

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。

本書は、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

本書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

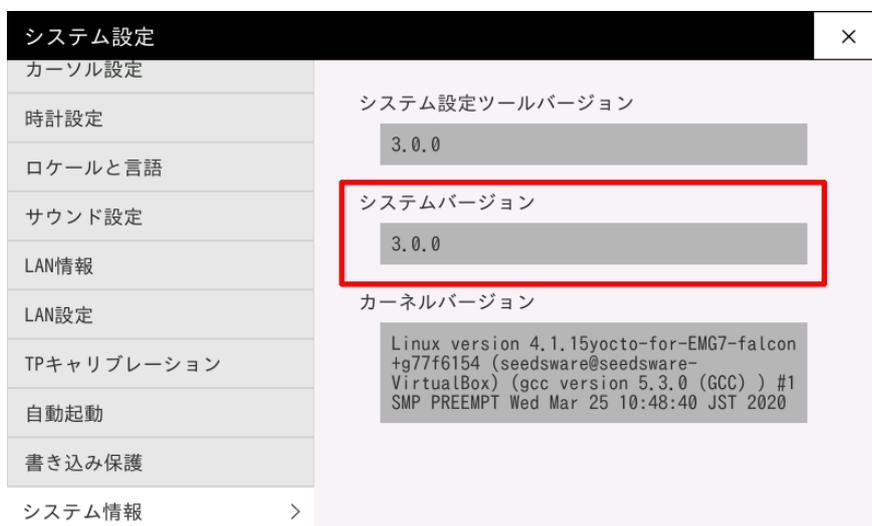
EM シリーズ本体のシステムバージョン

本書は、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 ~
-------------	---------



- 本書の著作権は、株式会社ディ・エム・シーが所有しています。
- 本製品および本書内容の一部、または全てを無断で掲載することは禁止されています。
- 本製品および本書の内容は予告なく変更することがあります。あらかじめご了承ください。
- 本書の内容は万全を期しておりますが、万一誤りや記載漏れ等お気づきの点がございましたら当社までご連絡ください。
- 本製品を使用したことによるお客様の損害やその他の不利益、または第三者からのいかなる請求につきましても当社はその責任を負いません。あらかじめご了承ください。
- Microsoft®、Windows®、Windows® 10、Windows® 11、Microsoft® .NET Framework は米国 Microsoft Corporation の米国およびその他の国における登録商標または商標です。
- その他の会社および製品名は、各社の商標または登録商標です。

目次

はじめに	i
目次	iii
1章 IS-APP について	1
1.1 IS-APP 概要	2
1.2 開発キットについて	3
1.3 InfoSOSA ビルダとは	6
1.4 IS-APP/IS-API/IS-APP SETTING の更新	7
2章 開発手順(チュートリアル)	9
2.1 InfoSOSA ビルダのインストール	13
2.2 Smart e-Studio のインストール	20
2.3 EM シリーズ本体と PC との接続	21
2.4 テンプレートパッケージ書き込み	28
2.5 システム仕様作成	35
2.6 InfoSOSA プロジェクトの作成	37
2.7 InfoSOSA シミュレータでの動作確認	52
2.8 IS-APP の通信設定	53
2.9 C++アプリケーションの作成	60
2.10 C++アプリケーションの起動設定	70
2.11 テンプレートパッケージのカスタマイズ	74
2.12 Smart e-Studio パッケージの書き込み	89
2.13 動作確認	91
2.14 各機能の詳細について	94
3章 リファレンス	95
3.1 IS-APP 詳細仕様	96
3.2 IS-API 詳細仕様	109
3.3 ISAPP SETTING	139
4章 その他	154
4.1 お問い合わせ	155

1 章 IS-APP について

章目次

1.1	IS-APP 概要.....	2
1.2	開発キットについて.....	3
1.3	InfoSOSA ビルダとは.....	6
1.4	IS-APP/IS-API/IS-APP SETTING の更新.....	7

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 開発キットについて

EM シリーズで IS-APP を使用して開発するには以下の2つが必要になります。

お持ちでない場合は、弊社担当営業までご連絡ください。

項目	内容
InfoSOSA 開発キット	InfoSOSA に関連したソフトウェア、ドキュメントが同梱されています。
EM シリーズ開発環境	EM シリーズをご使用頂く為の、ソフトウェア、ドキュメントが同梱されています。

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。

対象ユーザー

- ✓ 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。

対象ユーザー

- ✓ 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

上位通信テスト取扱説明書

上位通信テストの操作方法について記載しています。

※ 上位通信テストは、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
- ✓ 上位通信テストをご使用中に設定／操作の詳細を調べたい方

InfoSOSA Modbus 機器接続ガイド

InfoSOSA で Modbus プロトコル通信を使用する方法について記載しています。

対象ユーザ

- ✓ InfoSOSA の Modbus プロトコル通信機能について知りたい方

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 EM シリーズ開発環境に同梱されております。

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 Smart e-Studio のご使用方法を記載しています。Smart e-Studio は PC 上で動作するツールです。

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

EM シリーズ Smart e-Start マニュアル

EM シリーズの高速起動ソリューション Smart e-Start 対応製品のご使用方法を記載しています。

対象ユーザ

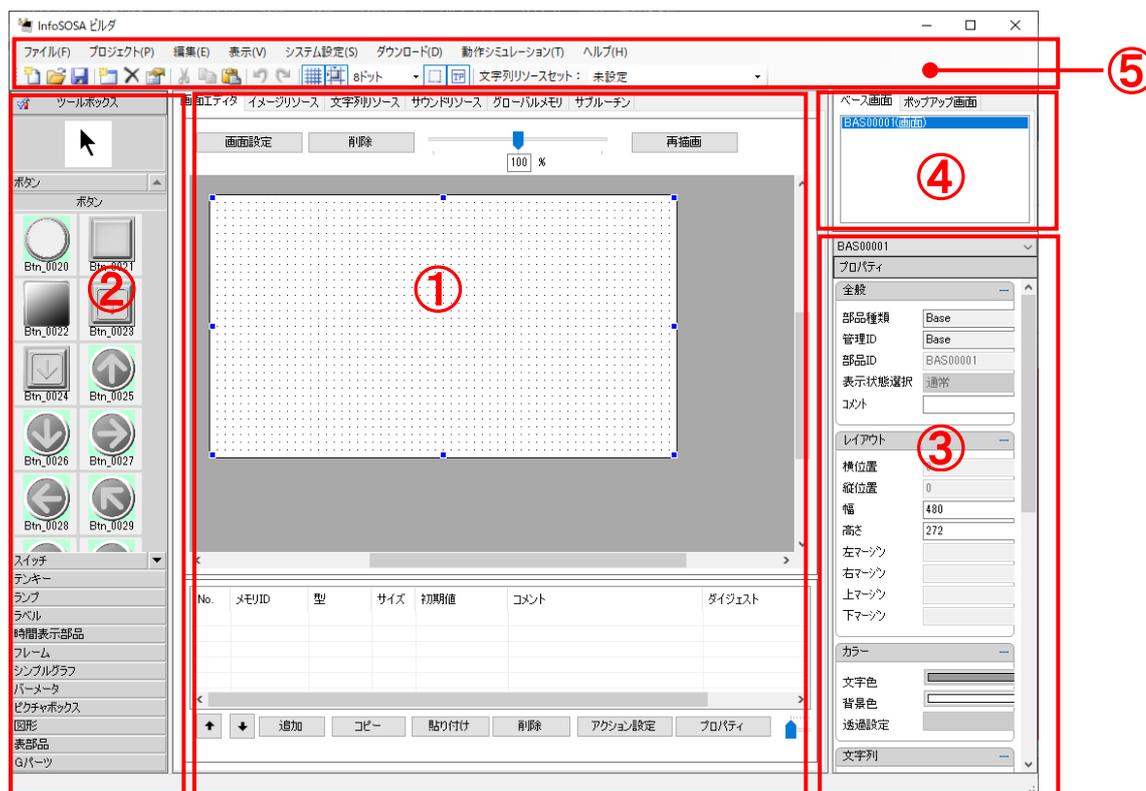
- ✓ Smart e-Start 対応 EM シリーズでソフトウェアを開発される方

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 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。

インストールする方法は以下の2種類の方法がございます。

必ずどちらかの手順を実施してください。InfoSOSA ビルダと InfoSOSA アプリケーションのバージョンが異なると正常に動作しません。

1.4.1 方法1:テンプレートパッケージを使用 (推奨)

後述のテンプレートパッケージを書き込むことで、更新が行われます。

テンプレートパッケージについては「[2 章 開発手順\(チュートリアル\)](#)」を参照ください。

1.4.2 方法2:手動で差し替える

テンプレートパッケージを使用しない場合は、手動で各ファイルを差し替えてください。

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。

※コマンドが実行できない場合は、/usr/bin/emsystem/wprotect_off と指定してください。

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

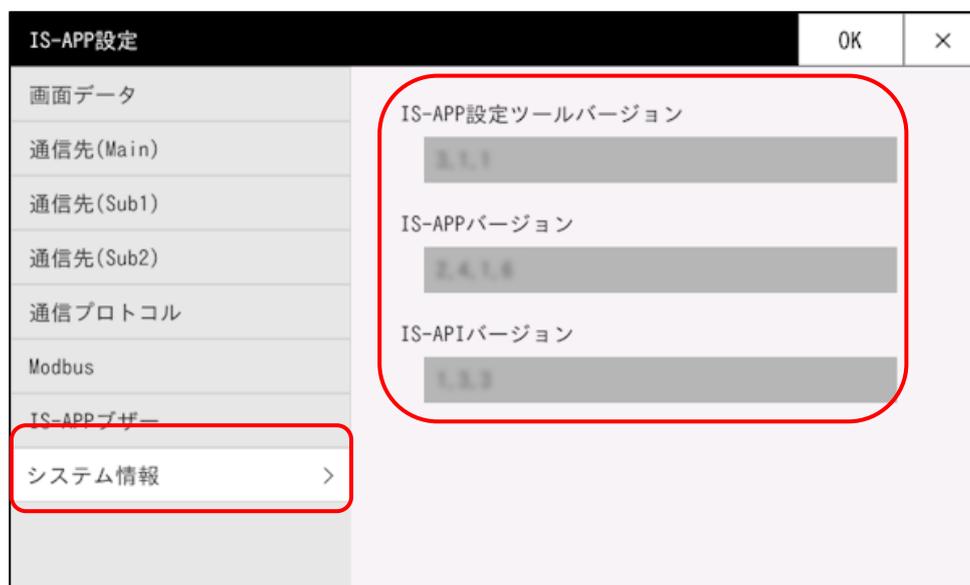
「is_app」「libisapi.so」「isapp_setting」は、InfoSOSA 開発キットデータ内にございます。
お使いの製品ごとに対応ファイルが異なりますのでご注意ください

型式	対応ファイル
EM8-***A7-****-**7	software¥IS-APP¥IS-APP-A7¥is_app
EMG8-***A7-****-**7	software¥IS-API¥Library¥IS-APP-A7¥libisapi.so
EMP-***A7-****-**7	software¥IS-APP_SETTING¥IS-APP-A7¥isapp_setting
EMG7-***A8-****-**7	software¥IS-APP¥IS-APP-A8¥is_app software¥IS-API¥Library¥IS-APP-A8¥libisapi.so software¥IS-APP_SETTING¥IS-APP-A8¥ 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。

2章 開発手順 (チュートリアル)

章目次

2.1	InfoSOSA ビルダのインストール.....	13
2.2	Smart e-Studio のインストール.....	20
2.3	EM シリーズ本体と PC との接続.....	21
2.4	テンプレートパッケージ書き込み.....	28
2.5	システム仕様作成.....	35
2.6	InfoSOSA プロジェクトの作成.....	37
2.7	InfoSOSA シミュレータでの動作確認.....	52
2.8	IS-APP の通信設定.....	53
2.9	C++アプリケーションの作成.....	60
2.10	C++アプリケーションの起動設定.....	70
2.11	テンプレートパッケージのカスタマイズ.....	74
2.12	Smart e-Studio パッケージの書き込み.....	89
2.13	動作確認.....	91
2.14	各機能の詳細について.....	94

概要

EM シリーズでは、InfoSOSA ビルダを使用してプログラミングを行わずに画面作成を行うことが可能です。

さらに InfoSOSA ビルダで実現が難しい機能は、C/C++などでプログラムを追加して、機能を拡張することも可能です。プログラム追加に関する項目は以下のアイコンを付加しています。



プログラム追加を行わない場合は、このアイコンが付いた項目はスキップ可能です。

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機能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の 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または IS-APP 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。

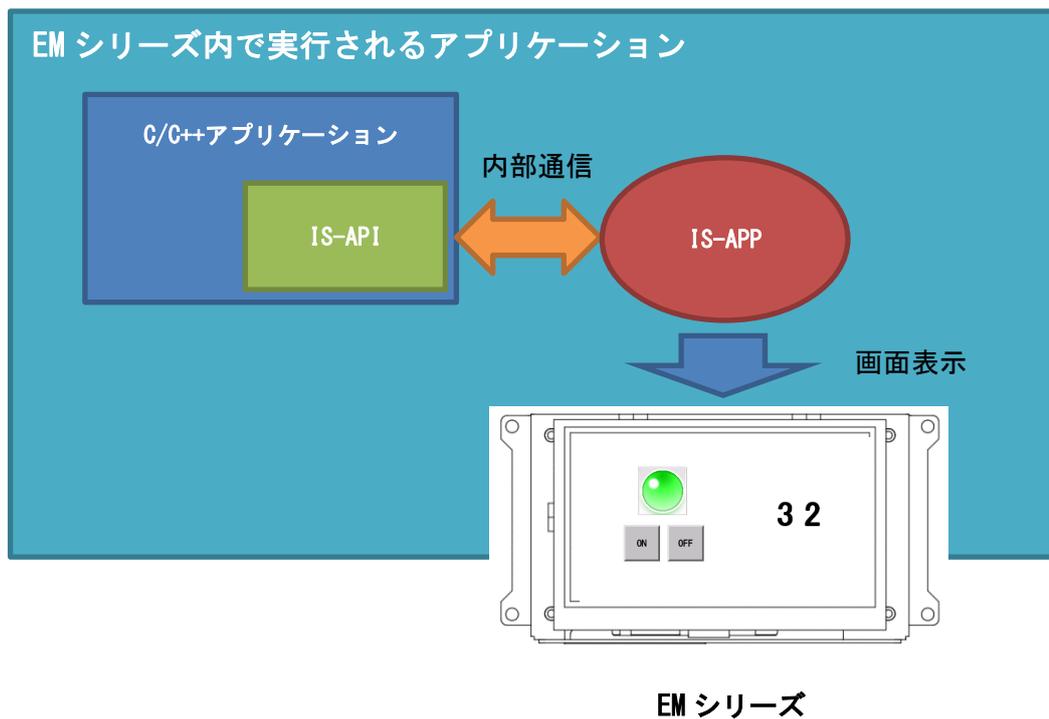
液晶画面上 (IS-APP) の ON/OFF ボタンをタッチすると画面上のランプが点灯消灯する

EX
EXTRA

液晶画面上 (IS-APP) の ON/OFF ボタンをタッチするとユーザ作成アプリケーション (C++プログラム) にイベントが通知される

EX
EXTRA

ON/OFF ボタンの状態に応じて、ユーザ作成アプリケーションから数字表示を更新する



開発の流れ

章	項目	内容
2.1	InfoSOSA ビルダのインストール	PC に画面作画ツールをインストールします。
2.2	Smart e-Studio のインストール	PC にデータ転送ツールをインストールします。
2.3	EM シリーズ本体と PC との接続	EM シリーズと接続ができるように PC の設定を変更します。
2.4	テンプレートパッケージの書き込み	EM シリーズを IS-APP の利用に適した状態に自動的に更新します。
2.5	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。
2.6	InfoSOSA プロジェクトの作成	InfoSOSA ビルダでボタンの配置や動作などの設定を行います。
2.7	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。
2.8	IS-APP の通信設定	IS-APP の起動パラメータ (通信、ブザーなど) 設定を行うファイルを作成します。
 2.9	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。
 2.10	C++アプリケーションの自動起動設定	InfoSOSA ビルダで IS-APP から C++アプリケーションを自動的に起動する設定を追加します。
2.11	テンプレートパッケージのカスタマイズ	テンプレートをベースにお客様のデータへの差し替え、追加を行い Smart e-Studio パッケージを作成します。
2.12	Smart e-Studio パッケージの書き込み	作成した Smart e-Studio パッケージを EM シリーズに書き込みます。
2.13	動作確認	実機での動作確認を行います。

2.1 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。

2.1.1 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) または Microsoft® Windows® 11 日本語版(64bit 版)※
必要なフレームワーク ランタイム	Microsoft® .NET Framework 3.5 Microsoft® .NET Framework 4.7 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いただきます。

※ Windows11 対応は Ver2.7.1 以降が対象になります。

2.1.2 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。

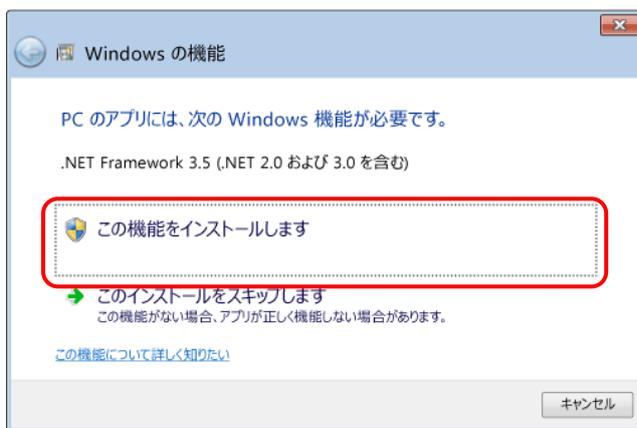
4. 生成されたフォルダ「InfoSOSABuilder***」内の「Setup.exe」を実行して下さい。

2.1.3 .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。

「.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ル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 Framework 3.5」のインストールを行ってください。

表示されない場合はそのまま次の項目に進んでください。



※ 「.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。

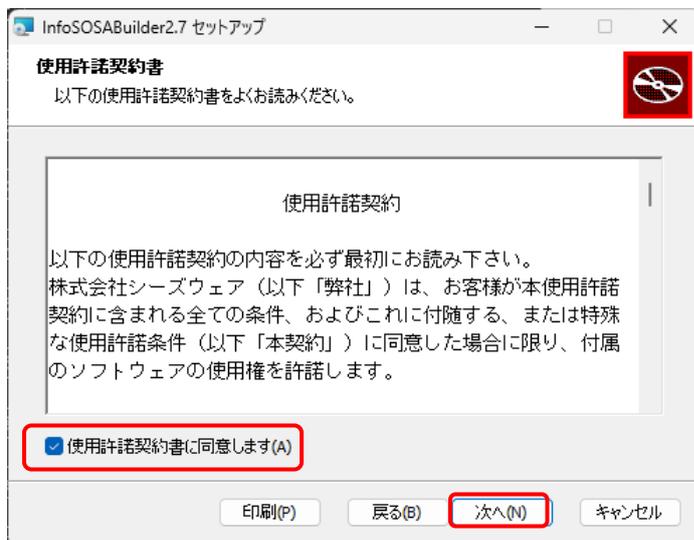
※ オフラインでインストールする場合は、Windows のインストールメディアをご使用頂く必要があります。

2.1.4 InfoSOSA ビルダ のインストール

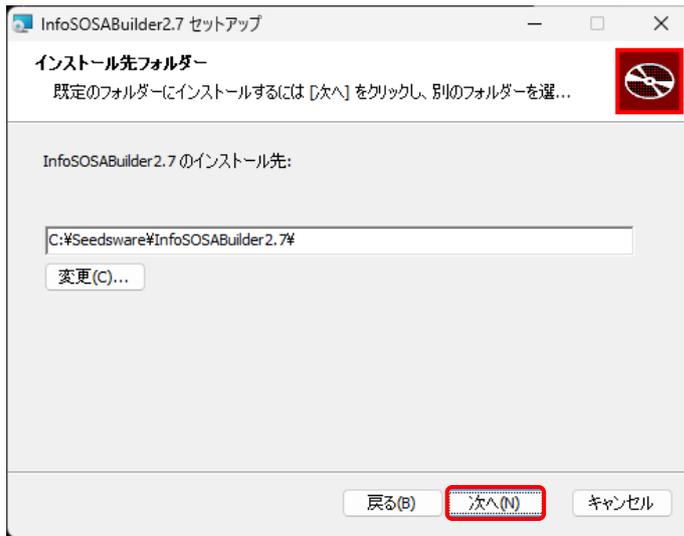
1. インストールを開始すると以下のようなダイアログが表示されます。
[次へ] ボタンをクリックします。



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。



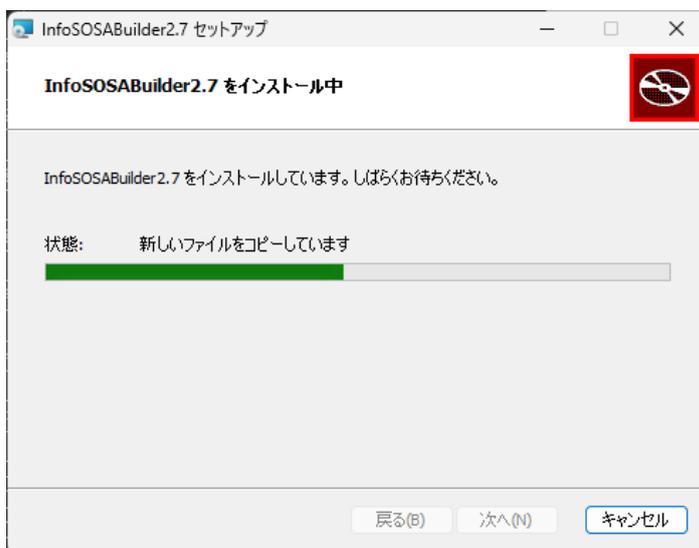
3. 保存先の指定を行い [次へ] ボタンをクリックします。



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。



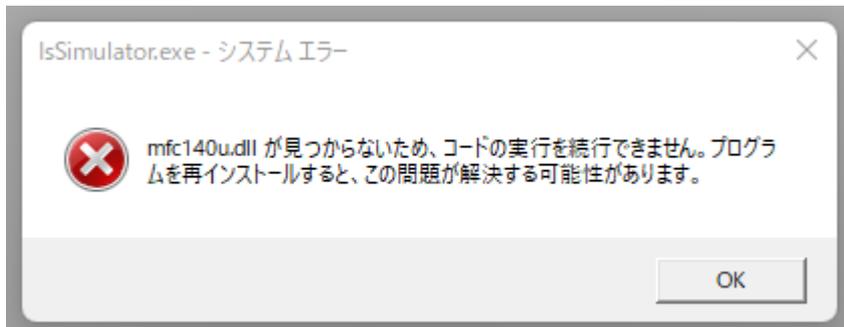
6. インストール完了です。



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。

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

PCに「Microsoft Visual C++ 2015 ランタイム」がインストールされていない場合はシミュレータ実行時に以下のようなエラーが表示されます。その場合は、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。

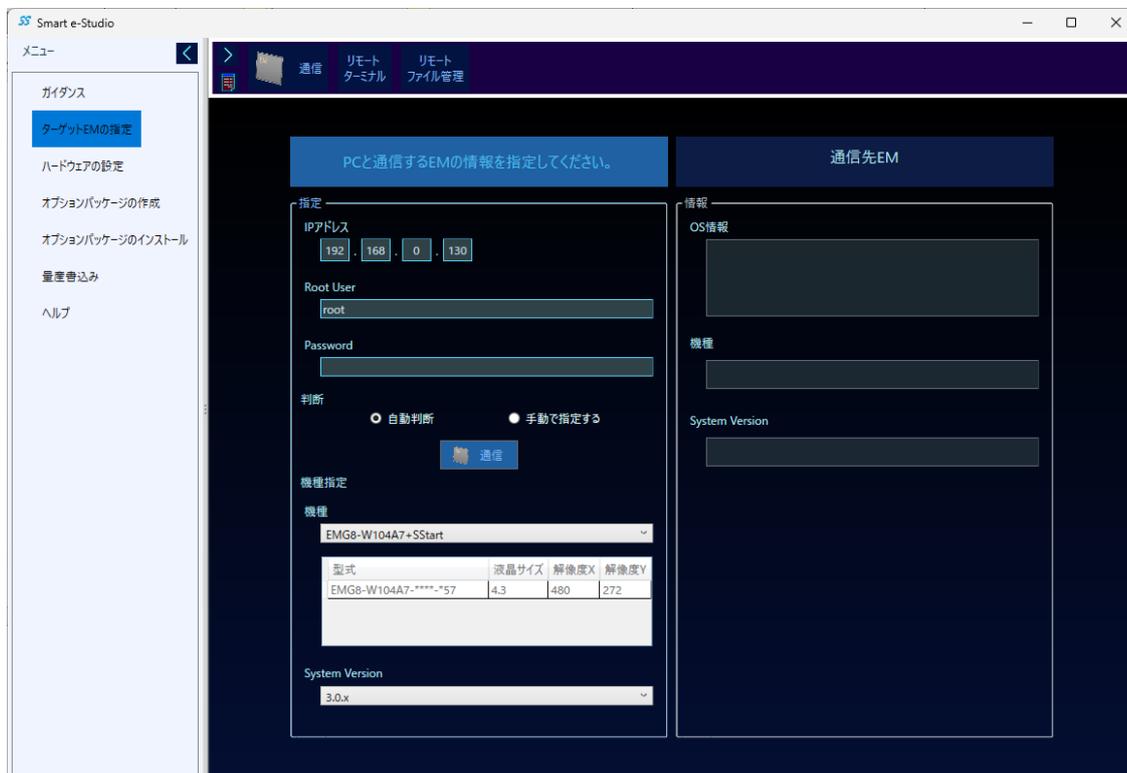
2.2 Smart e-Studioのインストール

「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。InfoSOSAビルダで作成した画面データをEMシリーズに転送するなど、EMシリーズのセットアップに使用します。

本書では、データの転送などに Smart e-Studio を使用することを前提に記載しておりますので、お使いの PC に Smart e-Studio をインストールしてください。

インストール方法は、別紙「Smart e-Studio 取扱説明書」を参照ください。

Smart e-Studio はEMシリーズ開発環境に同梱されおります。



2.3 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。

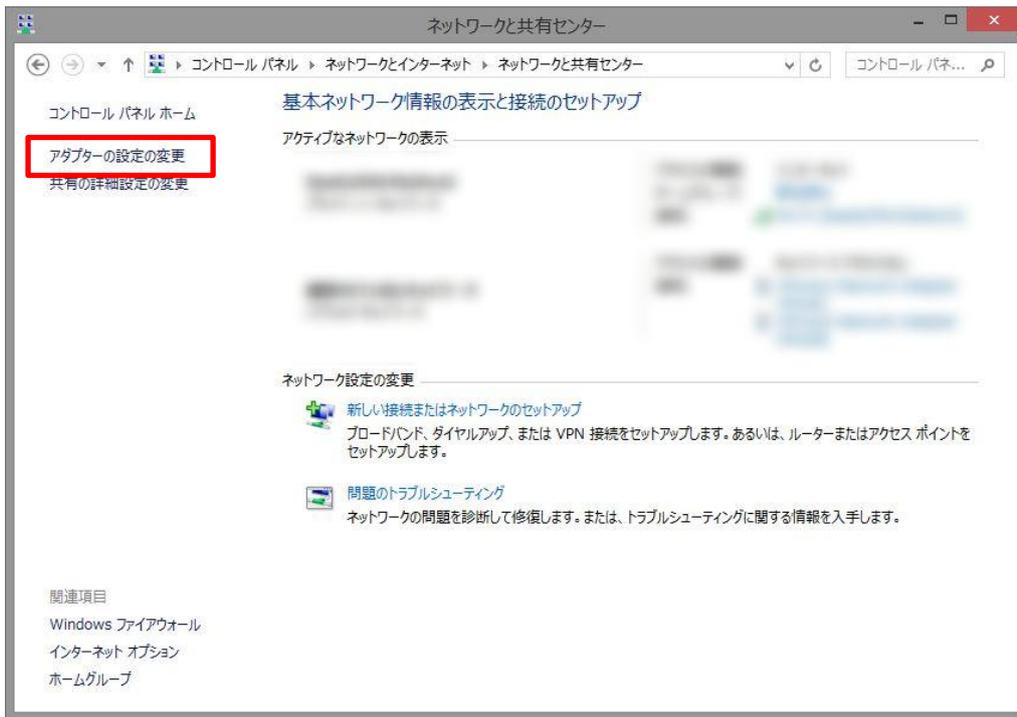
2.3.1 PC のネットワーク設定

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。

インターネット プロトコル バージョン 4 (TCP/IPv4)のプロパティ

全般

ネットワークでこの機能がサポートされている場合は、IP 設定を自動的に取得することができます。サポートされていない場合は、ネットワーク管理者に適切な IP 設定を問い合わせてください。

IP アドレスを自動的に取得する(O)

次の IP アドレスを使う(S):

IP アドレス(I): 192 . 168 . 0 . 100

サブネット マスク(U): 255 . 255 . 255 . 0

デフォルト ゲートウェイ(D): . . .

DNS サーバーのアドレスを自動的に取得する(B)

次の DNS サーバーのアドレスを使う(E):

優先 DNS サーバー(P): . . .

代替 DNS サーバー(A): . . .

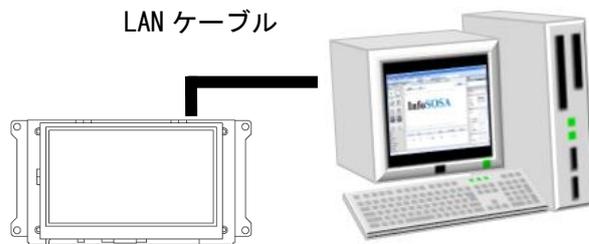
終了時に設定を検証する(L)

詳細設定(V)...

OK キャンセル

2.3.2 Smart e-Studio のターゲット EM 設定

1. EM シリーズ本体と PC を LAN ケーブルで接続します。

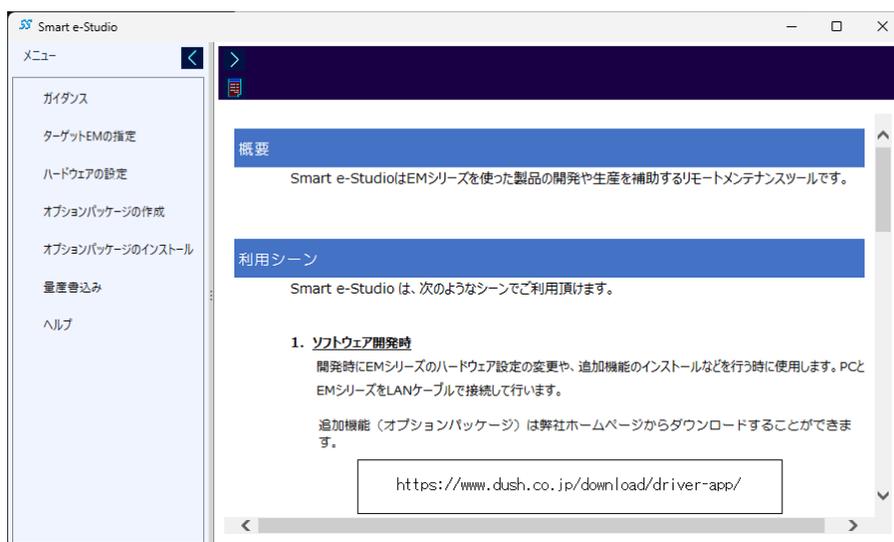


2. EM シリーズ本体と PC を LAN ケーブルで接続し、SSH プロトコルで接続できるようにネットワークの設定を行ってください。

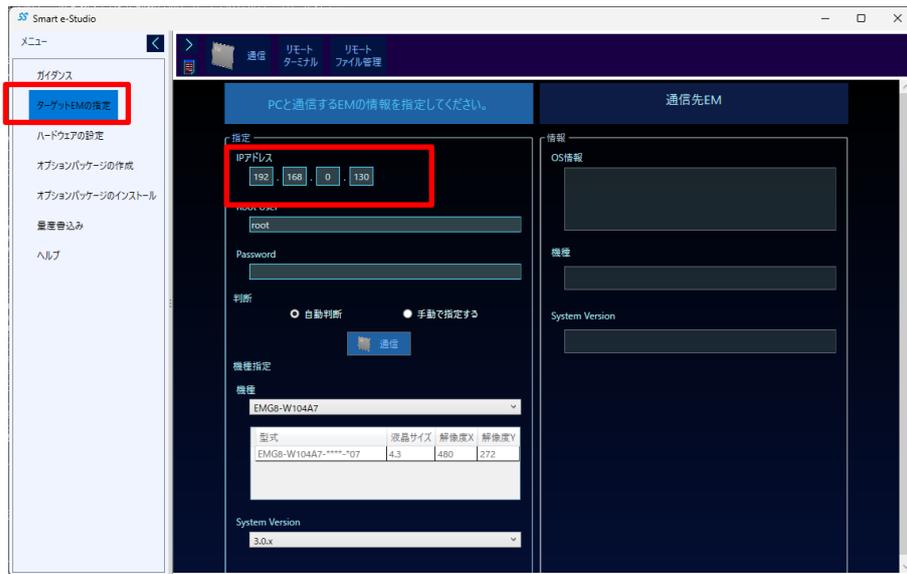
ネットワークの設定は、ネットワーク管理者にご確認お願い致します。

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 ※デフォルトインストール先

3. Smart e-Studio を起動してください。

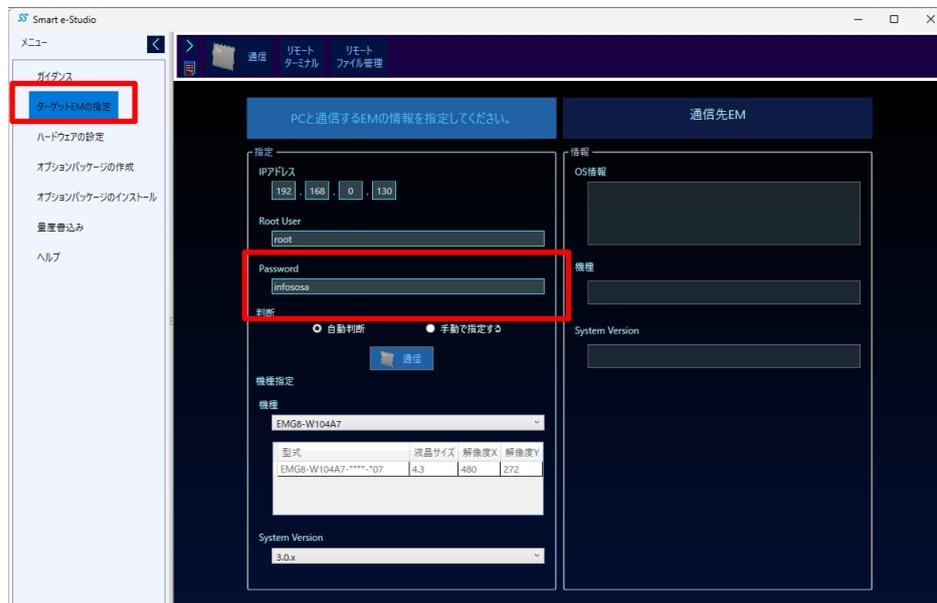


4. 「ターゲットEMの指定」ページを開き、「IPアドレス」項目にEMシリーズ本体のIPアドレスを入力してください。



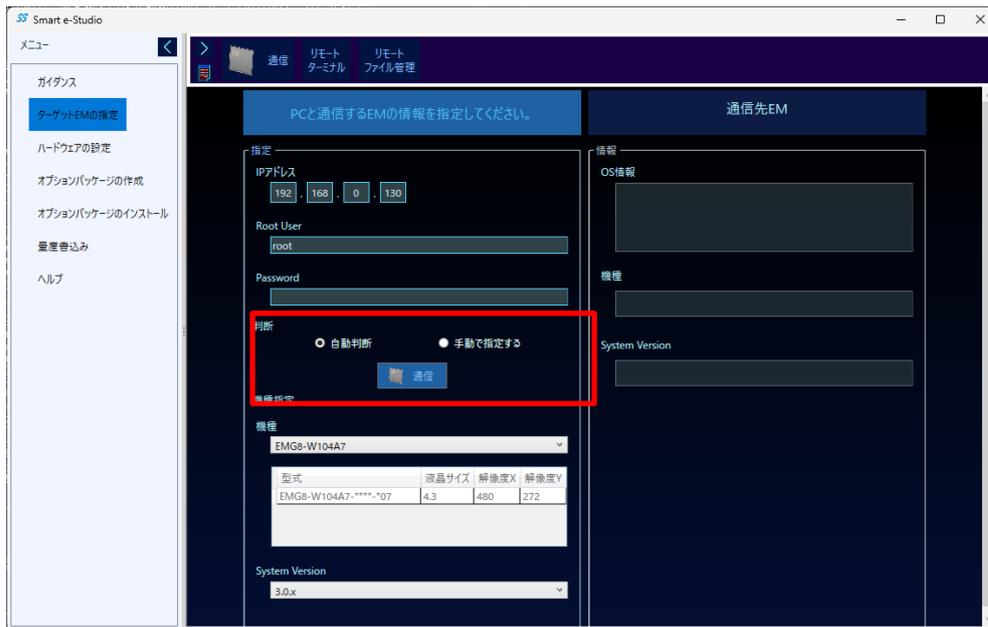
LAN ケーブルでの接続の場合、デフォルトは「192.168.0.130」になります。
 ※USB ケーブルでの接続や、IP アドレスを変更している場合は合わせてください。

5. 「Password」項目にEMシリーズ本体のrootパスワード「infososa」を入力してください。

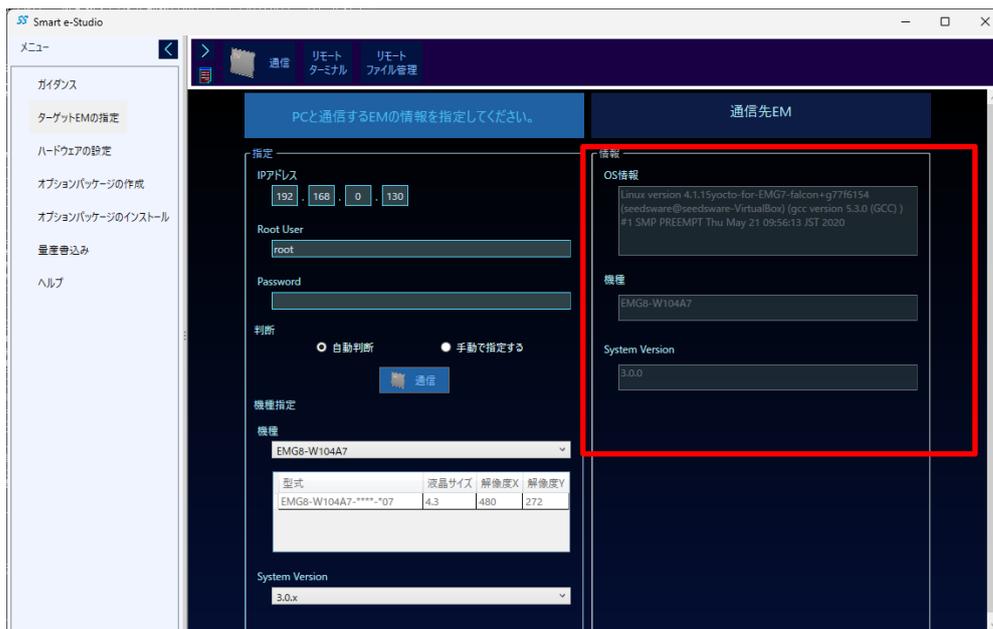


工場出荷状態ではrootパスワードは設定されていませんが、本手順でテンプレートパッケージを書き込むと「infososa」に設定されます。別のパスワードを設定する場合は合わせてください。

6. 「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。



【自動取得ができない場合】

手動取得が行えます。手順は別紙「Smart e-Studio 取扱説明書」を参照ください。

【通信エラーになる場合】

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。複数の EM シリーズと同時に接続することはできません。

2.4 テンプレートパッケージ書き込み

EM シリーズへの転送、設定変更を簡単に行えるように Smart e-Studio パッケージファイルのテンプレートをご用意しております。まず、テンプレートパッケージを書き込むことで、EM シリーズの設定を IS-APP のご利用に適した状態に自動的に更新します。

その後にテンプレートをカスタマイズすることで、開発中のお客様の作成したデータの書き込み、量産時の生産用データ (Smart e-Studio パッケージ) を作成できます。

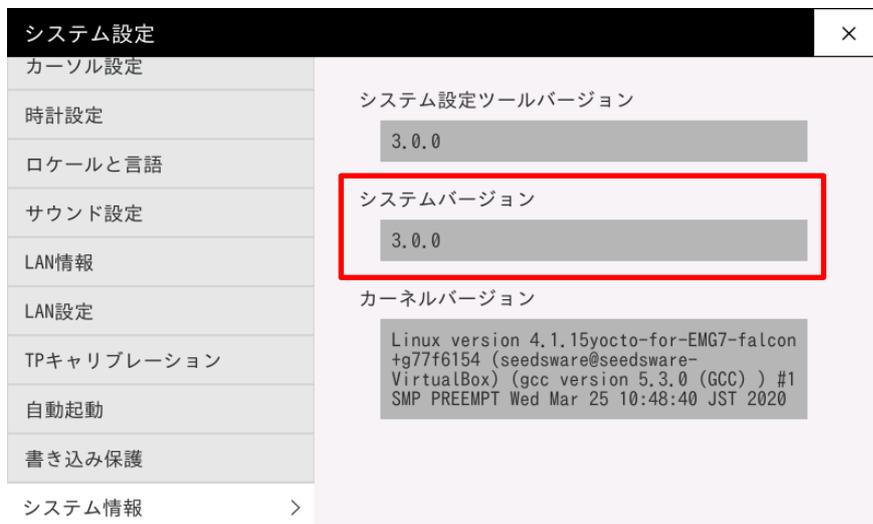
2.4.1 注意事項 1

EM シリーズ本体のシステムバージョンが以下である必要があります。
他のシステムバージョンの EM シリーズにはご使用できません。

対象システムバージョン	3.0.* 3.1.*
-------------	----------------

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。



2.4.2 注意事項 2

テンプレートパッケージを書き込むと EM シリーズ内の**ファイルの上書き/削除、システム設定の変更**が行われます。対象ファイルと同じパスにお客様が作成されたファイルがある場合はあらかじめバックアップを行ってください。詳細は「[2.11.5 テンプレートプロジェクト解説（高度な設定）](#)」を参照ください。

上書き/削除対象ファイル
/mnt/user/data/*
/mnt/user/isapp_run.sh
/usr/bin/is_app
/usr/bin/isapp_setting
/usr/lib/libisapi.so
/tmp/logo.bmp
/home/root/.config/lxsession/LXDE/autostart
/home/root/.config/lxsession/LXDE/autostart.org
/etc/init.d/user_mount.sh
/etc/init.d/user_mount.sh.org
/etc/rcS.d/S02banner.sh
/etc/rcS.d/S36udev-cache
/etc/udev-cache.tar.gz
/etc/rc5.d/S12rpcbind
/etc/rc5.d/S20bluetooth
/etc/rc5.d/S20ntpd
/etc/rc5.d/S22ofono
/etc/rc5.d/S70lighttpd
/etc/rc5.d/S90crond
/dev/mtd6 ※1
/mnt/user/resume.sh ※2
/mnt/user/suspend.sh ※2
/usr/bin/emsystem/SStart-core.sh ※2
/usr/bin/emsystem/SStart-core.sh.org ※2

※1 起動画面

※2 Smart e-Start 対応製品のみ

変更される設定
IS-APP/IS-API/IS-APP SETTING のインストール
IS-APP 画面データの転送
IS-APP 起動スクリプトの転送
起動画面の更新
輝度を「8」に設定
自動バックライト OFF を「無効」に設定
カーソル表示を「なし」に設定
有線 LAN (eth0) のモードを「STATIC(固定)」に設定
有線 LAN (eth0) の IP アドレスを「192.168.0.130」、サブネットマスクを「255.255.255.0」 デフォルトゲートウェイを「0.0.0.0」に設定
システムタッチ音を「なし」に設定
自動起動を「IS-APP」に設定
root パスワードを「infososa」に設定
Smart e-Start 機能を一時的に無効化 ※1
Smart e-Start 機能のレジュームポイントをプログラムで指定に設定 ※1
Smart e-Start 機能のサスペンド処理、レジューム処理を初期化 ※1
ユーザ領域 1、ユーザ領域 2 の書込保護の有効化 ※2
起動中の「Please wait:booting...」メッセージ表示を無効化
lxpanel (LXDE タスクバー)の起動停止 (スタートメニュー、LXDE タスクバーの表示が無効)
pcmanfm (PCManFileManager)の起動停止 (デスクトップ表示、壁紙、外部ストレージの自動マウントなどが無効)
udev-cache 処理を無効化
rpcbind daemon の起動を無効化
bluetooth daemon の起動を無効化
ntp daemon の起動を無効化
samba server の起動を無効化
oFono daemon の起動を無効化
lighttpd daemon の起動を無効化
cron daemon の起動を無効化

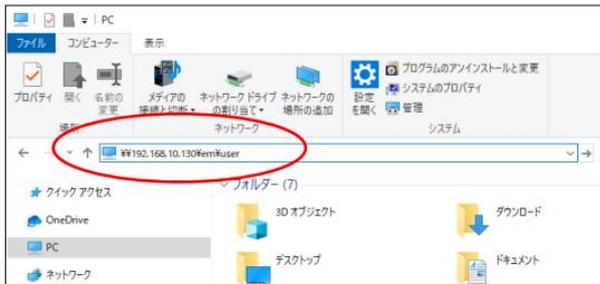
※1 Smart e-Start 対応製品のみ

※2 Smart e-Start 対応製品はユーザ領域 1 のみ

特にテンプレートパッケージを書き込むと以下の機能が停止しますのでご注意ください。

1. ファイル転送方法 (Samba)

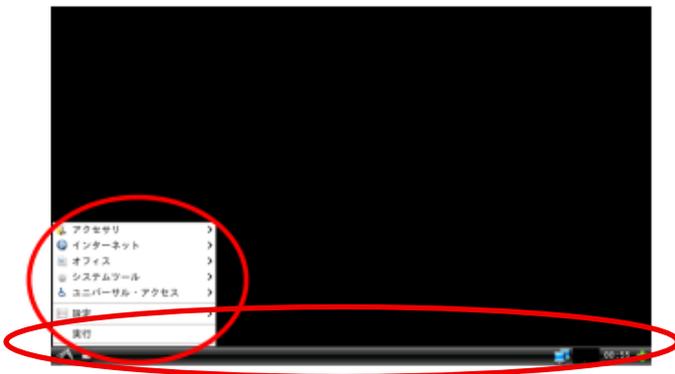
PC からエクスプローラーで EM とのファイル転送を行う機能



ファイル転送は複数の手段がある為、その内の1つである Samba サーバーの起動を停止することで起動時間短縮が見込めます。Samba サーバー停止時も Smart e-Studio を使用することでデータの書き込みを行います。

2. タスクバー、デスクトップ表示

開発用のスタートメニュー、タスクバー機能



開発用の機能を停止することで起動時間の短縮が見込め、起動時にタスクバーが表示されなくなります。その他、デスクトップ表示、壁紙、外部ストレージの自動マウントなどが無効になります。

3. Smart e-Start 対応製品について

Smart e-Start 機能（高速起動）はお客様のご使用方法に合わせて設定が必要になります。その為、テンプレートパッケージでは一時的に無効化する設定になっております。設定方法については、別紙「Smart e-Start マニュアル」を参照ください。

停止される機能を使用される場合は「[2.11.5 テンプレートプロジェクト解説（高度な設定）](#)」を参考にテンプレートパッケージをカスタマイズしてください。

2.4.3 書き込み手順

1. 「[2.3 EM シリーズ本体と PC との接続](#)」を参考に Smart e-Studio で EM シリーズ本体と通信が可能な状態にしてください。
2. テンプレートパッケージファイルを開いてください。

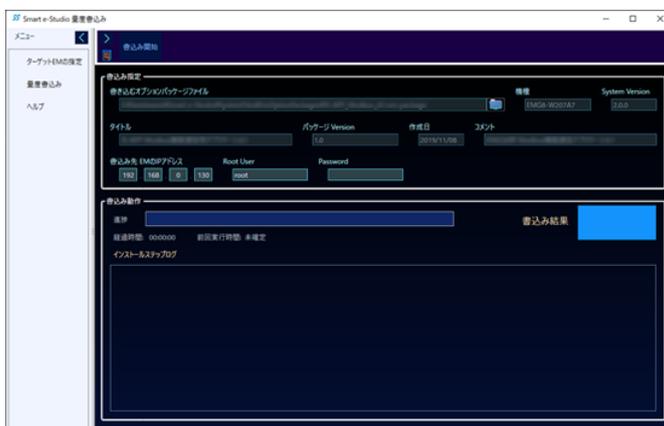
ファイルは InfoSOSA 開発キット内の以下のフォルダ以下にあります。

software¥EM シリーズ用 IS-APP 設定テンプレート

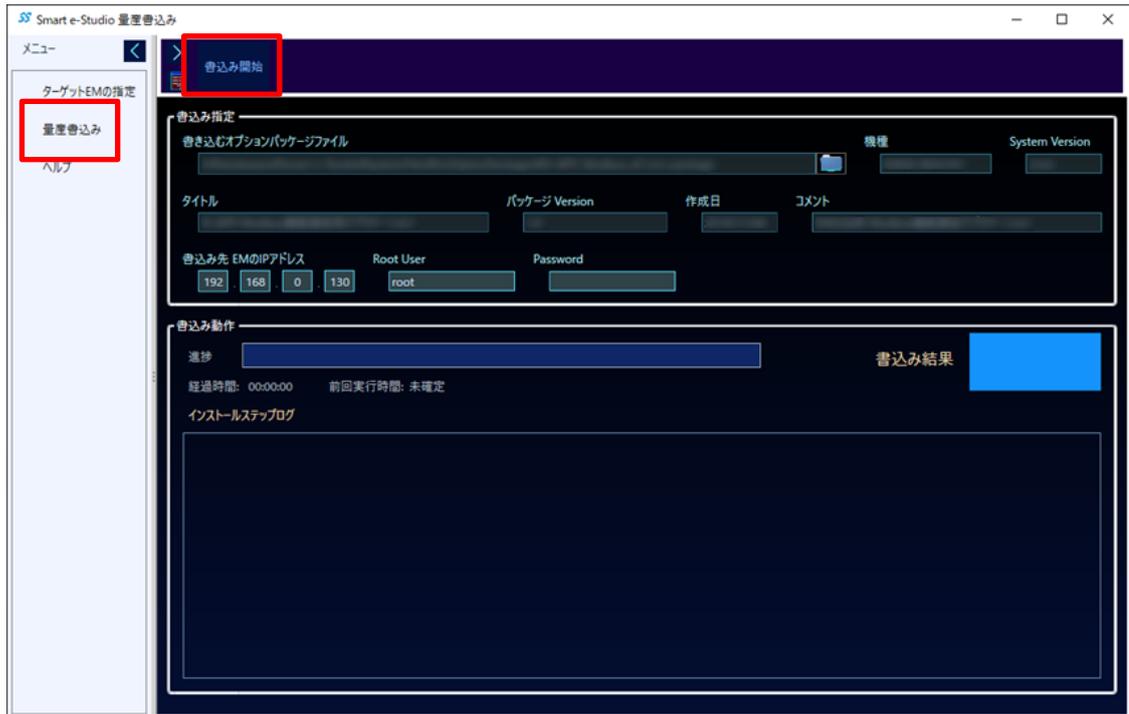
機種によって使用するファイルが異なります。テンプレートは ZIP 形式で圧縮されていますので、対象のファイルを任意の場所に展開してください。

型式	テンプレート
EM(G)8-W104A7-****-*07	標準製品¥EM(G)8-4.zip
EM(G)8-205A7-****-*07	標準製品¥EM(G)8-5.zip
EM(G)8-W207A7-****-*07	標準製品¥EM(G)8-7_EMP-7.zip
EM(G)8-W310A7-****-*07	標準製品¥EM(G)8-10.zip
EMP-W207A7-****-*07	標準製品¥EM(G)8-7_EMP-7.zip
EMG7-W207A8-****-*07	標準製品¥EMG7-7.zip
EMG7-310A8-****-*07	標準製品¥EMG7-10_12.zip
EMG7-312A8-****-*07	標準製品¥EMG7-10_12.zip
EM(G)8-W104A7-****-*57	Smart e-Start 対応製品¥EM(G)8-4-SS.zip
EM(G)8-205A7-****-*57	Smart e-Start 対応製品¥EM(G)8-5-SS.zip
EM(G)8-W207A7-****-*57	Smart e-Start 対応製品¥EM(G)8-7_EMP-7-SS.zip
EM(G)8-W310A7-****-*57	Smart e-Start 対応製品¥EM(G)8-10-SS.zip
EMP-W207A7-****-*57	Smart e-Start 対応製品¥EM(G)8-7_EMP-7-SS.zip

展開したフォルダ内の「EM_IS-APP_template_xxx.em-package」を開いてください。xxx 部分は製品によって異なります。em-package ファイルに関連付けが行われている Smart e-Studio が起動します。起動しない場合は Smart e-Studio を起動して「量産書き込み」ページから開いてください。



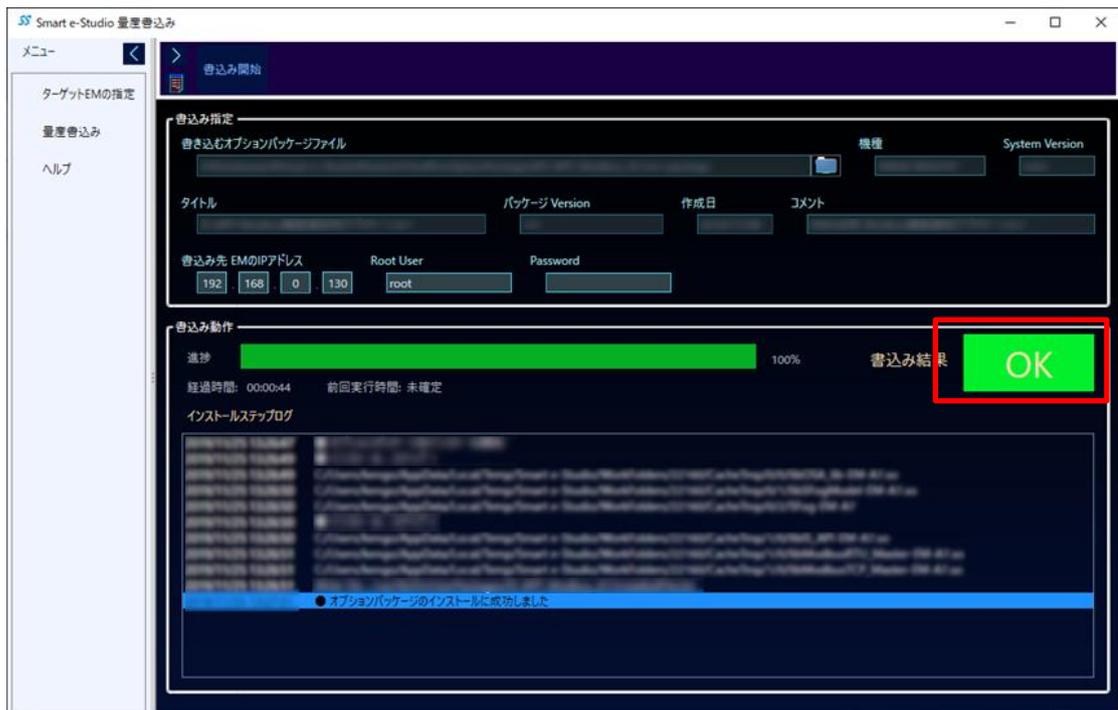
3. 「量産書き込み」ページを開き、「書き込み開始」を押してください。



4. 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。



書き込み結果に「OK」が表示されると完了です。



自動的に再起動して IS-APP (InfoSOSA アプリケーション) のサンプル画面が表示されます。

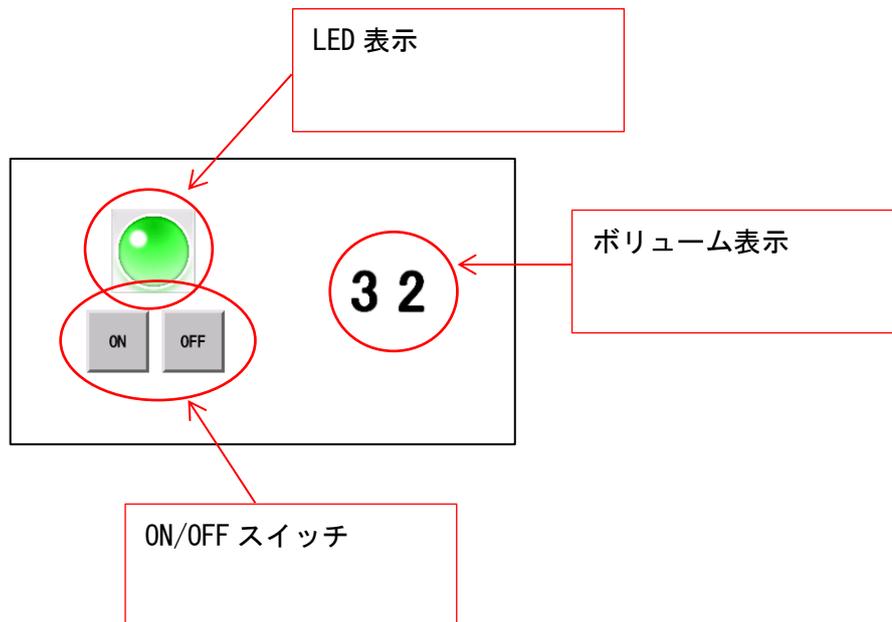


2.5 システム仕様作成

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-APP からの ON ボタンが押されたという通知を受けて、ボリューム表示の更新を開始する。
 IS-APP からの OFF ボタンが押されたという通知を受けて、ボリューム表示の更新を停止する。
 ボリューム表示の値は C++アプリケーション内でインクリメントを行い、IS-APP 側に反映する。

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。

2.6 InfoSOSAビルダの作成

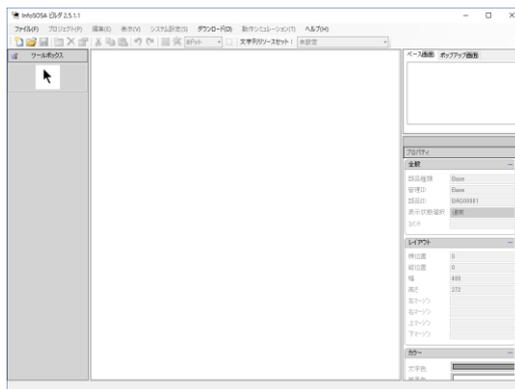
ここからは InfoSOSA ビルダを使用してプロジェクトの作成を行います。
次に進む前に PC(Windows)に InfoSOSA ビルダのインストールを行ってください。
インストール方法は「[2.1 InfoSOSA ビルダのインストール](#)」を参照ください。

MEMO

◆プロジェクトとは？

InfoSOSA ビルダで作成した画面設定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。

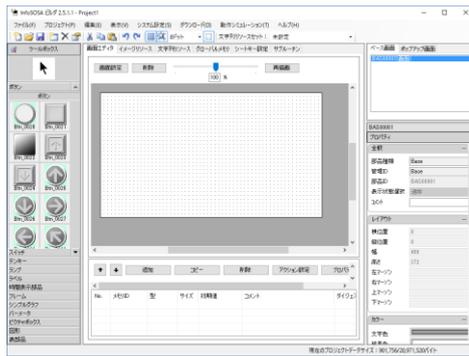
MEMO

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、お使いの製品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



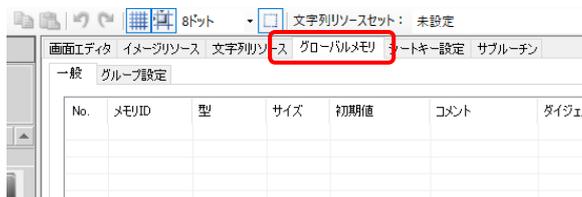
プロジェクトを作成すると左のような画面が表示されます。

続いて、画面を作成します。

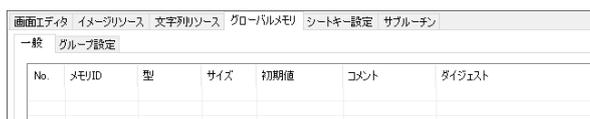
グローバルメモリの設定

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	プール	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。

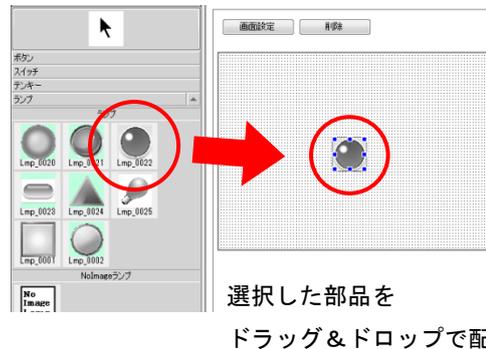
プロパティ	値	プロパティの意味
メモリ ID	GM_VOL01	メモリを識別する ID
型	バイト	メモリの種類
初期値	0	電源投入時の初期値
最小値	0	メモリの最小値
最大値	100	メモリの最大値
アンダーフロー	保持	メモリの最小値より小さな値が設定された場合の動作
オーバーフロー	保持	メモリの最大値より大きな値が設定された場合の動作

設定後のリストは以下ようになります。

画面エディタ						
イメージリソース						
文字列リソース						
グローバルメモリ						
シートキー設定						
サブルーチン						
一般						
グループ設定						
No.	メモリID	型	サイズ	初期値	コメント	ダイジェスト
0001	GM_LED01	ブール	-	0		
0002	GM_VOL01	バイト	-	0		0,保持,100,保持

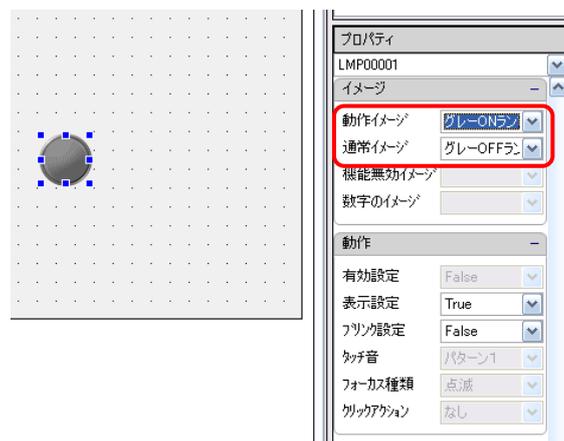
ランプ部品の設定

ランプの配置



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



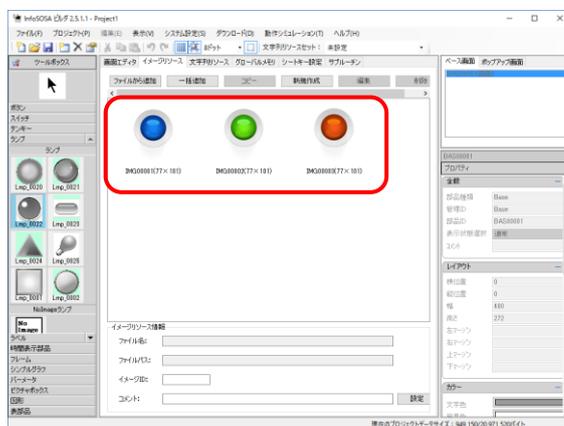
配置したランプを選択した状態で、ビルダ右部のプロパティエリアから「イメージ」の項目を設定します。この「動作イメージ」と「通常イメージ」を設定すると、ランプのイメージが変更できます。

「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

リンクデータ	
メモリ種類	グローバルメモリ
メモリID	GM_LED01
関係リンクキー	

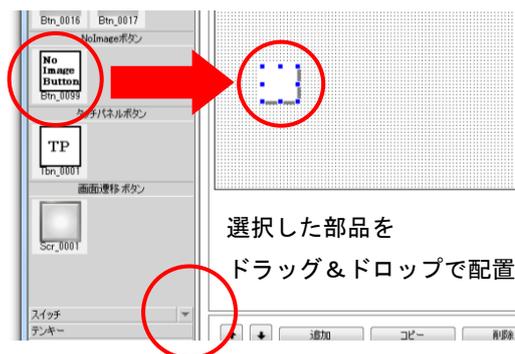
ランプとグローバルメモリの関連付けを行います。リンク設定を行うと指定したグローバルメモリの値が0の時に「通常イメージ」、1の時に「動作イメージ」が表示されるようになります。

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

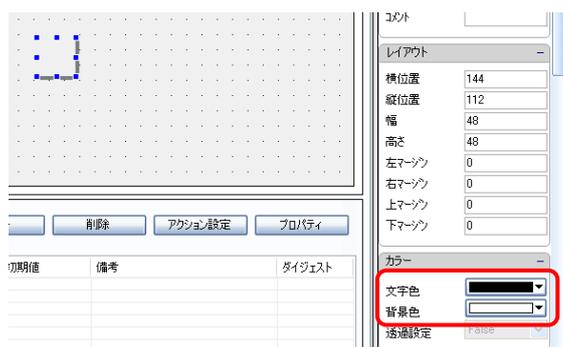
ボタン部品の設定

ボタンの配置



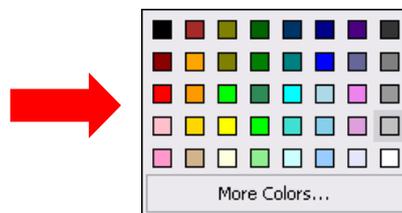
ビルダ左のツールボックスから「ボタン」を選択し「Btn_0099」を、画面にドラッグ&ドロップしてください。表示されていない場合は、▼でスクロールします。

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	
文字列	ON
水平位置	中央
垂直位置	中央
文字サイズ	16

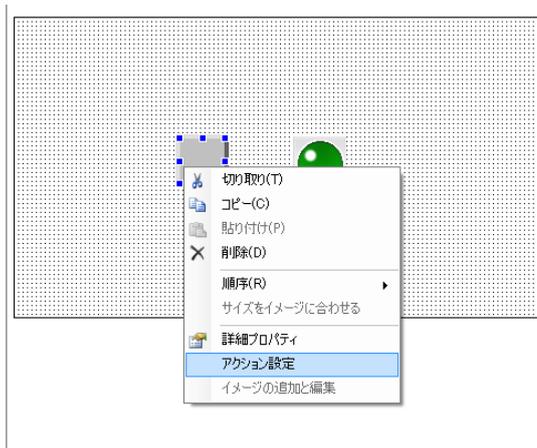
文字列プロパティを変更することで、ボタン上に文字を表示することができます。

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。

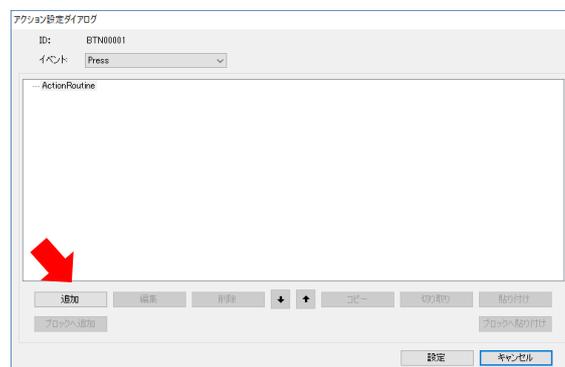


アクションを設定するには、画面に配置したボタンを右クリックし、「アクション設定」をクリックします。

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。

アクション更新ダイアログ

アクション

アクショングループ 数値演算

アクション 値設定

パラメータ

メモリ種類 グローバルメモリ

メモリID(数値型) GM_LED01

設定値 1

設定 キャンセル

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

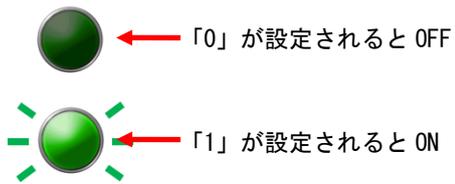
- ・アクショングループ
⇒使用したいアクションのカテゴリを選択します。
- ・アクション
⇒使用したいアクションを選択します。

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。

アクション更新ダイアログ

アクション
 アクショングループ: 数値演算
 アクション: 値設定

パラメータ
 メモリ種類: グローバルメモリ
 メモリID(数値型): GM_LED01
 設定値: 1

ここで設定する値を指定します。
 今回はランプを点灯させるので、値は「1」を指定します。

アクショングループは「数値演算」を選択します。

アクションは「値設定」を選択します。

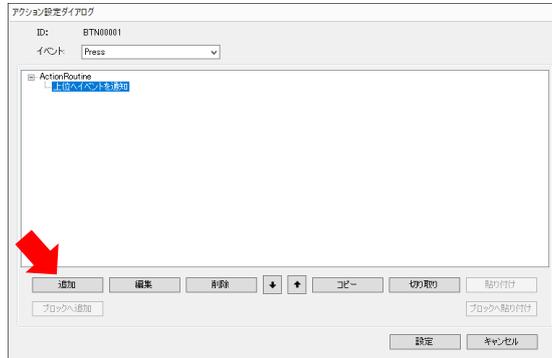
メモリ種類は「グローバルメモリ」を選択します。

ランプとリンクされている「GM_LED01」を選択します

設定 キャンセル

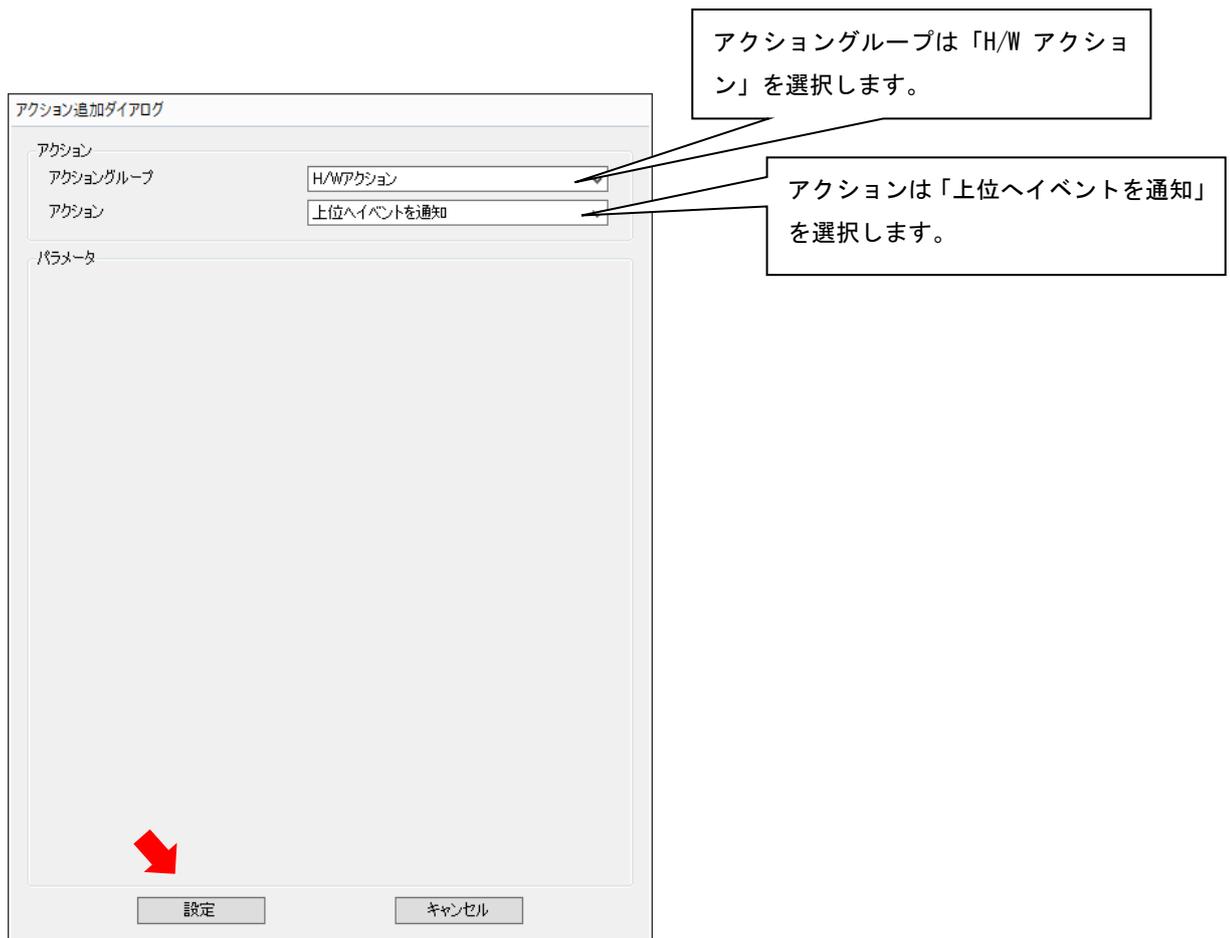
アクション設定②

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。



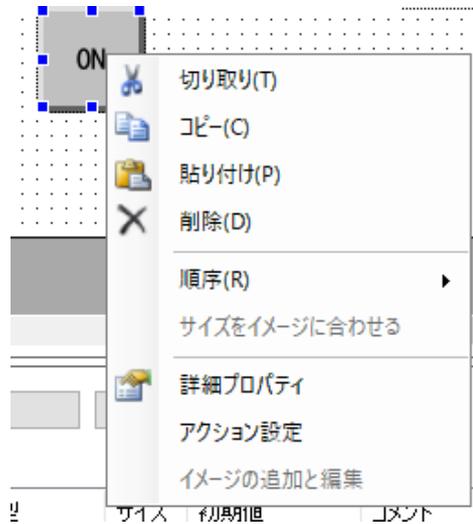
左下の「追加」を押してアクション追加ダイアログを開きます。

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。

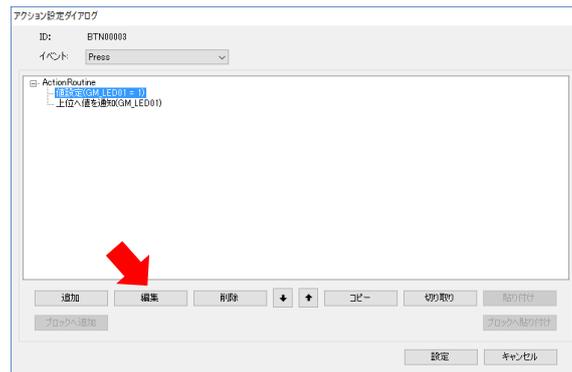
表示文字を「OFF」に変更

文字列	
文字列	OFF
水平位置	中央
垂直位置	中央
文字サイズ	16

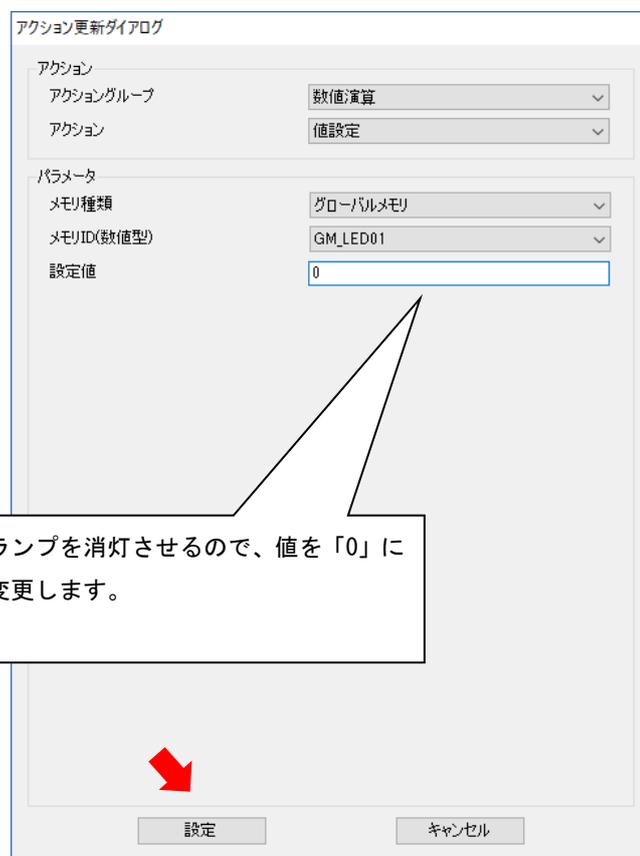
文字列プロパティを OFF

・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

リンクデータ	
メモリ種類	グローバルメモリ
メモリID	GM_VOL01
関係テンキー	

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

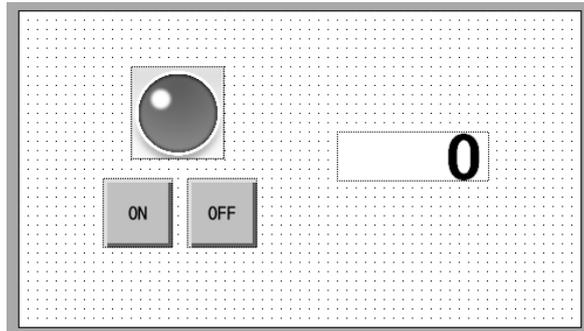
データ	
値	0
表示桁数	3

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。

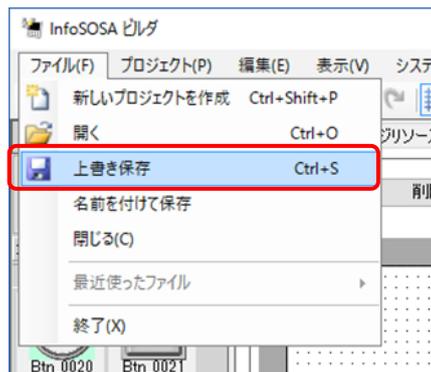
- 表示桁数： 3

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。

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。

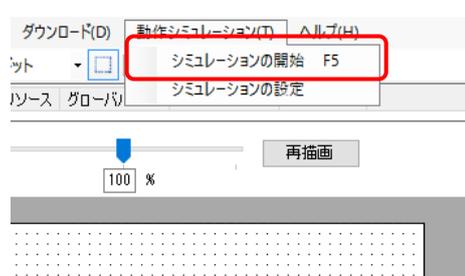


※ 別名でプロジェクトを保存する場合は「名前を付けて保存」をクリックします。

2.7 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。

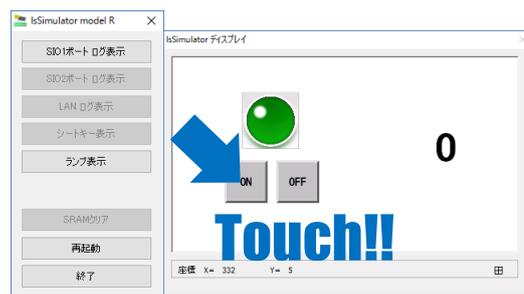
シミュレータの起動



作成したプロジェクトの動作をPC上で確認します。

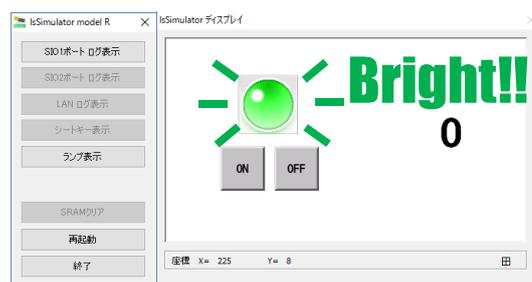
シミュレータの起動は、「動作シミュレーション」メニューの「シミュレーションの開始」から行います。

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。

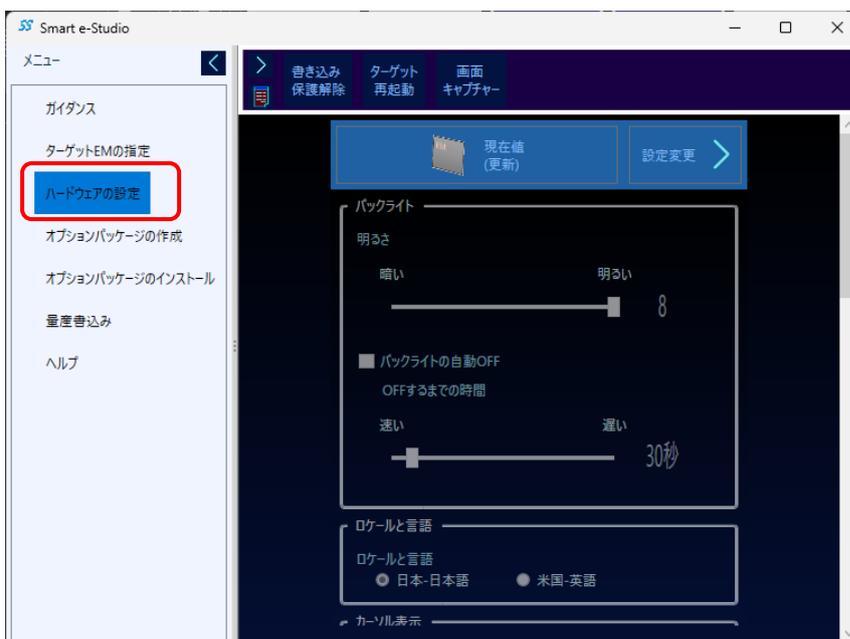
2.8 IS-APPの通信設定

IS-APP 設定ツール (ISAPP SETTING) を使用して、以下の設定を行います。

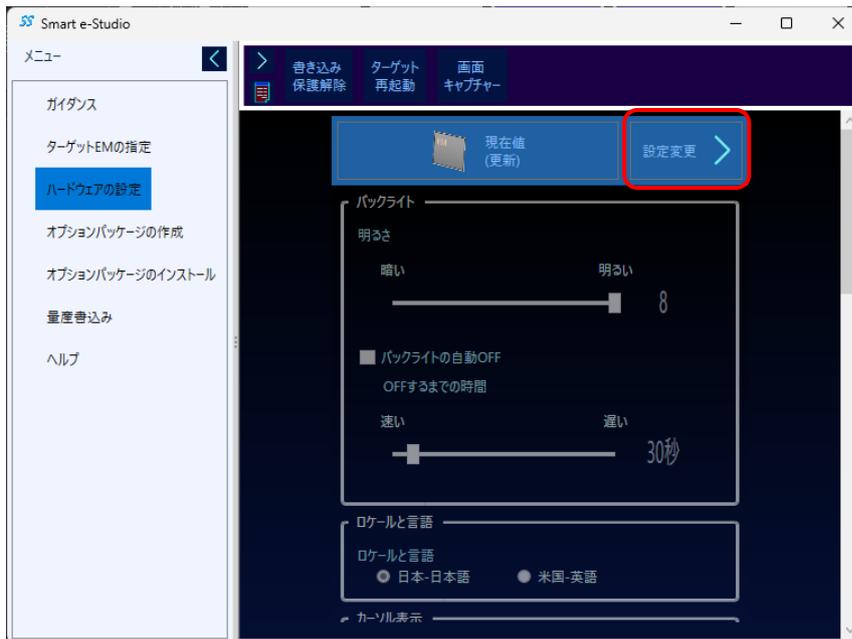
項目	内容
画面データ	IS-APP で表示する画面データフォルダを指定します。 通常は変更の必要はありません。
通信先(Main)	IS-APP の上位通信先選択、並びに通信設定を行います。
通信先(Sub1)	
通信先(Sub2)	
通信プロトコル	IS-APP が外部機器と上位通信を行う時のプロトコルを設定します。
Modbus	IS-APP の Modbus 通信関連の設定を行います。
IS-APP ブザー	IS-APP のブザー機能について設定を行います。

IS-APP 起動スクリプトの作成

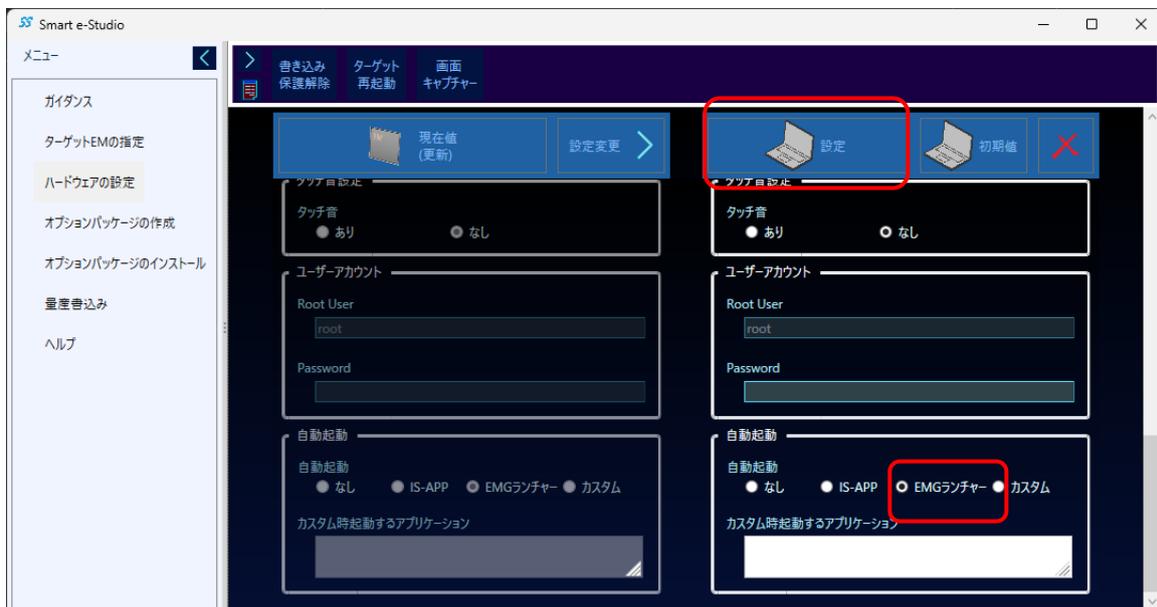
- 「[2.3 EM シリーズ本体と PC との接続](#)」を参考に Smart e-Studio で EM シリーズ本体と通信が可能な状態にしてください。
- Smart e-Studio を起動して「ハードウェアの設定」ページを表示します。



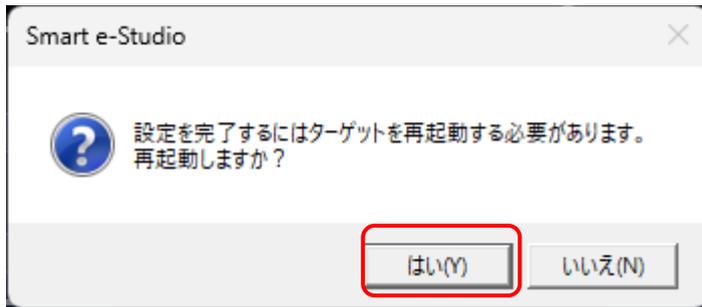
3. 「設定変更」ボタンを押します。



4. 「自動起動設定」を「EMGランチャー」に変更後に「設定」ボタンを押します。



5. 「はい」ボタンを押してください。EMシリーズが再起動します。



6. 再起動するとEMGランチャーが起動します。EMシリーズ本体を操作して、EMGランチャーの2ページ目から「IS-APP設定ツール」を起動します。

[1 ページ目]



[2 ページ目]



7. 以下のように設定してください。

項目	内容	
画面データ	/mnt/user/data	
通信先(Main)	IS-API	ポート番号：51111
通信先(Sub1)	なし	
通信先(Sub2)	なし	
通信プロトコル	InfoSOSA 標準プロトコル	通知方法：即時通知
Modbus	無効	
IS-APP プザー	有効	

上記設定では、IS-APP は C++アプリケーションと内部通信する設定になります。
IS-APP とマイコンボードを接続するには、本設定を変更する必要があります。
詳しくは「[3.3 ISAPP SETTING](#)」を参照ください。

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。



IS-APP 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
```

IS-APP が自動起動する状態に戻す場合は、再度本手順を行い 4. で「自動起動設定」を「IS-APP」に戻して「設定」ボタンを押してください。



IS-APP 起動スクリプトの取り出し

続いて EM シリーズ本体で作成した IS-APP 起動スクリプトファイルを PC に転送します。

お使いの PC に SFTP で接続可能な FTP クライアントソフトウェアをインストールしてください。ここでは、WinSCP を使用しています。

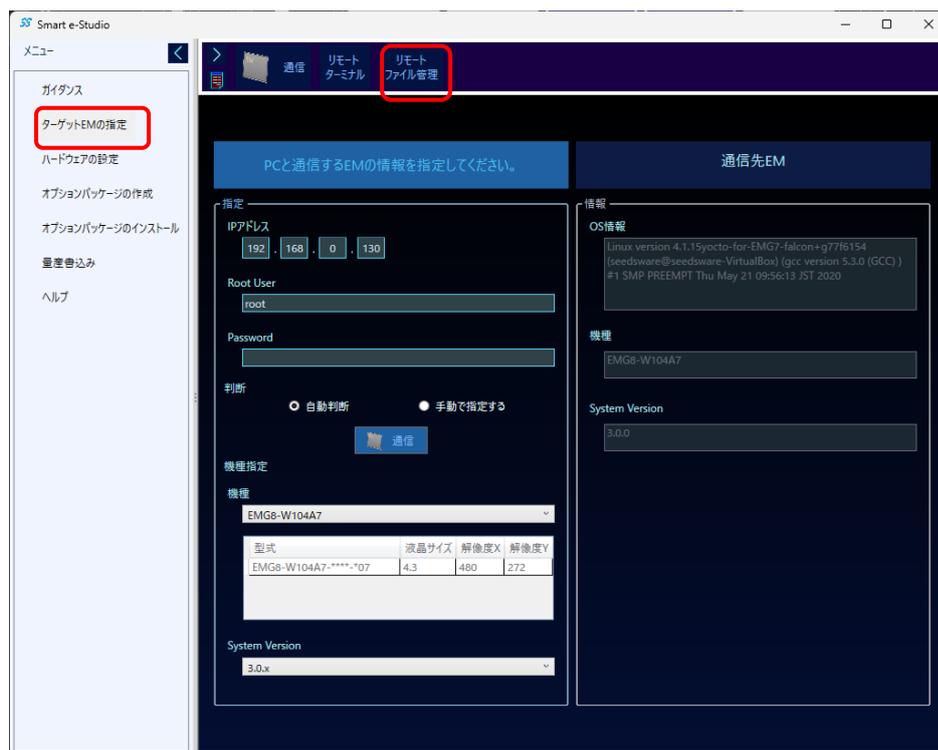
WinSCP の公開サイト

<https://winscp.net/eng/docs/lang:jp>

免責事項

WinSCP は「Martin Přikryl」により提供されているオープンソースソフトウェアです。弊社では WinSCP に関して、動作保証、操作サポート、損害の補償など一切の責任は負いません。

1. Smart e-Studio の「ターゲット EM の設定」ページを表示して「リモートファイル管理」を押します。



デフォルトでは、以下のアプリケーションを実行します。

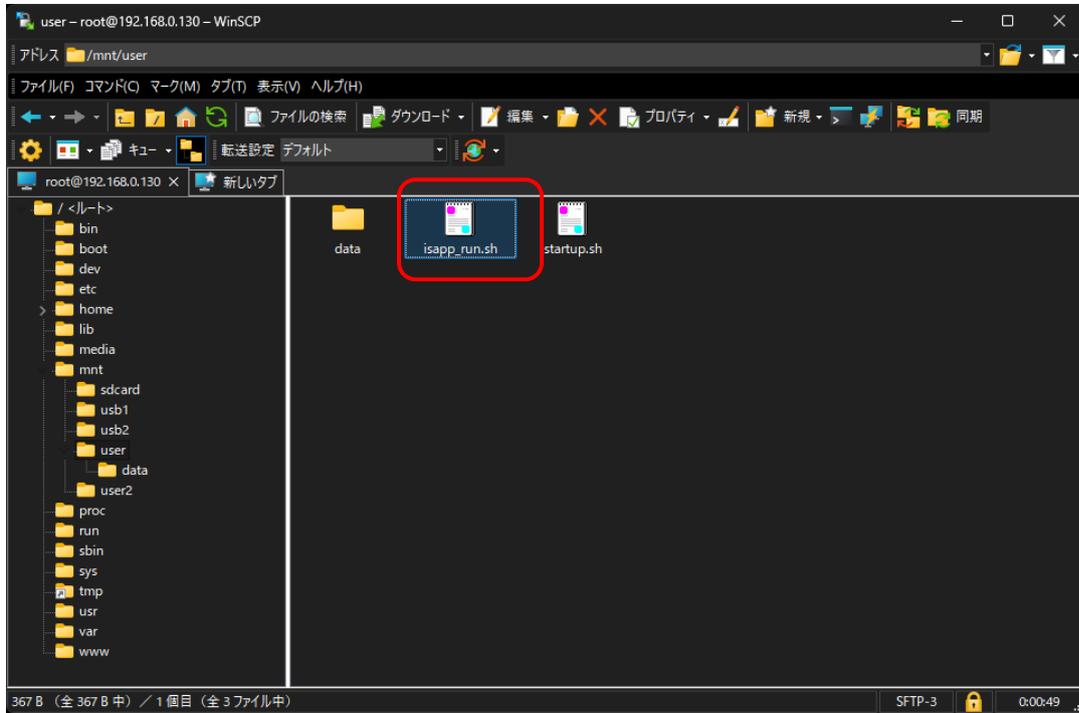
C:¥Program Files (x86)¥WinSCP¥WinSCP.exe

お使いの PC に WinSCP がインストールされていない、または他の場所にインストールされている場合は、エラーが表示されます。アイコンを右クリックすることで、起動するアプリケーション、起動オプションを変更することが可能です。

2. IS-APP 起動スクリプトを PC に転送してください。

IS-APP 起動スクリプト

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



2.9 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 環境)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。

IS-API ライブラリは、本項目で紹介していない機能もございます。詳細は「[3.2 IS-API 詳細仕様](#)」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>
#include <unistd.h>

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

CIsApi* pIsApi = NULL;
bool run = false;

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
int vol = 0;
CIsComVariant var;
std::string propertyID="@GLBMEM.GM_VOL01.VALUE";
while(true)
{
    if(run){
        vol++;
        var.SetValue(vol);
        pIsApi->setProperty(&propertyID, var);
        std::cout << "setProperty(" << var.GetValue() << ")" << std::endl;
        if(vol>=99) vol=0;
    }
    usleep(100000); //100ms
}

// [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;
    run = true;
}

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

```

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] CIsApi 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] CIsApi 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 CIsApi 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。
- ※ IS-API のメソッドはスレッドセーフではありません。コールバック関数内で実行しないでください。
- ※ 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。

■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 ※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

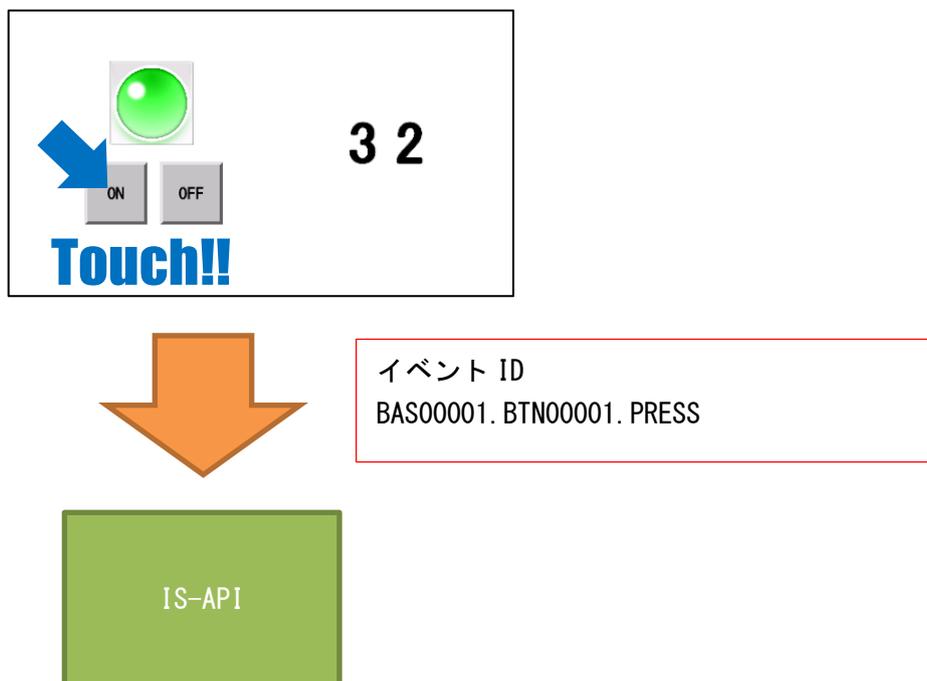
false: 失敗

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、ボタンを離れた時には「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、離れ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

ON ボタンが押された場合に、インクリメントした値を InfoSOSA のメモリにセットして画面上の数字を更新します。

※IS-API のメソッドはスレッドセーフではありません。1つのスレッドから実行してください。

■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 ※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 ※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、動作フラグを切り替えています。

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 環境) を操作します。

ソースファイルを保存したディレクトリに移動

Linux 環境のターミナル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。

事前に関係環境にツールチェーンのインストールが必要です。ツールチェー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
EM8-***A7-****-**7	IS-APP-A7 用スクリプト ※1
EMG8-***A7-****-**7	IS-APP-A7 用スクリプト ※1
EMP-***A7-****-**7	IS-APP-A7 用スクリプト ※1
EMG7-***A8-****-**7	IS-APP-A8 用スクリプト ※2

※2 IS-APP-A7 用スクリプト

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

※1 IS-APP-A8 用スクリプト

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

ツールチェーンのインストール先を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」 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 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
--------	----------

2.10 C++アプリケーションの起動設定



IS-APP 起動時に C++アプリケーションを起動する方法を記載します。

グローバルメモリの設定

InfoSOSA ビルダで作成したプロジェクトを開きます。

メモリの作成



「グローバルメモリ」タブを選択します。

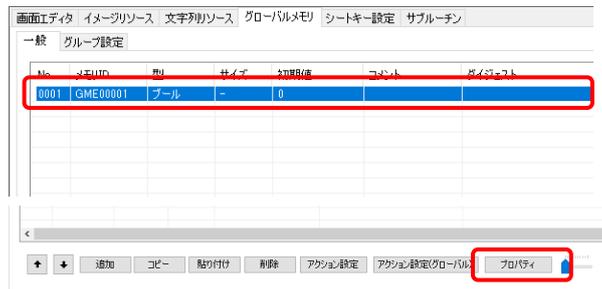


「追加」ボタンをクリックします。

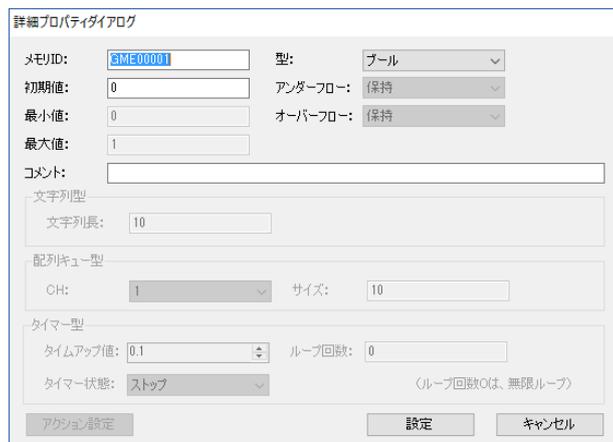


メモリが作成されました。

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	S_CMD	メモリを識別するID
型	文字列	メモリの種類
文字列長	64※	コマンドの最大文字数
初期値	(実行するコマンド)※	実行するコマンド

※お客様の環境に合わせて変更してください。

例：C++アプリケーションのファイル名が tutorial で/mnt/user/に配置した場合
/mnt/user/tutorial

2つ目のメモリの作成

同様の手順でメモリを作成します。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	S_PID	メモリを識別する ID
型	ダブルワード	メモリの種類

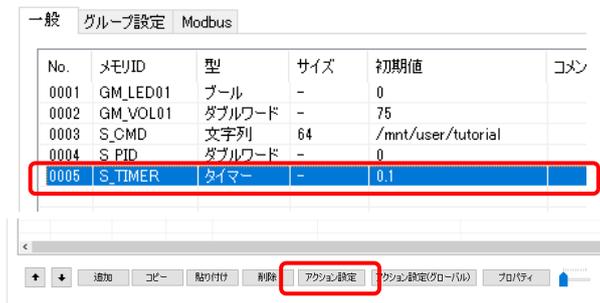
3つ目のメモリの作成

同様の手順でメモリを作成します。3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	S_TIMER	メモリを識別する ID
型	タイマー	メモリの種類
タイムアップ値(秒)	0.1※	アクションが実行されるまでの秒数
ループ回数	1	繰り返し回数
タイマー状態	スタート	タイマー動作の初期状態

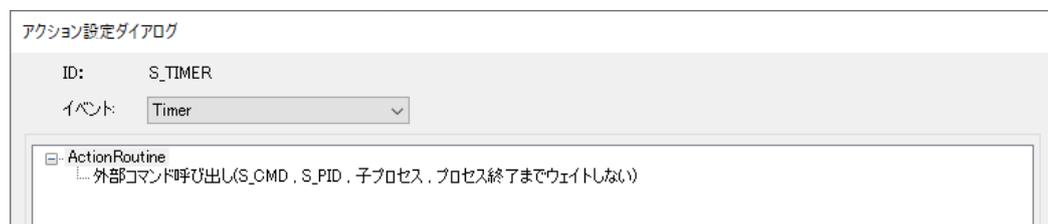
※C++アプリケーションの起動を遅延させる場合は値を調整してください。(単位：秒)

タイマー型メモリ アクション設定



「S_TIMER」メモリを選択して、「アクション設定」ボタンをクリックしてください。

以下のようにアクションを設定してください。



項目	設定
アクショングループ	外部コマンド
アクション	外部コマンド呼び出し
実行コマンドメモリ種類	グローバルメモリ
実行コマンドメモリ ID	S_CMD
PID 格納先種類	グローバルメモリ
PID 格納先メモリ ID	S_PID
外部コマンドで実行したプロセスとの関係	子プロセス
外部コマンド実行中の IS アプリの動作	プロセス終了までウェイトしない

2.11 テンプレートパッケージのカスタマイズ

2.11.1 IS-APP 転送用データ作成

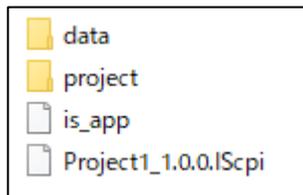
InfoSOSA ビルダを操作します。InfoSOSA ビルダで作成したプロジェクトは、以下の手順で転送用データに変換します。



「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） ※本手順では使用しません。
is_app	IS-APP の実行ファイル ※本手順では使用しません。
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]_[ユーザーバージョン].IScpi ※本手順では使用しません。

本手順では「data」フォルダのみ使用します。

2.11.2 起動画面用画像ファイル作成

電源を入れた時に最初に表示される起動画面用画像ファイルを作成します。
以下の形式でご用意してください。

ファイル名	logo.bmp
形式	非圧縮 24bit ビットマップ
サイズ	実機の解像度と同じ

2.11.3 Smart e-Studio パッケージの作成

テンプレート内のデータを差し替えて、お客様が作成したデータを EM シリーズに書き込む Smart e-Studio パッケージを作成します。

テンプレートは以下の構成になっています。

フォルダ/ファイル名		説明
_system		システムファイル ※変更の必要はありません
USER_DATA	data	InfoSOSA ビルダで作成した画面データ
	is_app	IS-APP アプリケーション (IS-APP)
	isapp_run.sh	IS-APP 起動スクリプト
	isapp_setting	IS-APP 設定ツール (IS-APP SETTING)
	libisapi.so	IS-API ライブラリ (IS-API)
	logo.bmp	起動画面用ファイル (ビットマップ画像ファイル)
EM_IS-APP_template_xxx.em-package ※		Smart e-Studio パッケージ (転送用ファイル)
EM_IS-APP_template_xxx.em-project ※		Smart e-Studio プロジェクト (編集用ファイル)

※ xxx 部分は製品によって異なります。以下_xxx 部分を省略して記載します。

1. テンプレートを任意の場所にコピーして作業してください。
2. 「USER_DATA\data」フォルダを削除してください。
3. 「[2.11.1 IS-APP 転送用データ作成](#)」で作成した「data」フォルダを「USER_DATA」フォルダ内にコピーしてください。※必ず古い data フォルダは削除した後にコピーしてください。
4. 「[2.8 IS-APP の通信設定](#)」で作成して PC に取り出した「isapp_run.sh」ファイルを「USER_DATA」フォルダ内にコピーして上書きしてください。

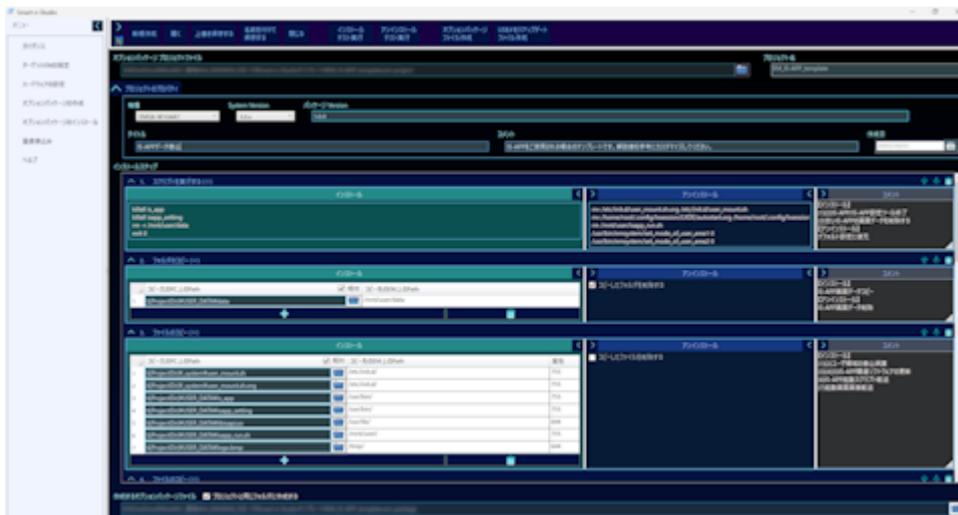
5. お使いの InfoSOSA 開発キットをバージョンアップした場合は、「is_app」「libisapi.so」「isapp_setting」の更新が必要になります。

InfoSOSA 開発キットから「USER_DATA」フォルダ内にコピーして上書きしてください。

「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
EM8-***A7-****-**7	software¥IS-APP¥IS-APP-A7¥is_app
EMG8-***A7-****-**7	software¥IS-API¥Library¥IS-APP-A7¥libisapi.so
EMP-***A7-****-**7	software¥IS-APP_SETTING¥IS-APP-A7¥isapp_setting
EMG7-***A8-****-**7	software¥IS-APP¥IS-APP-A8¥is_app software¥IS-API¥Library¥IS-APP-A8¥libisapi.so software¥IS-APP_SETTING¥IS-APP-A8¥ isapp_setting

6. 「[2.11.2 起動画面用画像ファイル作成](#)」で作成した「logo.bmp」を「USER_DATA」フォルダ内にコピーして上書きしてください。
7. 「EM_IS-APP_template.em-project」を開いてください。



8. インストールステップ「5. ハードウェアの設定」からバックライト、LAN 設定、root パスワードを設定してください。



デフォルトでは以下に設定されます。必要に応じて変更してください。
大項目のチェックを外すとインストール前の設定を引き継ぎます。

大項目	小項目	設定値
✓ バックライト	明るさ(輝度)	8(もっとも明るい)
	バックライトの自動 OFF 機能	有効
	OFF するまでの時間	-
ロケールと言語	ロケールと言語	(インストール前の設定を引き継ぐ)
✓ カーソル表示	カーソル表示	なし
✓ 有線 LAN 設定	有線 LAN を使用する	有効
	モード	STATIC(固定)
	IP アドレス	192.168.0.130 ※1 ※2
	サブネットマスク	255.255.255.0
	デフォルトゲートウェイ	0.0.0.0
✓ タッチ音設定	タッチ音	なし
✓ ユーザアカウント	root Password	infososa ※3
✓ 自動起動	自動起動	IS-APP
✓ Smart e-Start ※4	Smart e-Start 機能	無効
	レジュームポイント	プログラムで指定

※1 USB-Ether(usb0)で使用している為、「192.168.10.*」は使用できません。

※2 IP アドレスを変更した場合、Smart e-Studio の「ターゲット EM の指定」ページの IP アドレスも合わせる必要があります。

※3 root パスワードを変更した場合、Smart e-Studio の「ターゲット EM の指定」ページの Password も合わせる必要があります。

※4 Smart e-Start 対応製品のみ

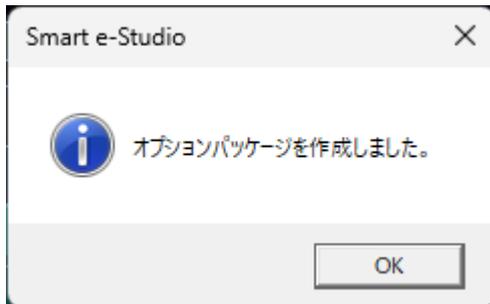
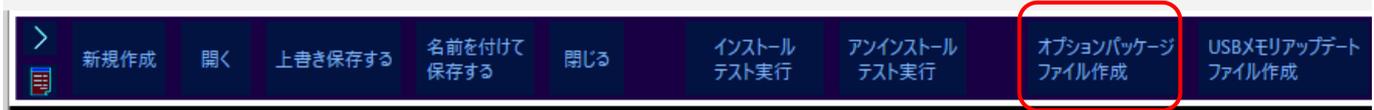
9. 必要に応じて以下の項目は変更可能です。お客様でのデータの管理にご利用ください。

項目	説明
パッケージ Version	用途自由です。任意の文字列を記入することができます。
タイトル	用途自由です。任意の文字列を記入することができます。
コメント	用途自由です。任意の文字列を記入することができます。
作成日	用途自由です。任意の日付を設定できます。

「上書き保存する」または「名前を付けて保存する」から保存してください。



10. 「オプションパッケージファイル作成」を押してください。



「EM_IS-APP_template.em-package」が更新されました。

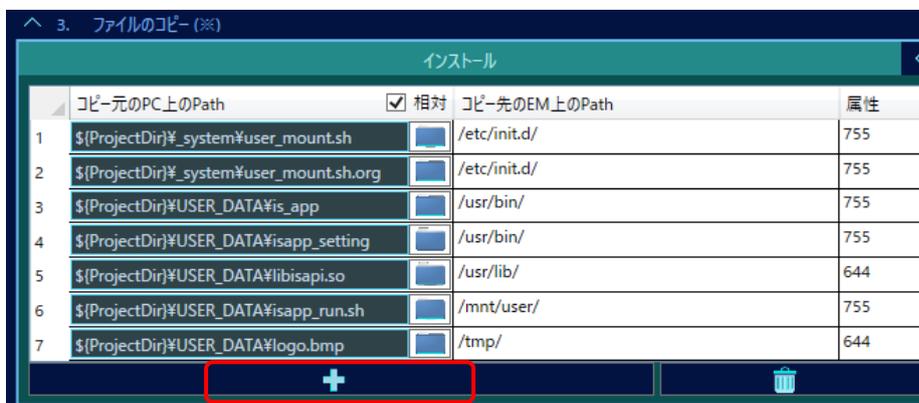
InfoSOSA ビルダのみで画面を作成された場合は以上で完了です。「[2.12 Smart e-Studio パッケージの書き込み](#)」に進んでください。

C++アプリケーションを追加したり、お客様独自にシステム設定/Linux をカスタマイズする場合は、「[2.11.4 Smart e-Studio パッケージの作成 \(高度な設定\)](#)」に進んでください。

2.11.4 Smart e-Studio パッケージの作成(高度な設定)



1. 「[2.9 C++アプリケーションの作成](#)」で作成した「tutorial」を「USER_DATA」フォルダ内にコピーしてください。
2. 「EM_IS-APP_template.em-project」を開いてください。
3. 「3. ファイルのコピー」の「+」ボタンを押してください。

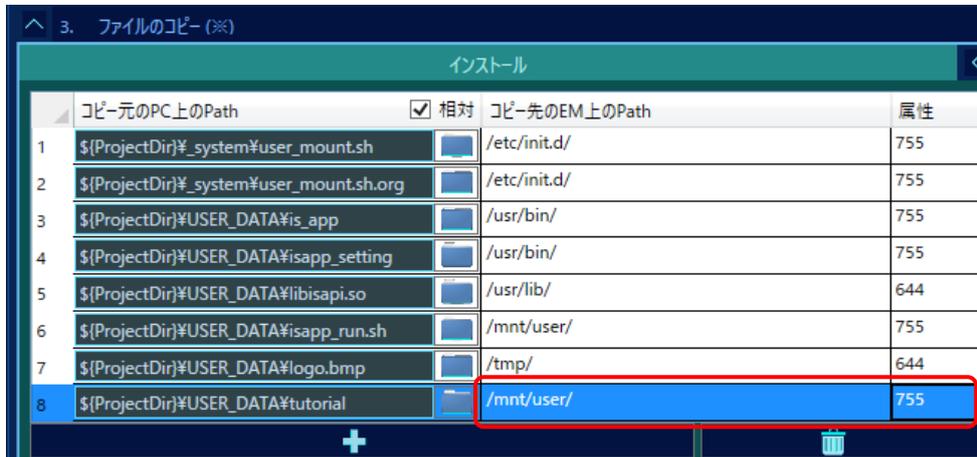


4. 追加された行のフォルダボタンを押して「USER_DATA」フォルダ内にコピーした「tutorial」ファイルを選択してください。



5. 追加された行の「コピー先のEM上のPath」と「属性」に以下のように記入してください。

項目	設定値
コピー先のEM上のPath	/mnt/user/
属性	755



上記の設定で PC 上の「tutorial」ファイルを EM シリーズ内の「/mnt/user/」にコピーして、属性（パーミッション）を 755（実行権限付加）に設定する処理が行われます。

他のファイルがある場合は、同様の手順で追加してください。

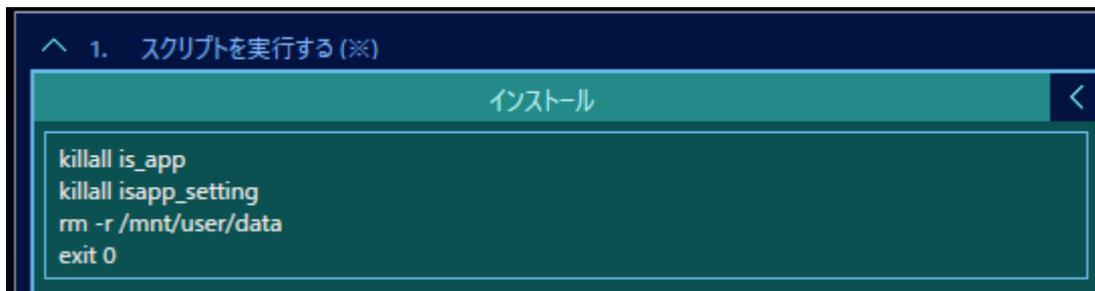
さらにお客様独自にシステム設定/Linux をカスタマイズする場合は「[2.11.5 テンプレートプロジェクト解説（高度な設定）](#)」「[2.11.6 アンインストール処理（高度な設定）](#)」を参考にカスタマイズしてください。

2.11.5 テンプレートプロジェクト解説(高度な設定)



テンプレートプロジェクトに設定されている処理を解説します。必要に応じて変更してください。
Smart e-Studio の操作については、別紙「Smart e-Studio 取扱説明書」を参照ください。

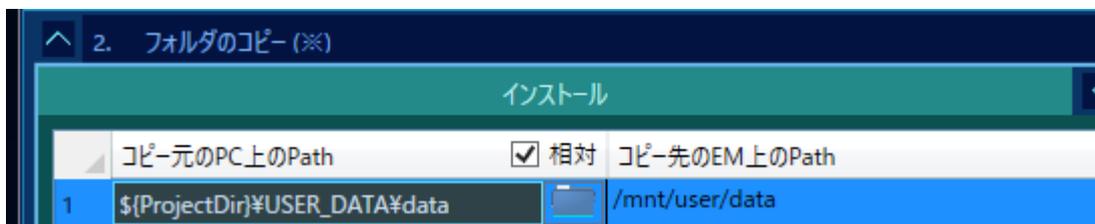
Step1 スクリプトを実行する



処理	内容
killall is_app	is_app 更新の為、終了します。
killall isapp_setting	isapp_setting 更新の為、終了します。
rm -r /mnt/user/data	IS-APP 画面データ更新の為、古いデータを削除します。
exit 0	古い画面データが存在しない場合でも成功とする為の記述です。

IS-APP、IS-APP 設定ツール、IS-APP 画面データを更新する為の準備を行っています。
基本的には変更する必要はございません。

Step2 フォルダのコピー



コピー元の PC 上の Path	コピー先の EM 上の Path	内容
\${ProjectDir}¥USER_DATA¥data	/mnt/user/data	IS-APP 画面データを EM シリーズへ転送します。

IS-APP 画面データを PC から EM シリーズへ転送する処理です。
基本的には変更する必要はございません。

Step3 ファイルのコピー

	コピー元のPC上のPath	<input checked="" type="checkbox"/> 相対	コピー先のEM上のPath	属性
1	\${ProjectDir}*_system*_user_mount.sh	<input checked="" type="checkbox"/>	/etc/init.d/	755
2	\${ProjectDir}*_system*_user_mount.sh.org	<input checked="" type="checkbox"/>	/etc/init.d/	755
3	\${ProjectDir}*_USER_DATA*_is_app	<input checked="" type="checkbox"/>	/usr/bin/	755
4	\${ProjectDir}*_USER_DATA*_isapp_setting	<input checked="" type="checkbox"/>	/usr/bin/	755
5	\${ProjectDir}*_USER_DATA*_libisapi.so	<input checked="" type="checkbox"/>	/usr/lib/	644
6	\${ProjectDir}*_USER_DATA*_isapp_run.sh	<input checked="" type="checkbox"/>	/mnt/user/	755
7	\${ProjectDir}*_USER_DATA*_logo.bmp	<input checked="" type="checkbox"/>	/tmp/	644

コピー元の PC 上の Path	コピー先の EM 上の Path	属性	内容
\${ProjectDir}*_system*_user_mount.sh	/etc/init.d/	755	ユーザ領域の書込保護設定更新
\${ProjectDir}*_system*_user_mount.sh.org	/etc/init.d/	755	
\${ProjectDir}*_USER_DATA*_is_app	/usr/bin/	755	is_app 更新
\${ProjectDir}*_USER_DATA*_isapp_setting	/usr/bin/	755	isapp_setting 更新
\${ProjectDir}*_USER_DATA*_libisapi.so	/usr/lib/	644	libisapi.so 更新
\${ProjectDir}*_USER_DATA*_isapp_run.sh	/mnt/user/	755	isapp_run.sh 転送
\${ProjectDir}*_USER_DATA*_logo.bmp	/tmp/	644	起動画面ファイル転送

システム設定の変更、IS-APP 関連アプリの更新、起動画面ファイルの転送処理です。

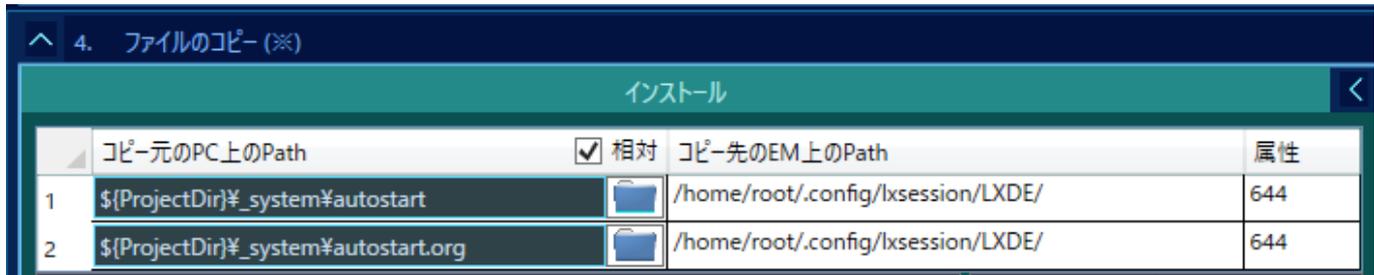
既存項目は基本的には変更する必要はありません。

お客様が作成したファイルは本ステップに追加してください。

以下は、Smart e-Start 対応製品のみ

コピー元の PC 上の Path	コピー先の EM 上の Path	属性	内容
\${ProjectDir}*_system*_resume.sh	/mnt/user/	755	resume.sh 初期化
\${ProjectDir}*_system*_suspend.sh	/mnt/user/	755	suspend.sh 初期化
\${ProjectDir}*_system*_Sstart-core.sh	/usr/bin/emsystem/	755	Smart e-Start 設定更新
\${ProjectDir}*_system*_Sstart-core.sh.org	/usr/bin/emsystem/	755	

Step4 ファイルのコピー



コピー元の PC 上の Path	コピー先の EM 上の Path	属性	内容
\${ProjectDir}¥_system¥autostart	/home/root/.config/lxsession/LXDE/	644	デスクトップ、タスクバー
\${ProjectDir}¥_system¥autostart.org	/home/root/.config/lxsession/LXDE/	644	の非表示

デスクトップ、タスクバーの非表示を行う為、「lxpanel」「pcmanfm」の起動を停止しています。このアプリは以下の処理を行っています。

プログラム名	機能
lxpanel (LXDE タスクバー)	スタートメニュー、LXDE タスクバーの表示など
pcmanfm (PCMan File Manager)	デスクトップ表示、外部ストレージの自動マウントなど

この機能をご使用される場合は、本ステップを削除してください。

Step5 ハードウェアの設定

「[2.11 テンプレートパッケージのカスタマイズ](#)」を参照ください。

Step6 スクリプトを実行する

```

5. スクリプトを実行する (※)
インストール
rm /etc/rc5.d/S02banner.sh
rm /etc/rc5.d/S36udev-cache
rm /etc/udev-cache.tar.gz
rm /etc/rc5.d/S12rpcbind
rm /etc/rc5.d/S20bluetooth
rm /etc/rc5.d/S20ntpd
rm /etc/rc5.d/S20samba.sh
rm /etc/rc5.d/S22ofono
rm /etc/rc5.d/S70lighttpd
rm /etc/rc5.d/S90crond
exit 0

```

処理	内容
rm /etc/rc5.d/S02banner.sh	起動中の“Please wait: booting...”表示を無効化します。
rm /etc/rc5.d/S36udev-cache	udev-cache 処理を無効化します。
rm /etc/udev-cache.tar.gz	udev-cache が作成されていた場合は削除します。
rm /etc/rc5.d/S12rpcbind	rpcbind daemon の起動を無効化します。
rm /etc/rc5.d/S20bluetooth	bluetooth daemon の起動を無効化します。
rm /etc/rc5.d/S20ntpd	ntp daemon の起動を無効化します。
rm /etc/rc5.d/S20samba.sh	samba server の起動を無効化します。
rm /etc/rc5.d/S22ofono	oFono daemon の起動を無効化します。
rm /etc/rc5.d/S70lighttpd	lighttpd daemon の起動を無効化します。
rm /etc/rc5.d/S90crond	cron daemon の起動を無効化します。
exit 0	既に無効化されている場合でも成功とする記述です。

不要なサービスを停止して、IS-APP のみご使用のお客様に適した設定に変更しています。

停止しているサービスをご使用される場合は、該当行を削除してください。

Samba サーバーを使用する場合は「rm /etc/rc5.d/S20samba.sh」の行を削除してください。

Step7 スクリプトを実行する

```

^ 6. スクリプトを実行する (※)
インストール
psplash -w /tmp/logo.bmp /dev/mtd6
/usr/bin/emsystem/set_mode_of_user_area1 1
/usr/bin/emsystem/set_mode_of_user_area2 1

```

処理	内容
psplash -w /tmp/logo.bmp /dev/mtd6	Step3 で転送した起動画面ファイルを起動画面保存領域に書き込みます
/usr/bin/emsystem/set_mode_of_user_area1 1	ユーザ領域 1 の書き込み保護設定
/usr/bin/emsystem/set_mode_of_user_area2 1	ユーザ領域 2 の書き込み保護設定 ※1

※1 標準製品のみ

システム設定の変更、起動画面ファイルの書込処理です。
基本的には変更する必要はございません。

2.11.6 アンインストール処理(高度な設定)



インストールしたパッケージは Smart e-Studio からアンインストールすることが可能です。アンインストール時には以下の処理が実行されます。必要に応じてカスタマイズしてください。
※ アンインストール時は最終ステップから逆順で実行します。

スクリプトを実行する

```

> アンインストール <
In -s ../init.d/banner.sh /etc/rcS.d/S02banner.sh
In -s ../init.d/udev-cache /etc/rcS.d/S36udev-cache
In -s ../init.d/rpcbind /etc/rc5.d/S12rpcbind
In -s ../init.d/bluetooth /etc/rc5.d/S20bluetooth
In -s ../init.d/ntpd /etc/rc5.d/S20ntpd
In -s ../init.d/samba.sh /etc/rc5.d/S20samba.sh
In -s ../init.d/ofono /etc/rc5.d/S22ofono
In -s ../init.d/lighttpd /etc/rc5.d/S70lighttpd
In -s ../init.d/crond /etc/rc5.d/S90crond

```

インストールステップ6で停止したサービスを有効に戻しています。

スクリプトを実行する

```

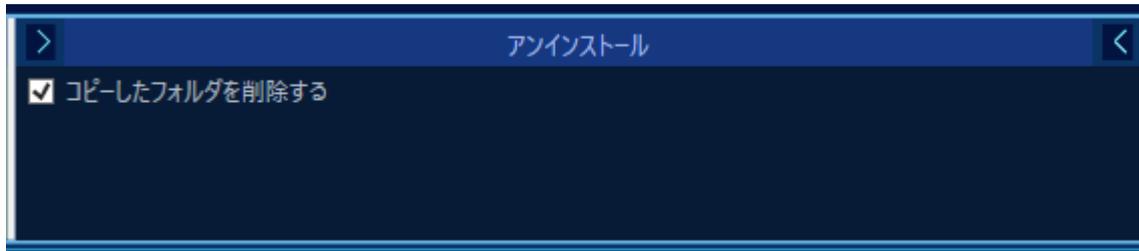
> アンインストール <
em-setting-cui set AutoStartupMode=launcher
passwd -d root

```

インストールステップ5で変更した設定を以下に再変更しています。

処理	内容
em-setting-cui set AutoStartupMode=launcher	自動起動を「EMG ランチャー」に設定します。
passwd -d root	root パスワードを削除しています。

フォルダの削除



処理	内容
/mnt/user/data を削除する	インストールしたファイルを削除しています。

スクリプトを実行する



処理	内容
mv /etc/init.d/user_mount.sh.org /etc/init.d/user_mount.sh	ユーザ領域の書込保護設定を元に戻しています。
mv /home/root/.config/lxsession/LXDE/autostart.org /home/root/.config/lxsession/LXDE/autostart	デスクトップ、タスクバーの非表示の解除
rm /mnt/user/isapp_run.sh	インストールしたファイルを削除しています。
/usr/bin/emsystem/set_mode_of_user_area1 0	ユーザ領域 1 の書き込み保護設定解除
/usr/bin/emsystem/set_mode_of_user_area2 0	ユーザ領域 2 の書き込み保護設定解除 ※1
mv /usr/bin/emsystem/SStart-core.sh.org /usr/bin/emsystem/SStart-core.sh	Smart e-Start 設定を元に戻しています。 ※2

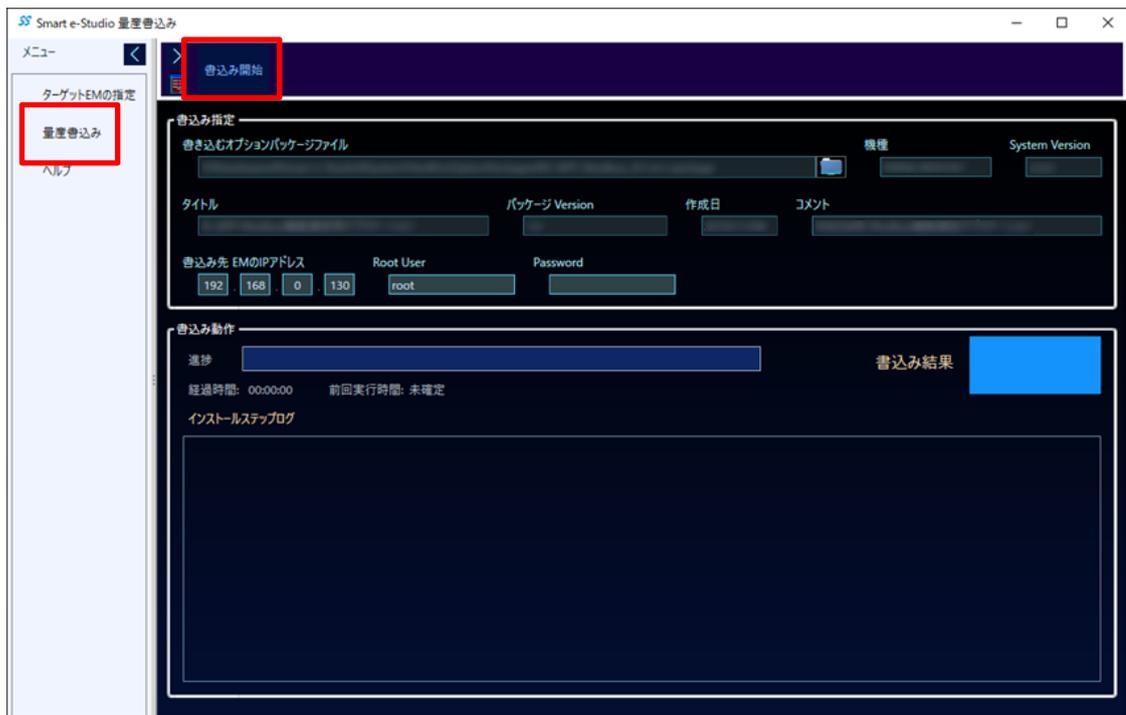
※1 標準製品のみ

※2 Smart e-Start 対応製品のみ

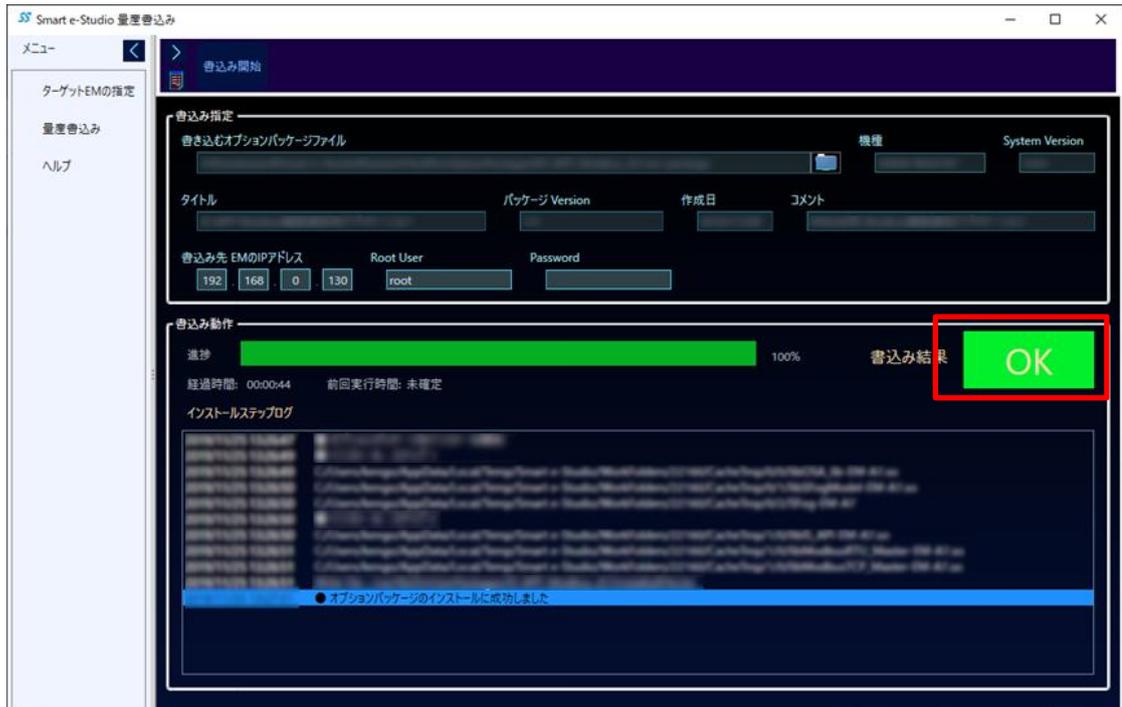
2.12 Smart e-Studioパッケージの書き込み

お客様が作成したデータを EM シリーズに書き込みます。
手順はテンプレートパッケージの書き込みと同様です。

1. 「[2.3 EM シリーズ本体と PC との接続](#)」を参考に Smart e-Studio で EM シリーズ本体と通信が可能な状態にしてください。
2. 「[2.11 テンプレートパッケージのカスタマイズ](#)」で作成した「EM_IS-APP_template.em-package」を開いてください。
3. 「量産書き込み」ページを開き、「書き込み開始」を押してください。

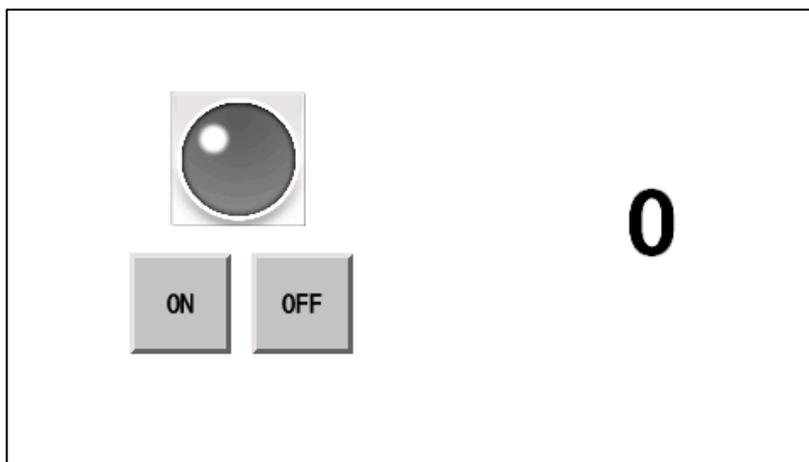


書き込み結果に「OK」が表示されると完了です。



以上で書き込み完了です。

自動的に再起動して本手順で作成した画面が表示されます。



2.13 動作確認

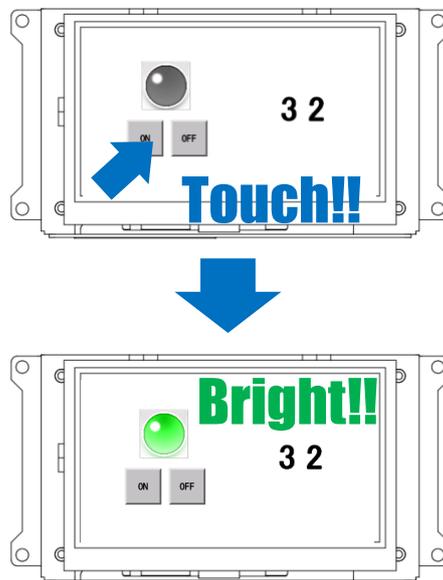
EM シリーズに転送したデータの動作確認を行います。

電源を入れると設定した起動画面が表示されて、その後に IS-APP が起動します。

2.13.1 InfoSOSA ビルダで設定した処理

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。

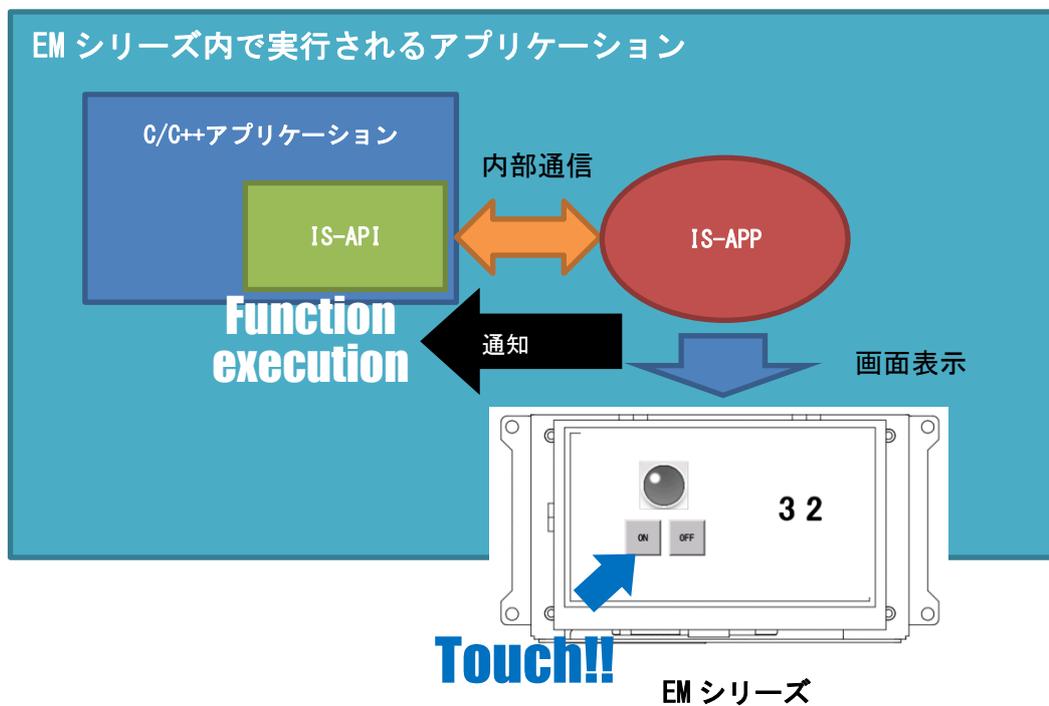


2.13.2 C++プログラムで設定した処理



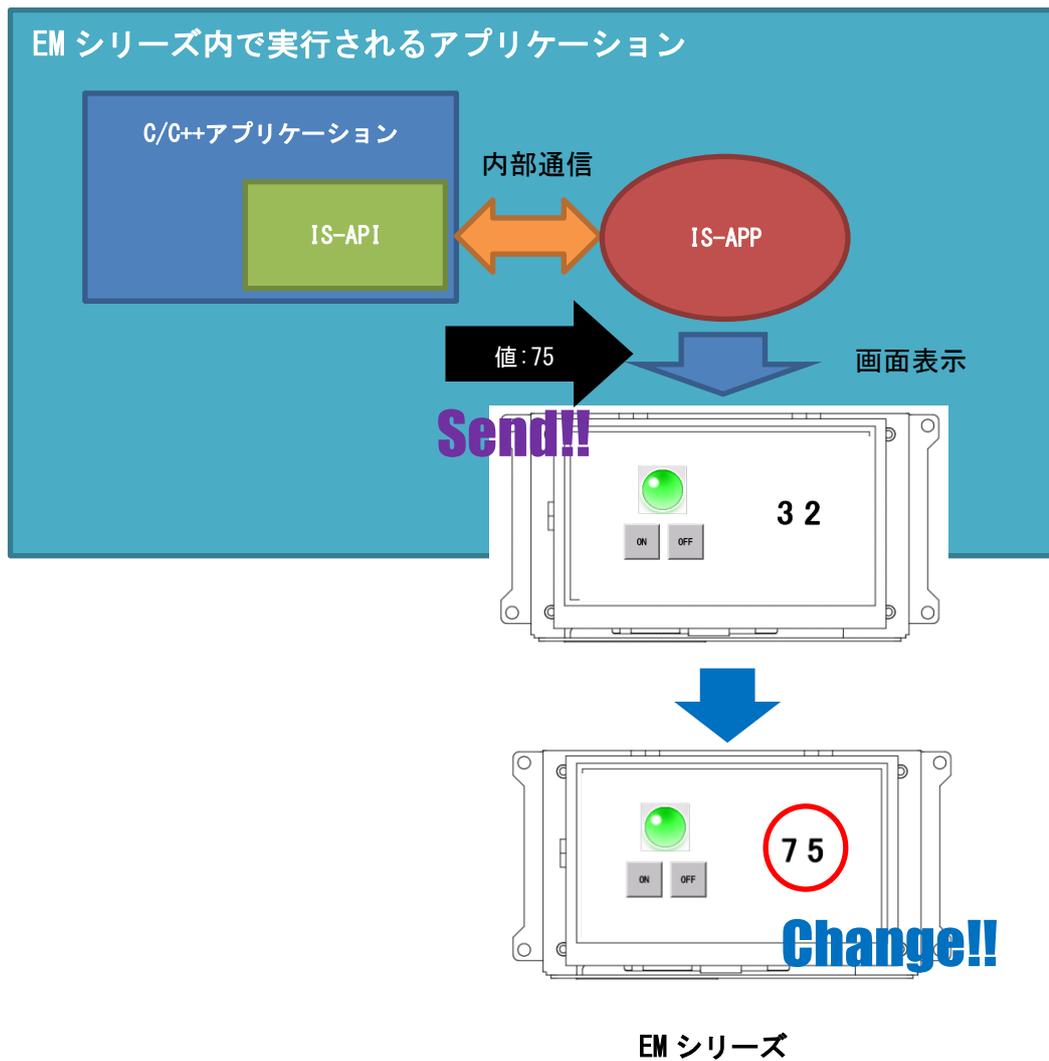
液晶画面上 (IS-APP) の ON ボタンをタッチすると数値が連続して増加します。OFF ボタンをタッチすると数値の増加が停止します。

これは、ボタンを押した時 IS-APP がイベントを送信し、そのイベントを受信して C++アプリケーション内の IS-API がイベントに対応した関数を実行しています。



その実行された関数で動作フラグを操作し、メイン関数でフラグの状態に応じて IS-API を使用して IS-APP のメモリ「GM_VOL01」の値を書換えています。

メモリ「GM_VOL01」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の表示が変化します。



2.14 各機能の詳細について

各機能の詳細は以下を参照ください。

InfoSOSA ビルダ/IS-APP

InfoSOSA ビルダ (IS-APP) は、様々な部品やアクションがあり、プログラミングを行わずに画面を作成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 アプリケーションの詳細仕様は「[3.1 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、「[3.2 IS-API 詳細仕様](#)」を参照ください。

ISAPP SETTING

IS-APP 設定ツール (ISAPP SETTING) の詳細仕様は、「[3.3 ISAPP SETTING](#)」を参照ください。

Smart e-Studio

Smart e-Studio には USB メモリでの更新用データを作成する機能もございます。オプション機能になる為、最初に PC を使って EM シリーズ本体に USB メモリアップデート機能をインストールする必要がありますが、出荷後に現地では PC を使わずにアップデートすることが可能です。詳しくは、別紙「Smart e-Studio 取扱説明書」を参照ください。

3章 リファレンス

章目次

3.1	IS-APP 詳細仕様	96
3.2	IS-API 詳細仕様.....	109
3.3	ISAPP SETTING.....	139

3.1 IS-APP詳細仕様

3.1.1 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。

コマンドライン引数が設定された起動スクリプトを IS-APP 設定ツールで作成することも可能です。

詳しくは「[3.3 ISAPP SETTING](#)」を参照ください。

開発の進め方は、「[2章 開発手順\(チュートリアル\)](#)」を参照ください。

3.1.2 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、「[1.4 IS-APP/IS-API/IS-APP SETTING の更新](#)」を参照ください。

3.1.3 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。

3.1.4 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

3.1.5 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	×

使用するシステムフォント種類はInfoSOSAビルダで日本語、韓国語、ヨーロッパ言語、中国語(簡体字、繁体字)から選択します。英語(アルファベット)は全てのフォントに含まれています。詳細は別紙「InfoSOSA リファレンスマニュアル」を参照ください。

3.1.6 コマンドライン引数

コマンドライン引数が設定された起動スクリプトを IS-APP 設定ツールで作成することが可能です。詳しくは「[3.3 ISAPP SETTING](#)」を参照ください。

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 --rsmdir

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。

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`
- アクション「上位へ文字列型メモリの文字列を出力する」実行時の出力先は、Main 引数にシリアルを指定した場合のみ出力されます。LAN、IS-API を指定した場合や、Sub 引数には出力されません。

- 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}00s000000080001000003DC{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。 デフォルト : SIO1 のデバイスファイルが自動設定されます SIO1 : /dev/com1 SIO2 : /dev/com2
<通信速度>	通信速度を指定します。 4800 : 4800bps 9600 : 9600bps 19200 : 19200bps 38400 : 38400bps 57600 : 57600bps 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。 none：パリティ無し ※デフォルト odd：奇数パリティ 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。 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。 1～31 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。 0～10 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。 0～1000 (単位 10ms) 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。 0～3000 (単位 10ms) 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。 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。 0~65535 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。 0~99 (単位 秒) 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。 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。 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*。 0~65535 デフォルト値：51111 ※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。 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。 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。 0~65535 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。 0~65535 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。 immediate：即時通知 ※デフォルト 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。 1~5(秒) 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。 0~3 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。 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。 ※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
```

3.2 IS-API詳細仕様

3.2.1 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。

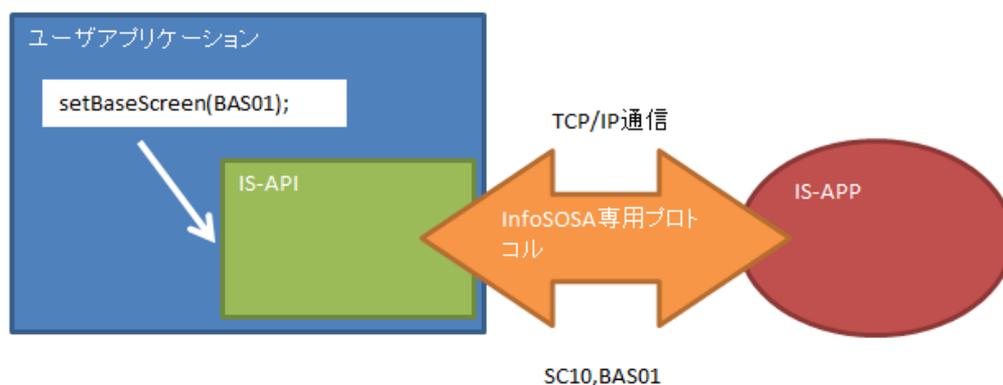
IS-APIを使用するためには、IS-APPの通信設定をIS-APIにして起動する必要があります。

IS-APPの通信設定については、「[3.1.6 コマンドライン引数 通信設定](#)」を参照ください。

尚、コマンドライン引数が設定された起動スクリプトをIS-APP設定ツールで作成することも可能です。詳しくは「[3.3 ISAPP SETTING](#)」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。

開発の進め方は、「[2章 開発手順\(チュートリアル\)](#)」を参照ください。



3.2.2 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	
CIIsApi. h CIIsComString. h CIIsComVariant. h CIIsComVariantList. 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、「[1.4 IS-APP/IS-API/IS-APP SETTING の更新](#)」を参照ください。

3.2.3 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。

3.2.4 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。

3.2.5 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

3.2.6 API 一覧

■ClsApi 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。 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。 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
<code>bool setGroupMemory(std::string* _groupID, CIsComVariantList& _varList)</code>	IS-APP のグローバルメモリグループに値を設定します。
<code>bool getGroupMemory(std::string* _groupID, CIsComVariantList& _varList)</code>	IS-APP のグローバルメモリグループの値を取得します。
<code>bool callSubRoutine(std::string* _subroutineID)</code>	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。
<code>bool registerCallback(std::string* _eventID, EventCallbackFunc _func)</code>	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。
<code>bool unregisterCallback(std::string* _eventID)</code>	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
<code>bool SetValue(int value)</code>	CIsComVariant クラスに値を設定します。
<code>int GetValue()</code>	CIsComVariant クラスの値を取得します。

■CIsComString クラス

公開機能	内容
<code>bool SetString(int bytes, char* str)</code>	CIsComString クラスに文字列を設定します。
<code>char* GetStringBuffer()</code>	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。
<code>int GetStringLength()</code>	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
<code>bool Add(CIsComVariant& elem)</code>	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。
<code>CIsComVariant* GetElem(int index)</code>	CIsComVariantList クラスから指定した要素を取得します。
<code>int Size()</code>	CIsComVariantList クラスに登録されている要素数を取得します。
<code>bool Clear()</code>	CIsComVariantList の全ての登録を削除します。
<code>CIsComVariant* ReplaceElem(int index, CIsComVariant& elem)</code>	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。

3.2.7 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ありません。1つのスレッドから実行してください。

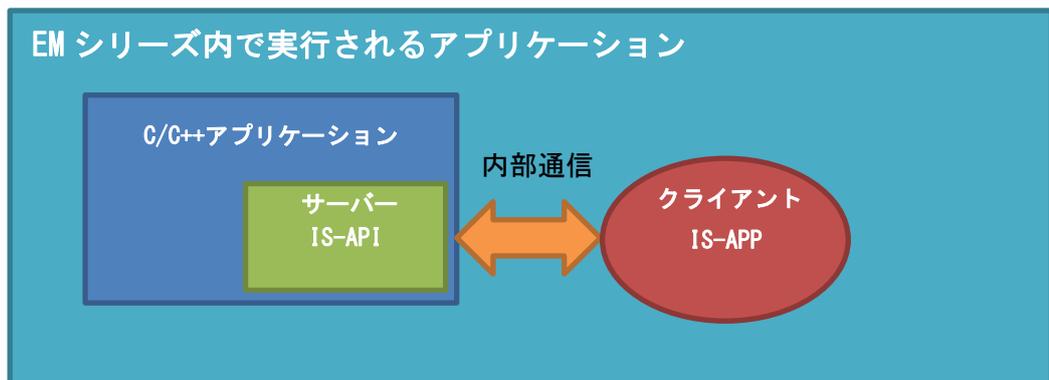
IS-APP 側が TCP/IP クライアントの場合は、「ClsApi」クラスを使用します。

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。

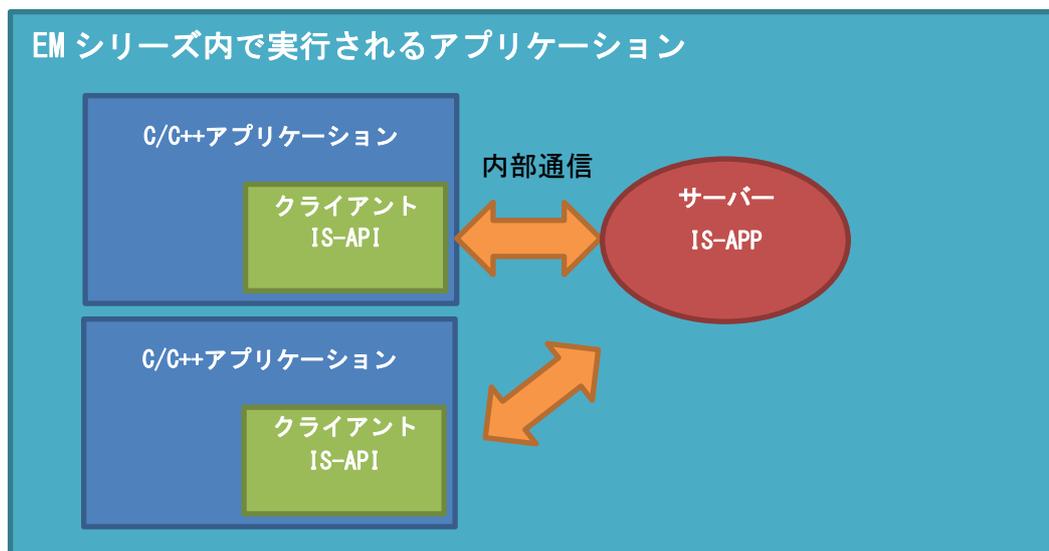
【対応表】

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 CIsApi_c : IS-API が接続するポート番号 ※デフォルトは 51111 です。
unsigned int startTimeout	CIsApi : 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。 CIsApi_c : 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 (XYX)

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

S102

■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ーバージョン ※ユーザーバージョンは 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) 範囲 : 500~5000
int _msec	ブザーを鳴らす時間 (ms) 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 ※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 ※文字列の文字コードは必ず UTF8 で設定してください。 ※”,” カンマ (0x2C)、制御文字は使用できません。

■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 ※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 ※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 ※文字列の文字コードは必ず UTF8 で設定してください。 ※文字列に”,” カンマ (0x2C)、制御文字は使用できません。

■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 = "GRP0001";
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 ※空である必要があります。 ※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 ※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(ClsComVariantList &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);
```

3.2.8 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

Cl sComVar iant. h

公開機能

SetValue

Cl sComVar iant 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

```
Cl sComVar iant var;
```

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

GetValue

CIsComVariant 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;
```

3.2.9 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

ClsComVariant.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 ※文字列の文字コードは必ず UTF8 で設定してください。 ※文字列に”,” カンマ(0x2C)、制御文字は使用できません。

■戻り値

```
true:成功  
false:失敗
```

■使用例

```
ClsComString 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

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

3.2.10 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

ClsComVariantList.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(ClsComVariant& elem)
```

■パラメータ

パラメータ	説明
ClsComVariant& elem	追加する ClsComVariant 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

```
ClsComVariant* GetElem(int index)
```

■パラメータ

パラメータ	説明
int index	取り出す ClsComVariant データの位置

■戻り値

ClsComVariant 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

CIsComVariantList クラスの指定した位置に CIsComVariant 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));
```

3.3 ISAPP SETTING

3.3.1 概要

IS-APP のコマンドライン引数が設定された起動スクリプトを作成するツールです。

3.3.2 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

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。
更新方法は、「[1.4 IS-APP/IS-API/IS-APP SETTING の更新](#)」を参照ください。

3.3.3 使用方法

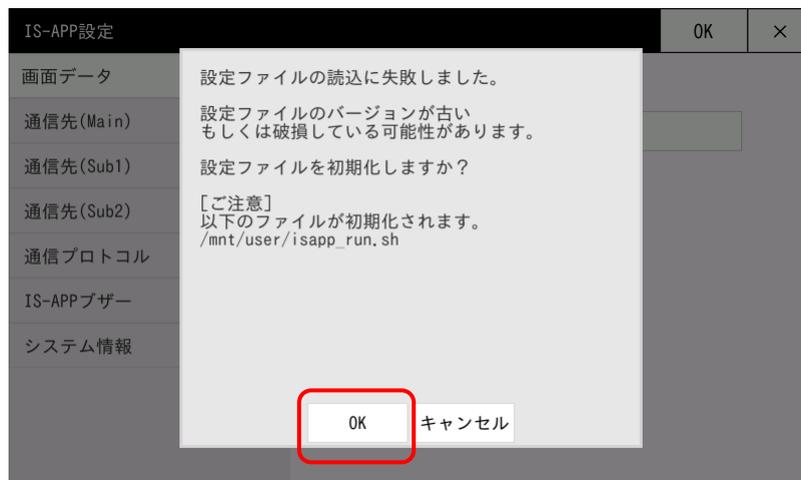
EMG ランチャーから ISAPP SETTING を起動します。

[EMG ランチャー] - [ISApp 設定]



次の場合は、下図のダイアログが表示されます。

- 設定ファイルのバージョンが古い
- 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 の起動スクリプトの作成

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

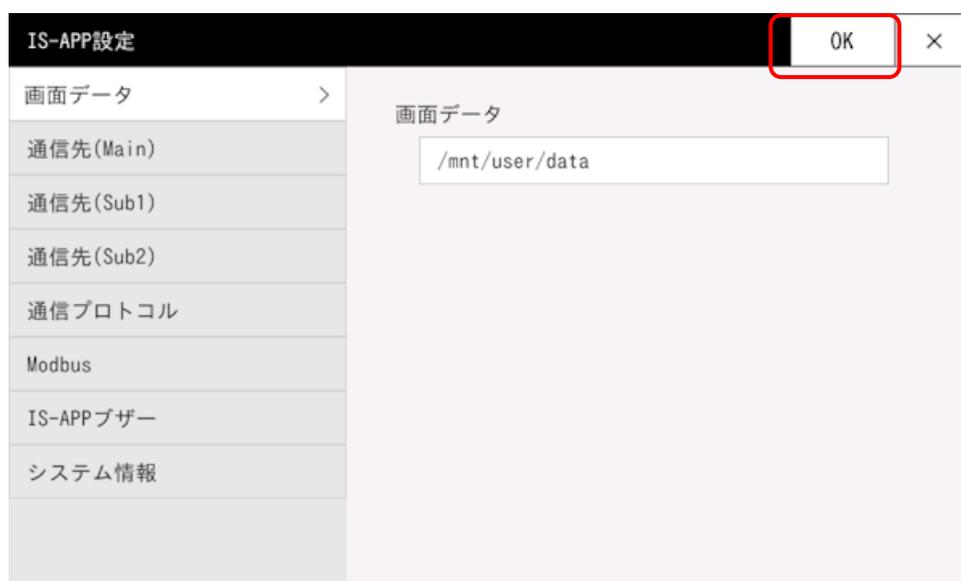
項目	説明
画面データフォルダ	IS-APP で表示する画面データフォルダを指定します。
通信設定	IS-APP の外部機器との通信設定を行います。
通信プロトコル	IS-APP が外部機器と通信を行う時のプロトコルを設定します。
ブザー	IS-APP がブザーデバイスを使用するか設定します。

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
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 ミキサー名：default ボリューム名：PCM ※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、「[3.1.6 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。

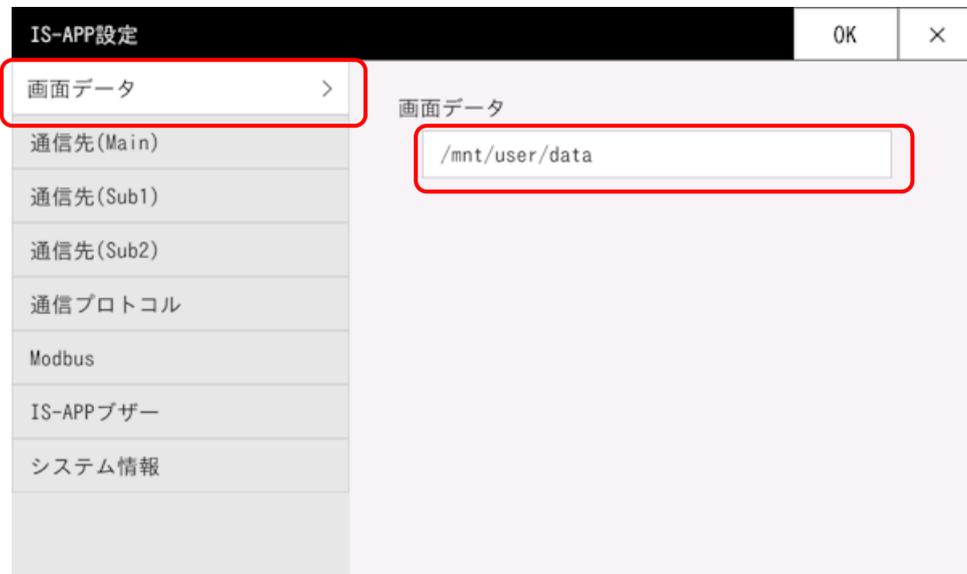


起動スクリプトは以下の場所に作成されます。

```
/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。
- アクション「上位へ文字列型メモリの文字列を出力する」実行時の出力先は、通信先(Main)に SIO を指定した場合のみ出力されます。LAN、IS-API を指定した場合や、通信先(Sub1)、通信先(Sub2)には出力されません。
- 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}00s000000080001000003DC{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、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。

The screenshot shows the 'IS-APP設定' (IS-APP Settings) dialog box. The left sidebar contains a menu with '通信先(Main)', '通信先(Sub1)', and '通信先(Sub2)' highlighted by a red box. The main area shows the '通信先' (Destination) field set to 'IS-API' and the 'IS-APIのポート番号' (IS-API Port Number) field set to '51111'. The dialog has 'OK' and 'X' buttons in the top right corner.

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。

The screenshot shows the 'IS-APP設定' dialog box. On the left, a menu lists '画面データ', '通信先(Main)', '通信先(Sub1)', '通信先(Sub2)', '通信プロトコル', 'Modbus', 'IS-APPブザー', and 'システム情報'. The '通信先(Main)' item is highlighted with a red box. On the right, the '通信先' field is set to 'LAN', and the 'モード' field is set to 'UDP/IP'. Below these, the 'アドレス' field is set to '192.168.0.100' and the 'ポート番号' field is set to '51111'. The dialog box has 'OK' and '×' buttons in the top right corner.

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。

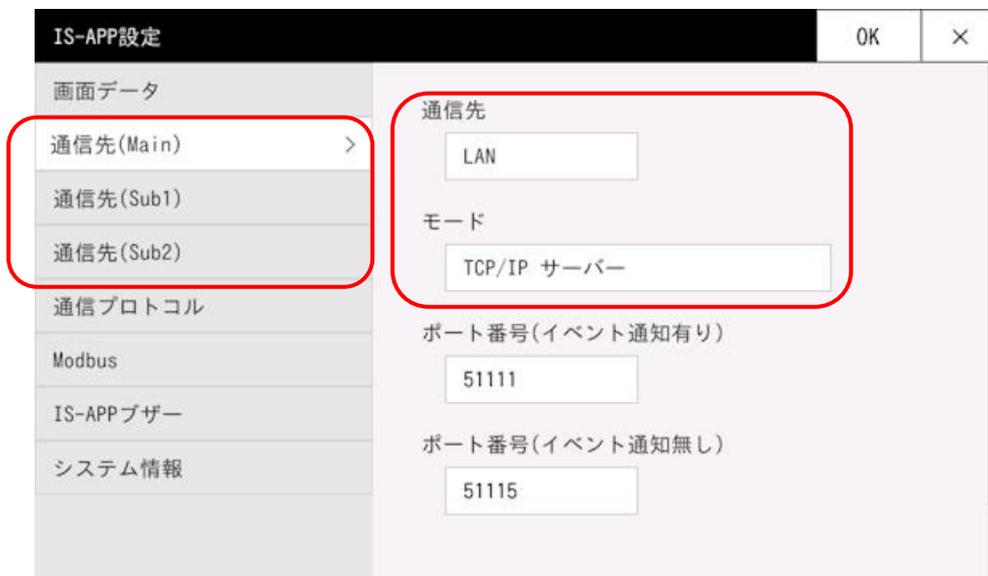
The screenshot shows the 'IS-APP設定' dialog box. On the left, a menu lists '画面データ', '通信先(Main)', '通信先(Sub1)', '通信先(Sub2)', '通信プロトコル', 'Modbus', 'IS-APPブザー', and 'システム情報'. The '通信先(Main)' item is selected and highlighted with a red box. The main area shows the following settings:

- 通信先: LAN
- モード: TCP/IP クライアント
- アドレス: 192, 168, 0, 100
- ポート番号: 51111
- TCPコネクション確立要求間隔(s):

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。 最大コネクション数：1 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。 ※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。 最大コネクション数：3 ※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。

Modbus

Modbus プロトコル通信機能を有効にする場合は、メニューから「Modbus」を選択してください。



Modbus プロトコル通信機能については、別紙「InfoSOSA Modbus 機器接続ガイド」を参照ください。

ブザー

IS-APP でブザーを有効にする場合は、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。

4章 その他

章目次

4.1	お問い合わせ	155
-----	--------------	-----

4.1 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/

Microsoft®、Windows®、Windows® 10、Windows® 11、Microsoft® .NET Framework は米国 Microsoft Corporation の米国およびその他の国における登録商標または商標です。その他、記載されている会社名、製品名は各社の登録商標または商標です。

2025年12月 第13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 (代) FAX：(03)-6721-6732

URL：https://www.dush.co.jp/

本製品及び本書は著作権法によって保護されていますので、無断で複写、複製、転載、改変する事は禁じられています。