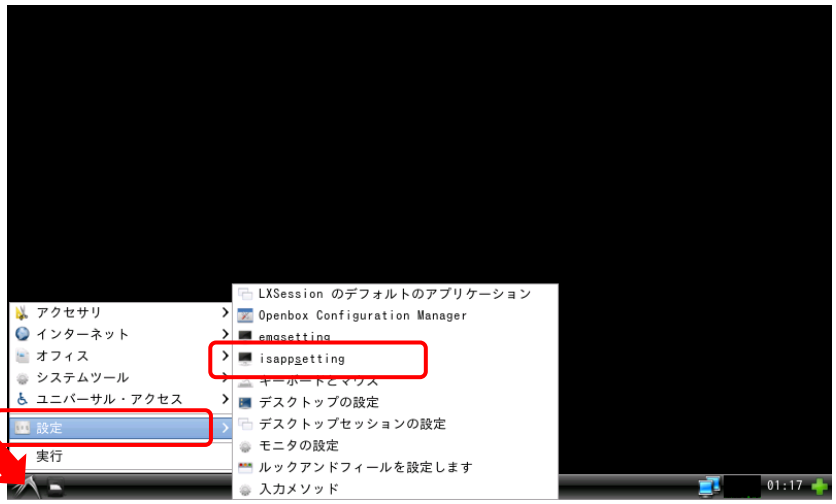
ここからは、EM シリーズ本体を操作します。

EMG ランチャーまたは、スタートメニューから ISAPP SETTING を起動します。

［EMG ランチャー］－［ISApp 設定］



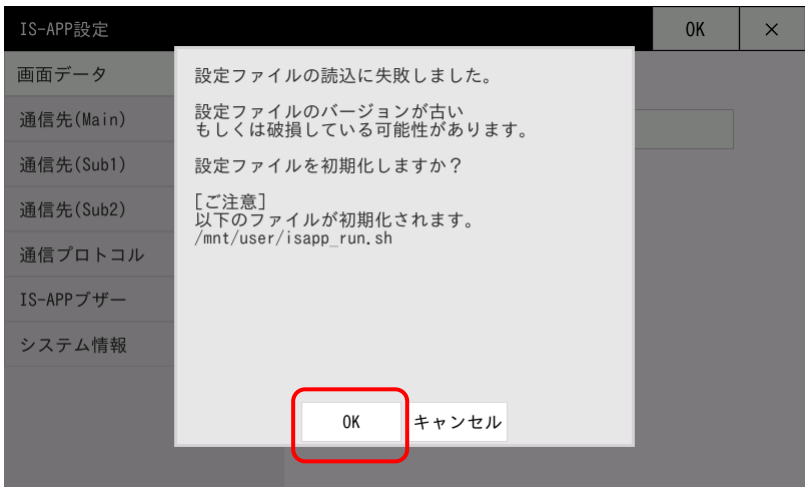

［スタートメニュー］－［設定］－［isappsetting］



次の場合は、下図のダイアログが表示されます。

・設定ファイルが未対応

・設定ファイルが破損している




「OK」ボタンをタッチしてください。

※以下のファイルが初期化されます。

/mnt/user/isapp_run.sh

## IS-APP の実行スクリプトの作成

IS-APP は、通信設定などを起動時のコマンドライン引数として指定します。
ISAPP SETTING を使用すると GUI でコマンドライン引数の設定を行うことができます。
（コマンドライン引数が設定された状態の起動スクリプトを作成します）

起動スクリプトは以下の場所の IS-APP を実行するように作成されます。

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

設定可能なコマンドライン引数一覧

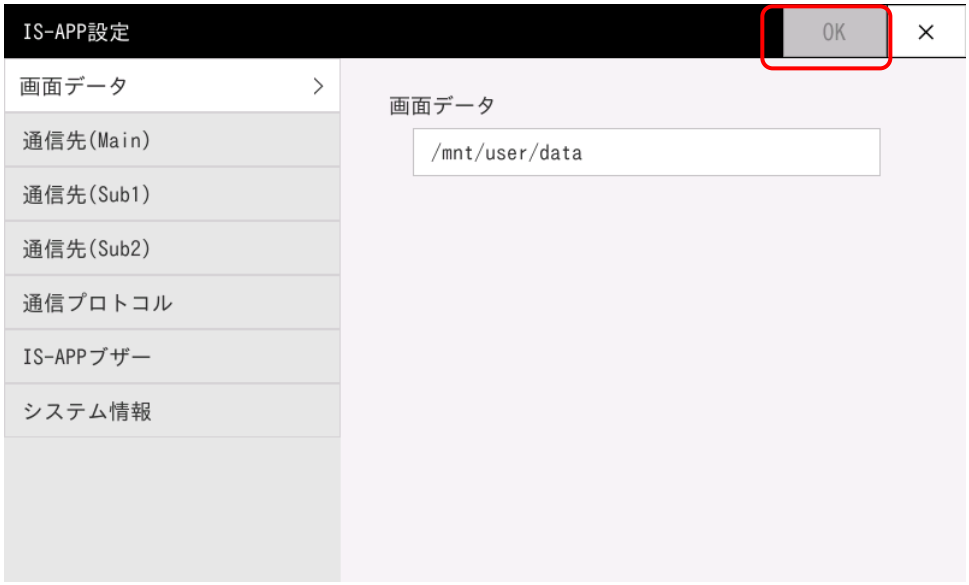
| 項目 | 説明 |
|---|---|
| 画面データフォルダ | IS-APP で表示する画面データフォルダを指定します。 |
| 通信設定 | IS-APP の外部機器との通信設定を行います。 |
| 通信プロトコル | IS-APP が外部機器と通信を行う時のプロトコルを設定します。 |
| ブザー | IS-APP がブザーデバイスを使用するか設定します。 |

設定が固定されるコマンドライン引数一覧

| 項目 | 説明 | 設定 |
|---|---|---|
| ウィンドウ | IS-APP のウィンドウサイズと位置を指定します。 | 自動 |
| サウンド | IS-APP のサウンド機能を設定します。 | デバイス名：default<br>ミキサー名：default<br>ボリューム名：PCM<br>※サウンド機能が無い機種は無効になります。 |

※　上記の設定を変更する場合は、「2.1.7 コマンドライン引数」を参考に、起動スクリプトを修正してください。

設定完了後、右上の OK ボタンをタッチすると起動スクリプトファイルが更新されます。

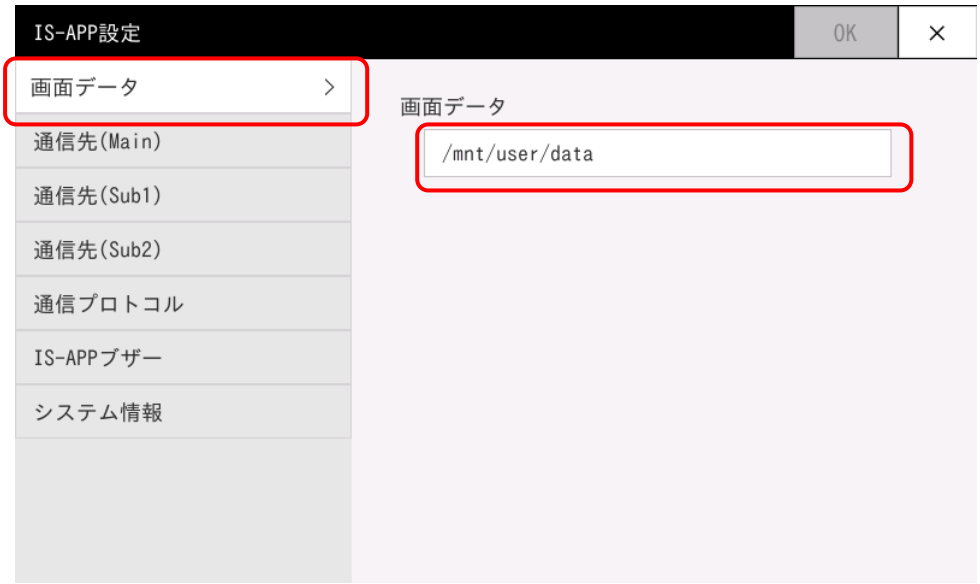



起動スクリプトは以下の場所に作成されます。

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

## 画面データ

画面データフォルダを変更する場合は、メニューから「画面データ」を選択し、フォルダパスが表示されているエリアをタッチしてください。



画面データフォルダを選択してください。


OK ボタンをタッチしてください。

### 通信先(Main) / 通信先(Sub1) / 通信先 (Sub2)

IS-APP の通信先を最大３つ設定できます。
指定可能な組み合わせは以下になります。

| 通信先(Main) | 通信先(Sub1) | 通信先(Sub2) |
|---|---|---|
| SIO | なし | なし |
| LAN | なし | なし |
| IS-API | なし | なし |
| SIO | LAN | なし |
| SIO | IS-API | なし |
| LAN | SIO | なし |
| IS-API | SIO | なし |
| SIO | SIO | なし |
| SIO | SIO | LAN |
| SIO | SIO | IS-API |
| LAN | SIO | SIO |
| IS-API | SIO | SIO |

**【ご注意】**

- 通信先(Sub1)、通信先(Sub2)は順不同です。
- LAN、IS-API はどちらかしか指定できません。LAN（IS-API 含む）で複数接続する場合は、LAN のモードを TCP/IP サーバーに設定してください。
- SIO を複数指定する場合は、同じ通信ポート(COM1/COM2)は指定できません。
- アクション「イベントを上位へ通知する」「値を上位へ通知する」実行時のイベント通知先は、通信先(Main)に指定したインタフェースになります。通信先(Sub1)、通信先(Sub2)には通知されません。
- 起動伝文は全ての通信先に送信されます。※TCP/IP サーバーを除く
  [起動伝文(シリアル)]
  {STX}00s00000008000100000003DC{ETX}
  [起動伝文(LAN)]
  s000000800010000
- IS-API と接続した場合、文字コード設定(SI03)が自動的に「Unicode （UTF-16LE）」に設定されます。IS-API をご使用される場合は、「Shift JIS」に設定しないでください。IS-API が正常に動作しなくなります。
- 文字コード設定は、全ての通信インタフェースに適用されます。IS-API を使用される場合は、SIO や LAN も「Unicode （UTF-16LE）」で通信してください。

### 通信先(IS-API)

通信を内部通信（IS-API）に設定する場合は、通信先に「IS-API」を選択してください。


| 項目 | 説明 |
|---|---|
| IS-API のポート番号 | 通信先アプリケーション(IS-API)のポート番号を指定します。 |

### 通信先(SIO)

通信をシリアル通信（SIO）に設定する場合は、通信先に「SIO」を選択してください。


| 項目 | 説明 |
|---|---|
| 通信ポート ※1 | COM1(SIO1)、COM2(SIO2)から選択します。 |
| 種別※1 | RS232C／RS422／RS485 から選択します。 |
| 通信速度(bps) ※1 | 通信速度を設定します。 |
| パリティ※1 | パリティを設定します。 |
| 自局アドレス ※2 | 機器の自局アドレスを設定します。 |
| 衝突再送回数 ※2 | 通信が衝突した場合の再送回数を設定します。 |
| 衝突再送間隔(ms) ※2 | 通信が衝突した場合の再送間隔を設定します。（単位：ミリ秒） |
| 送信権タイムアウト(ms) ※2 | 通信回線の空きを待つ場合のタイムアウトを設定します。（単位：ミリ秒） |

※1　製品により使用可能な設定は異なります。お使いの製品仕様書をご確認ください。
※2　種別に RS485 を選択した時のみ表示されます。

### 通信先(LAN) UDP/IP

通信を LAN 通信(UDP/IP)に設定する場合は、通信先に「LAN」を選択し、モードを「UDP/IP」に設定してください。


| 項目 | 説明 |
|---|---|
| アドレス | 通信先機器の IP アドレスを設定します。 |
| ポート番号 | 通信先機器のポート番号を指定します。 |

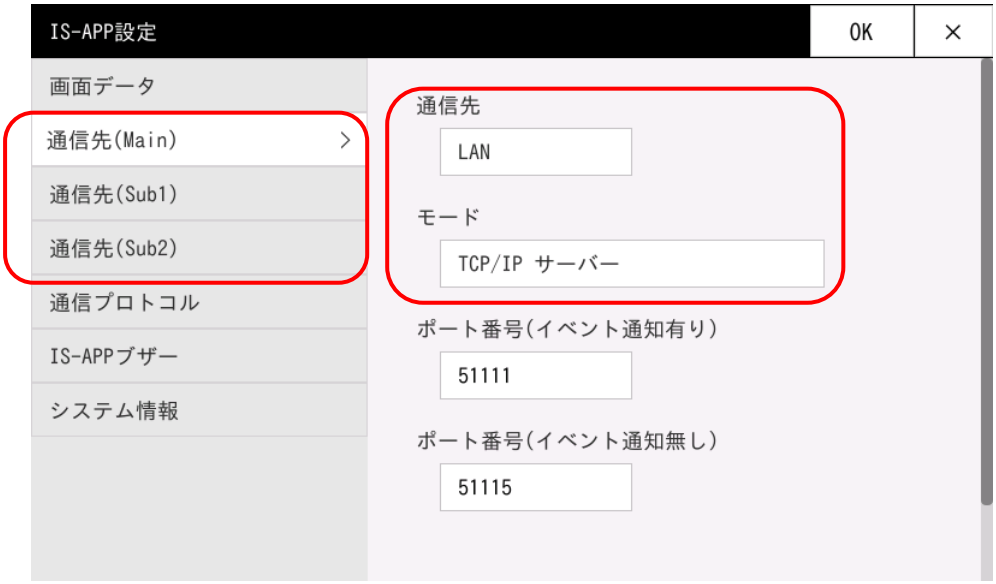
## 通信先(LAN) TCP/IP クライアント

通信を LAN 通信(TCP/IP クライアント)に設定する場合は、通信先に「LAN」を選択し、モードを「TCP/IP クライアント」に設定してください。


| 項目 | 説明 |
|---|---|
| アドレス | 通信先機器の IP アドレスを設定します。 |
| ポート番号 | 通信先機器のポート番号を指定します。 |
| TCP コネクション確立要求間隔(s) | TCP/IP 通信時の接続要求の間隔を設定します。(秒) |

## 通信先(LAN) TCP/IP サーバー

通信をLAN通信(TCP/IP サーバー)に設定する場合は、通信先に「LAN」を選択し、モードを「TCP/IP サーバー」に設定してください。




| 項目 | 説明 |
|---|---|
| ポート番号（イベント通知有り） | 待ち受けポート番号を指定します。<br>最大コネクション数：１<br>このポートに接続した機器に対してのみイベント通知を行います。<br>※通信先(Main)に設定する必要があります。通信先(Sub)に設定した場合は、イベント通知有りポートに対してもイベント通知は行われません。 |
| ポート番号（イベント通知無し） | 待ち受けポート番号を指定します。<br>最大コネクション数：３<br>※通信先(Main)に設定した場合でも、このポートに接続した機器に対してはイベント通知は行われません。 |

## 通信プロトコル

通信プロトコルを設定する場合は、メニューから「通信プロトコル」を選択してください。

**【ご注意】**

- 本設定は全ての通信先に適用されます。
- 通信先に IS-API※が含まれる場合は、「InfoSOSA 標準プロトコル／即時通知」の設定にする必要があります。※TCP/IP サーバーで IS-API と接続する場合も含む

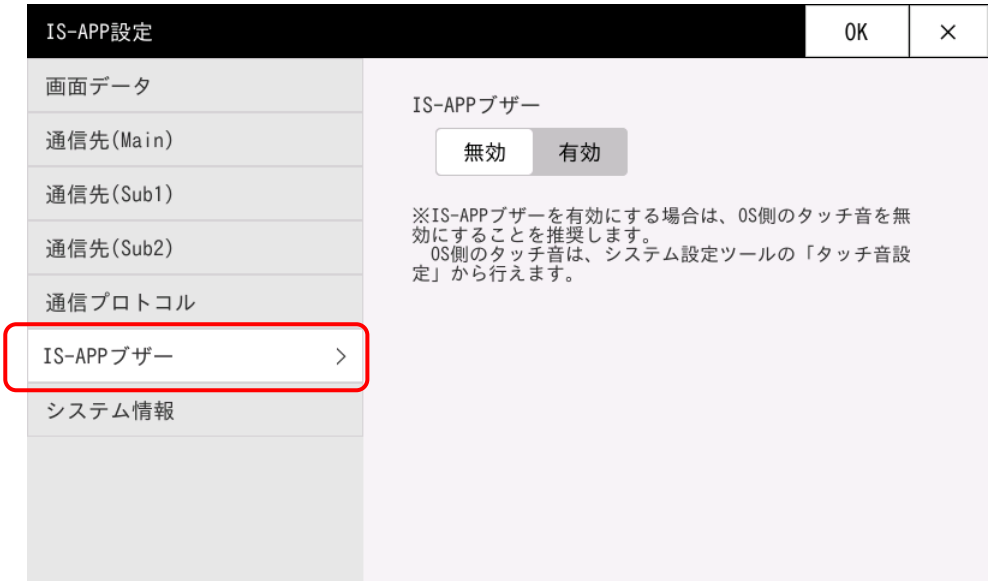

| 項目 | 説明 |
|---|---|
| 通信プロトコル | InfoSOSA 標準プロトコル／通常プロトコルから選択します。 |
| 通知方法 | 即時通知／上位から要求から選択します。 |
| イベント応答監視時間 ※1 | 通常プロトコルの再送までの時間を指定します。（単位：秒） |
| 再送回数 ※1 | 通常プロトコルの再送回数を指定します。 |

※1　通信プロトコルに通常プロトコルを選択した時のみ表示されます。

通信プロトコル、通知方法については、別紙「InfoSOSA リファレンスマニュアル」を参照ください。

## ブザー

ブザーを設定する場合は、メニューから「ブザー」を選択してください。

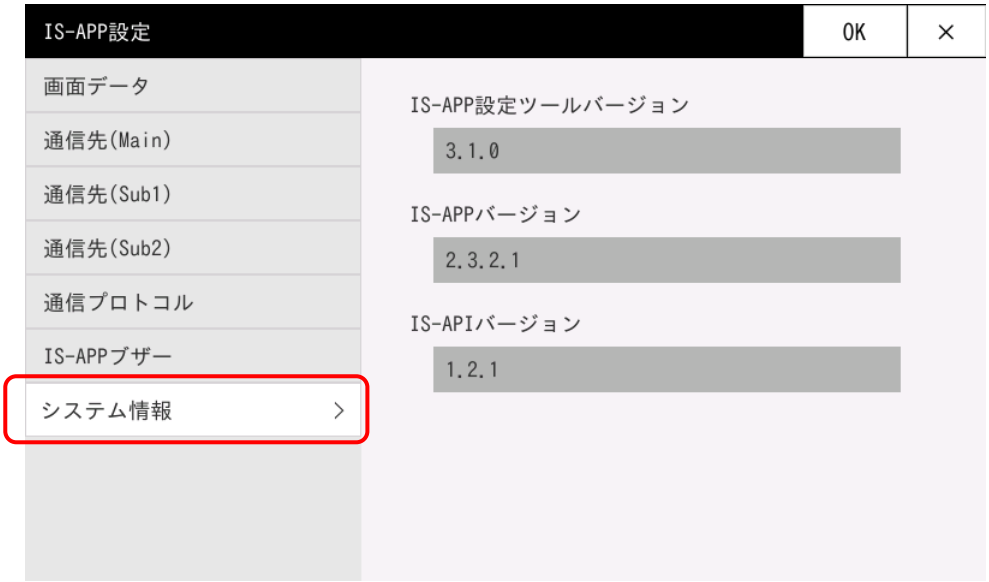



| 項目 | 説明 |
|---|---|
| IS-APP ブザー | 無効／有効から選択します。 |

※IS-APP ブザーを有効にする場合は、OS 側のタッチ音を無効にすることを推奨します。
※OS 側のタッチ音は、システム設定ツールの「タッチ音設定」から行えます。

## システム情報

バージョンを表示します。



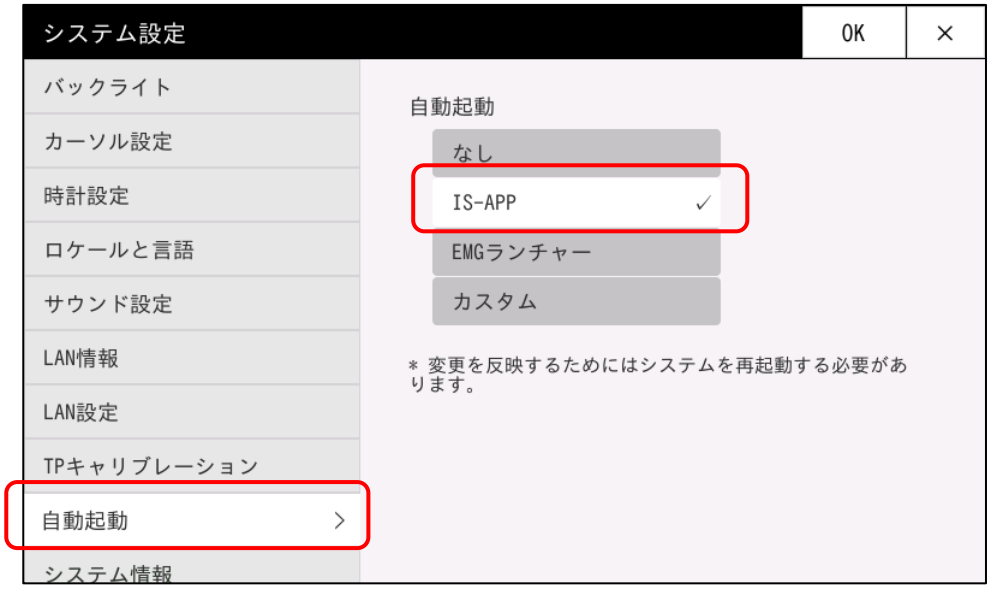

| 項目 | 説明 |
|---|---|
| IS-APP 設定ツールバージョン | 本ツールのバージョンを表示します。 |
| IS-APP バージョン | /usr/bin/is_app のバージョンを表示します。 |
| IS-API バージョン | /usr/lib/libisapi.so のバージョンを表示します。 |

## IS-APP の自動起動設定

電源 ON 時に自動的に IS-APP を起動する設定は、システム設定ツールから行えます。

自動起動は、起動時に以下のスクリプトを実行するように設定します。

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

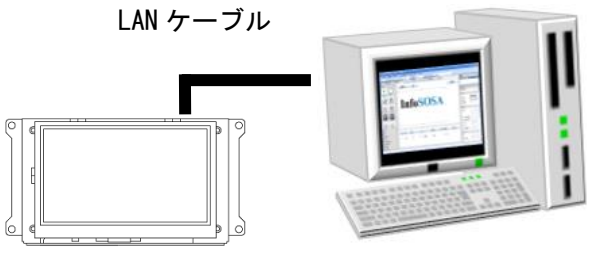



詳しくは、別紙「EM シリーズ ツールマニュアル」を参照ください。

## 自動起動設定後の画面データの更新

IS-APP の画面データを更新する場合は、以下の手順を行ってください。
通信設定を変更する場合は、次項目（通信設定の変更）を参照ください。

事前にネットワーク設定が必要です。
「1.5　EM シリーズ本体と PC との接続」の手順を行ってください。




EM シリーズ本体と PC を LAN ケーブルで接続します。

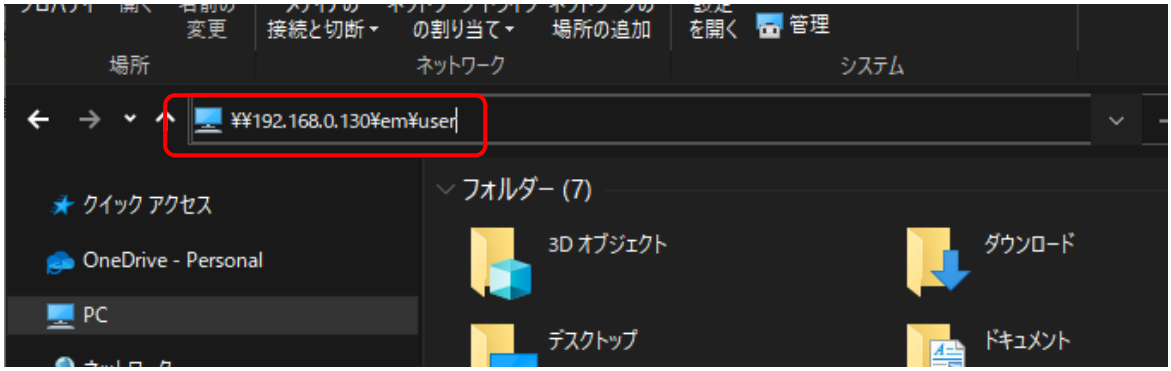
エクスプローラーなどで、以下のアドレスにアクセスします。

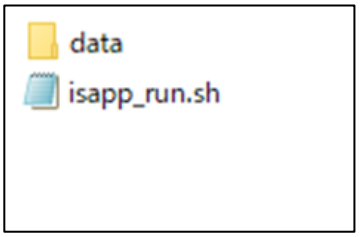
| | |
|---|---|
| EM シリーズ ユーザフォルダパス | ¥¥192.168.0.130¥em¥user |

※IP アドレスは工場出荷時のデフォルト値です。

以下のユーザでログインします。

| | |
|---|---|
| ユーザ | root |
| パスワード | 無し |




EM シリーズ本体に保存された画面データフォルダ（デフォルト名「data」）を削除します。

「data」フォルダを削除後、EM シリーズに、更新した画面データをコピーしてください。

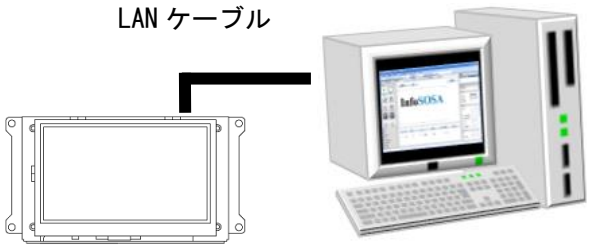
**【ご注意】**
**画面データを更新する場合は、必ず転送済みの画面データを一度フォルダごと削除してから転送してください。（フォルダ内に古いファイルがあると誤作動の原因になります）**

EM シリーズ本体の電源を入れ直すと更新された画面が表示されます。

## 自動起動設定後の通信設定の変更

IS-APP の通信設定を変更する場合は、一時的に自動起動を行わないようにする必要があります。

事前にネットワーク設定が必要です。
「1.5　EM シリーズ本体と PC との接続」の手順を行ってください。




EM シリーズ本体と PC を LAN ケーブルで接続します

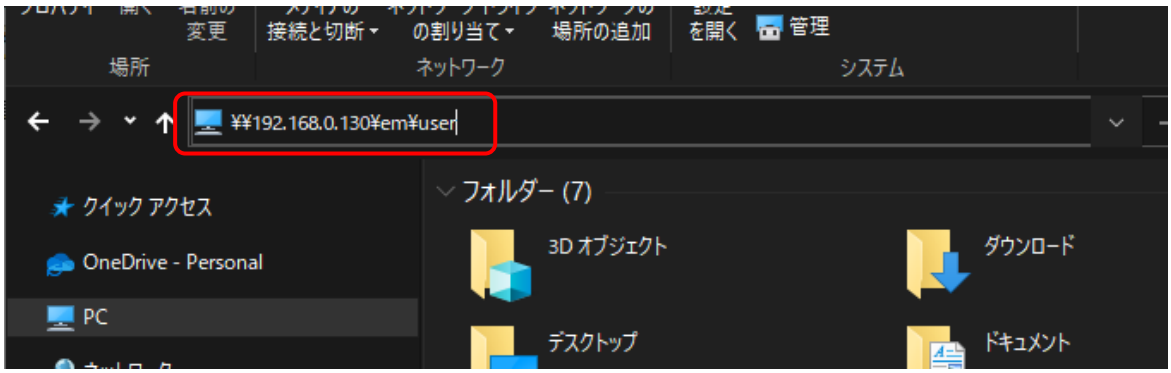
エクスプローラーなどで、以下のアドレスにアクセスします。

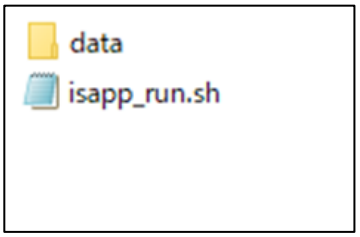
| | |
|---|---|
| EM シリーズ ユーザフォルダパス | ¥¥192.168.0.130¥em¥user |

※IP アドレスは工場出荷時のデフォルト値です。

以下のユーザでログインします。

| | |
|---|---|
| ユーザ | root |
| パスワード | 無し |






EM シリーズ本体に保存された IS-APP の実行スクリプトを（isapp_run.sh）を削除します。

EM シリーズ本体の電源を入れ直します。
IS-APP の自動起動が一時的に行なわれなくなっていますので、ISAPP SETTING を起動し、再設定を行ってください。

### EM シリーズのユーザフォルダにアクセスできない場合

EM シリーズでは、LAN ケーブル用の IP アドレス「192.168.0.130」と USB ケーブル用の IP アドレス「192.168.10.130」の２つが存在しています。LAN ケーブル用の IP アドレスは、システム設定ツールより変更可能ですが、USB ケーブル用の IP アドレス「192.168.10.130」と同じにしてしまうと通信が行えなくなりますのでご注意ください。
また、お客様環境に IP アドレスが「192.168.0.130」になっている別の機器が存在していると通信が行えませんのでご注意ください。

# 3章 その他

**章目次**

# 3.1　お問い合わせ

本書に関するお問い合わせは、下記へお願い致します。

**お電話でのお問い合わせ**

☎ 06-6147-6645

株式会社ディ・エム・シー　大阪技術センター
受付時間：平日　9：00～17：00
※土日・祝祭日・年末年始を除く

**メールでのお問い合わせ**

お問い合わせフォームで受け付けています。下記からご連絡ください。

✉ www.dush.co.jp/contact/

**よくあるご質問と回答集**

🌐 www.dush.co.jp/support/faq/



2023 年 12 月　第 10 版
発行所　株式会社ディ・エム・シー
〒108-0074　東京都港区高輪 2-18-10 高輪泉岳寺駅前ビル 11F
TEL：(03)-6721-6731 ㈹　FAX：(03)-6721-6732
URL：https://www.dush.co.jp/