Smart Analog

Smart Analog 開発チュートリアル TSA-IC300 版

R20AN0262JJ0100
Rev.1.00
2014.02.28

要旨

本アプリケーションノートでは、Renesas が開発・販売している「Smart Analog IC300」を初めてお使い頂く皆様を対象に、開発の流れをご理解頂くために各ツールの使い方を実際の例に沿って説明します。

動作確認デバイス

Smart Analog IC 300(RAA730300DFP)、RL78/G1A(R5F10ELEAFB)

目次

1. 概説

本アプリケーションノートは、回路構成可変アンプ型の Smart Analog IC 300 と RL78/G1A を搭載した Smart Analog IC 300 評価ボード「TSA-IC300」（以降、「TSA-IC300」）および Smart Analog の開発環境を使用し、圧力センサの駆動と LED を制御するプログラムの開発例を説明するものです。

1.1 開発環境

Smart Analog 製品の開発環境は最新版の開発ツールをお使いください。

ツール名称	推奨バージョン
センサ選定用 Web シミュレータ	Renesas VA Version 2.0.0.0 以降
アナログフロントエンド回路設計ツール	SA-Designer V1.02.00 以降
統合開発環境	CubeSuite+ V2.00.00 以降

2.　ご用意していただくもの

本資料の内容に沿って実際に開発を体験頂くには以下のものをご用意ください。

ご用意頂くもの一覧

2.1　テセラ・テクノロジー社製 Smart Analog IC 搭載評価ボード「TSA-IC300」　×1

2.2　Renesas 製オンチップデバッギングエミュレータ「E1」　×1

2.3　メトロダインマイクロシステム社製圧力センサ「MPS-2407-015AD」　×1

2.4　LED　×8

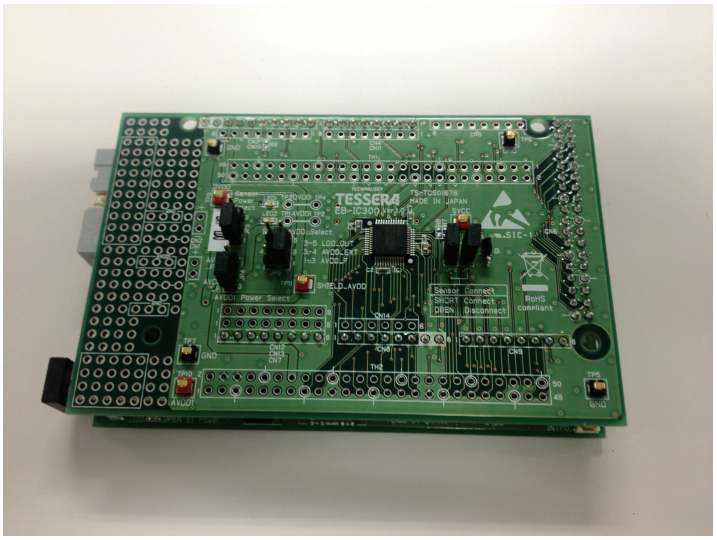
※本資料では、緑色 LED×3、黄色 LED×3、赤色 LED×2 を用いて説明します。

2.5　抵抗 470Ω(1/8W)　×8

2.6　ブレッドボード　×1

2.7　電池ケース (単三 2 本用、単三乾電池 2 本含む)　×1

2.1　テセラ・テクノロジー社製 Smart Analog IC 搭載評価ボード「TSA-IC300」



本ボードは Smart Analog IC 300(RAA730300DFP)を搭載したボード EB-IC300 と RL78/G1A (R5F10ELEAFB)を搭載したボード EB-RL78/G1A の 2 枚構成になります。

※本資料と併せて、TSA-IC300 のユーザーズマニュアルもご参照ください。下記、テセラ・テクノロジー社の URL よりダウンロードできます。

「TSA-IC300」のお求め頂き方

法人のお客様 - 弊社代理店へご注文ください。

個人のお客様 - テセラ・テクノロジー社 web サイトからご購入頂けます。

- テセラ・テクノロジー社：http://www.tessera.co.jp/tsa-ic300.html

2.2　Renesas 製オンチップデバッギングエミュレータ「E1」

本資料では上記評価ボード「TSA-IC300」に接続して、デバッガ/フラッシュプログラマとして使用します。

「E1」のお求め頂き方

法人のお客様 - 弊社代理店へご注文ください。

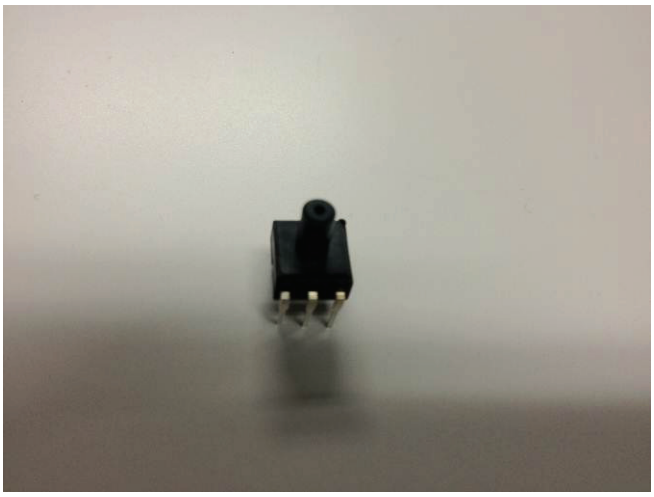
個人のお客様 - web からご注文の場合：RS オンライン、チップワンストップ、他でご購入頂けます。

- RS オンライン： http://jp.rs-online.com/web/p/processor-microcontroller-development-kits/7330317/
- チップワンストップ： http://www.chip1stop.com/dispDetail.do?partId=RNSS-0000126

また、マルツパーツ館他での店頭でもお求めいただけます。

- マルツパーツ館：http://www.marutsu.co.jp/

2.3　メトロダインマイクロシステム社製圧力センサ「MPS-2407-015AD」



本資料において Smart Analog で駆動するセンサです。

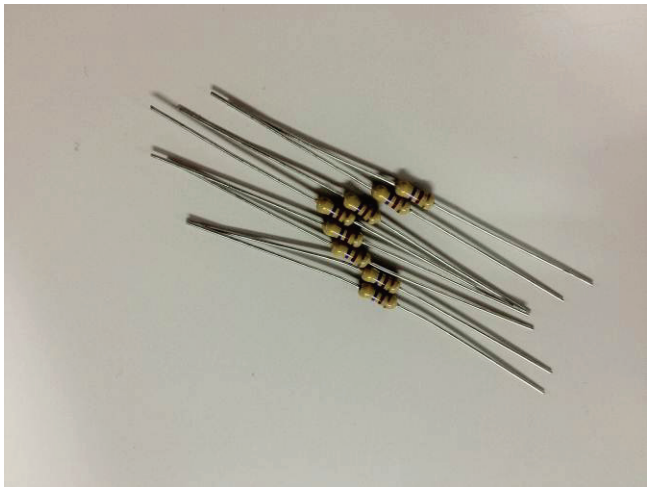
パーツ販売をされている各種ショップやオンラインショップ等でお求めいただけます。

※本資料と併せて、MPS-2400 シリーズのデータシートもご参照ください。データシートのダウンロードについては、「5.2 Renesas VA を使ってみよう！」でも紹介します。

2.4　LED

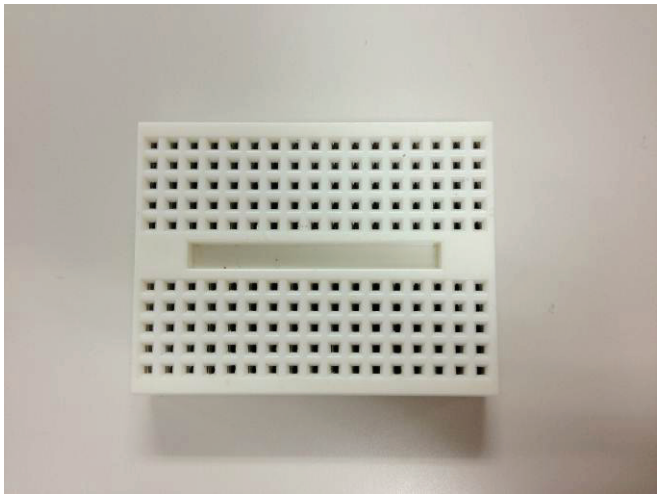
センサの挙動を受けて RL78/G1A から制御し、センサ動作状況を LED で表示します。
パーツ販売をされている各種ショップやオンラインショップ等でお求めいただけます。
なお、色については何色でも構いませんが、8 個用意して下さい。
※本資料では、緑色 LED×3、黄色 LED×3、赤色 LED×2 の、計 8 個を用いて説明します。

2.5　抵抗 470Ω(1/8W)



RL78/G1A のポートから LED を駆動する際の電流設定抵抗です。
パーツ販売をされている各種ショップやオンラインショップ等でお求めいただけます。
抵抗は 8 個用意してください。

2.6　ブレッドボード



センサと LED の配線に使用します。

パーツ販売をされている各種ショップやオンラインショップ等でお求めいただけます。

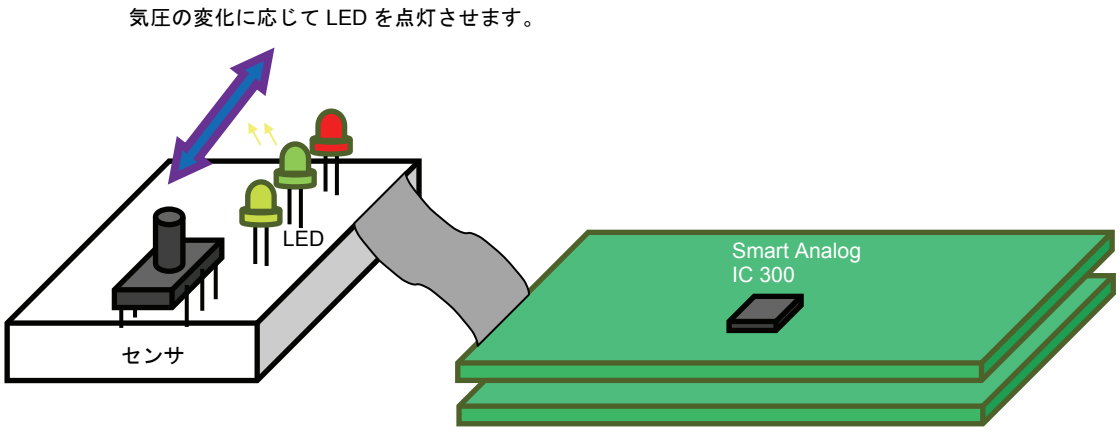
2.7　電池ケース (単三 2 本用、単三乾電池 2 本含む)

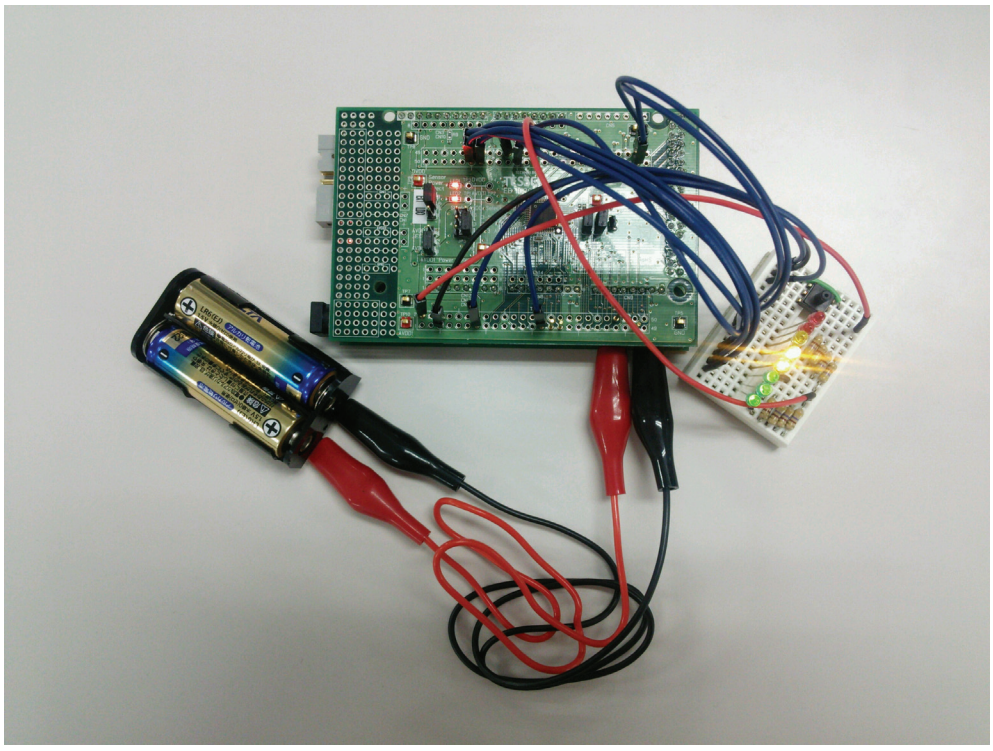
完成したプログラムをマイコンに書込んで動作確認をする際に使用します。

パーツ販売をされている各種ショップやオンラインショップ等でお求めいただけます。

3. 実際に開発するもの

圧力センサで大気圧をセンシングして、気圧の変化に応じて LED を点灯させます。
圧力センサから出力される信号を Smart Analog で最適な値に「増幅」して、ADC でデジタル化します。
デジタル化された信号の大きさに合わせて LED を点灯させるように制御します。





実物のイメージ

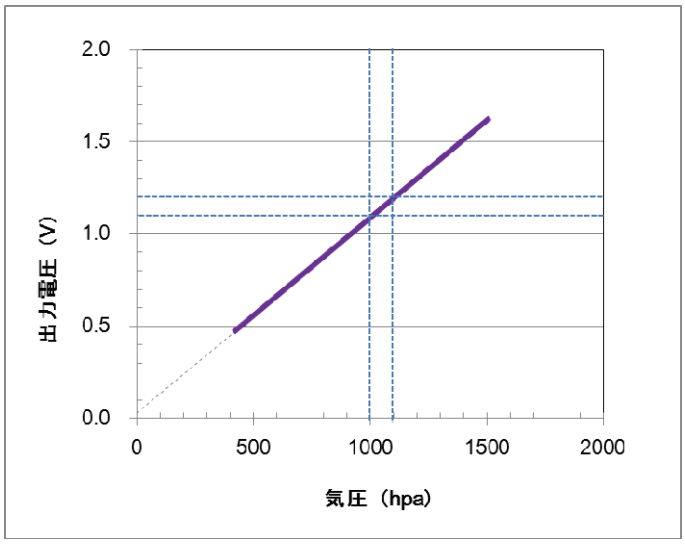
本資料では、電源が ON になった際にセンシングした気圧の値を基準として、そこから気圧が高くなると赤色の LED が点灯、気圧が低くなると緑色の LED が点灯するレベル・ゲージを作製します。

気圧センサの感度には、本資料で使用した気圧センサに対し、事前に評価した値を使用します。

※ただし、Renesas VA によるシミュレーションを行う時は、シミュレータに登録されたデータシートの値を使用してください。

事前評価では、以下のグラフに示すような Linear 特性が確認できました。

※本資料では、以下のグラフから、100hpa の気圧変化に対し「TSA-IC300」上で約 100mV の電圧変化が、10hpa の気圧変化に対し「TSA-IC300」上で約 10mV の電圧変化が発生すると考えます。



事前評価で確認された気圧の変化に対する電圧の変化

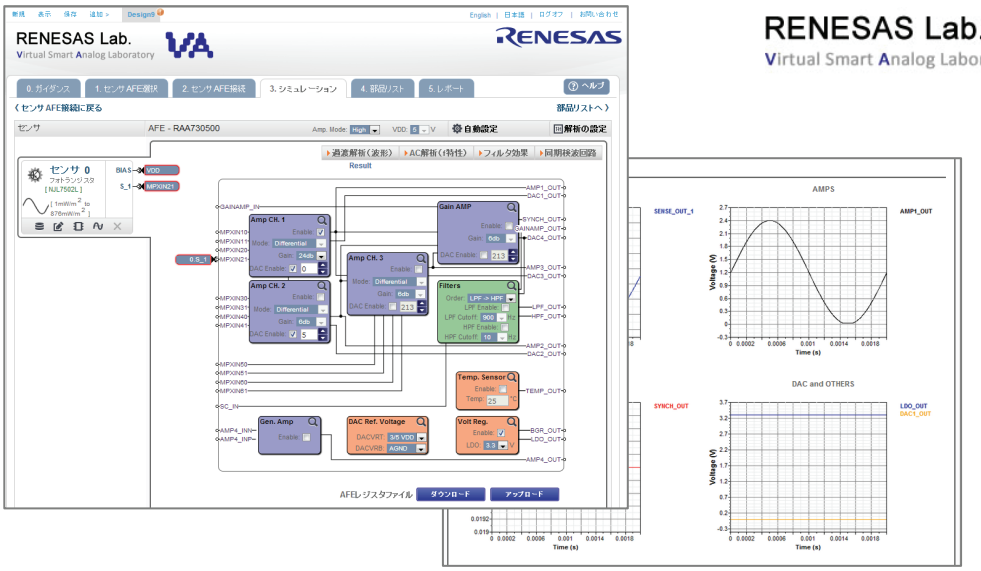
レベル・ゲージの動作については、「7.10 プログラムを編集する」の節でも詳しく紹介します。

4.　全体の流れ

本資料では以下の手順に従って開発の流れを解説します。

4.1　Renesas VA でシミュレーションを実行

シミュレータ(Renesas VA)を使ってシミュレーションします。



RENESAS Lab.
Virtual Smart Analog Laboratory

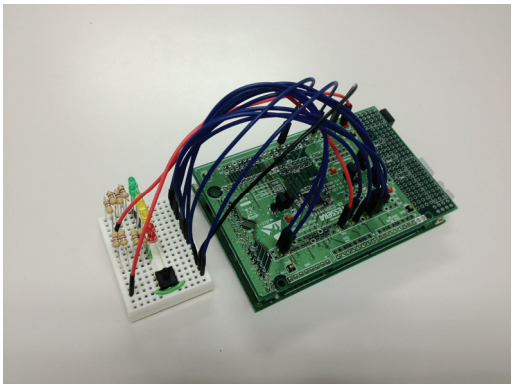
出来上がるもの

・シミュレーション結果

・Smart Analog 回路パラメータ（AFE レジスタファイル）

4.2　評価回路を作成

センサ、LED をブレッドボードに取り付けて回路を組み、評価ボードに配線します。

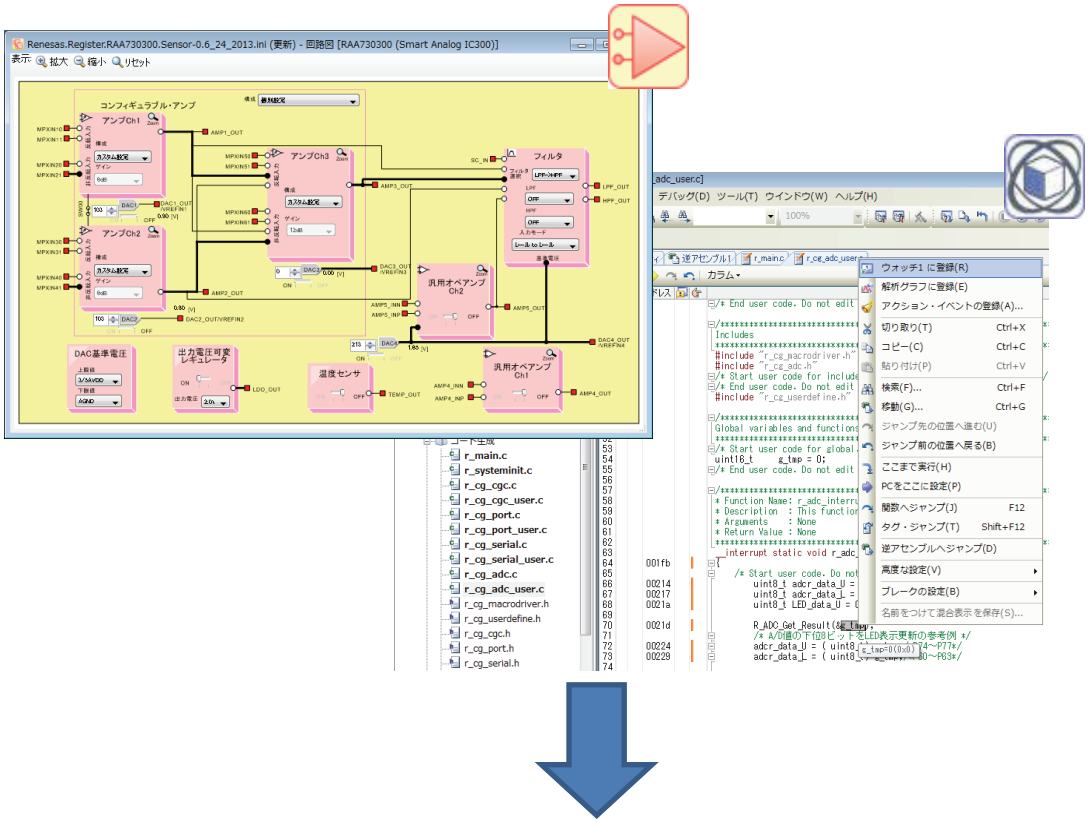


出来上がるもの

・センサ、LED 回路

4.3　SA-Designer でパラメータ読込 ～ CubeSuite+に取込みプログラムを作成

Renesas VA で生成した Smart Analog の AFE レジスタファイルを SA-Designer で C ソースに変換し、CubeSuite+にパラメータを取り込んでセンサの駆動と LED を制御するプログラムを作成します。

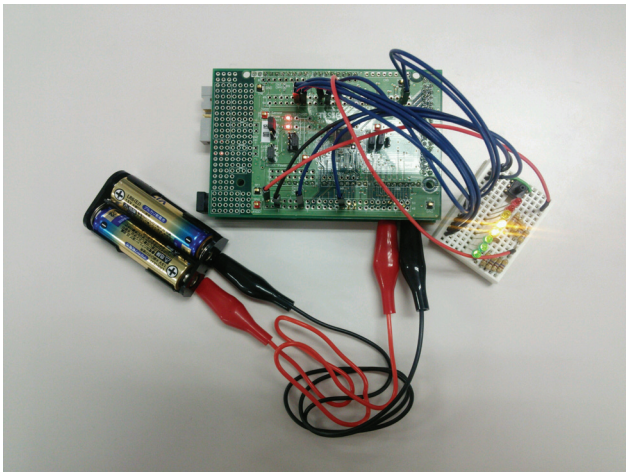


出来上がるもの

・コンパイル済み
　プログラム
　(ロードモジュール)

4.4　評価ボードで動作確認

4.3で作ったプログラムが期待通りの動作をしているかを評価ボードで確認します。



出来上がるもの

・センサ評価環境
　及びプログラム

5. Renesas VA でシミュレーションをしてみよう！

まずは Smart Analog＋センサのシミュレーションをしてみましょう。

Smart Analog＋センサのシミュレーションツールとして Renesas が用意している Web シミュレータ「Renesas VA」を利用してシミュレーションします。

5.1 そもそも Web シミュレータとは？

Smart Analog＋センサのアナログ回路設計とシミュレーションを Web ブラウザのみでできるツールです。

専用のプログラムをダウンロードしてインストール、といった作業が必要なく、インターネットにつながる環境があれば、何時でもアクセス、シミュレーションが可能です。

Renesas VA では以下の機能をサポートしています。

・回路設計
　Smart Analog とセンサを組合わせて、Smart Analog 内部の回路設計ができます。

・過渡解析
　センサからの入力信号を Smart Analog でアナログ信号処理変換した出力波形を確認できます。

・AC 解析
　Smart Analog で処理した波形の振幅周波数特性を確認できます。

・フィルタ効果解析
　Smart Analog のフィルタの特性を確認できます。

・同期検波回路動作解析
　Smart Analog に搭載された同期検波回路の動作が確認できます。

・シミュレーション内容の読出し、保存、パラメータのダウンロード
　シミュレーションした結果の保存や読出し、Smart Analog の AFE レジスタファイルのアップロードとダウンロードができます。

5.2　Renesas VA を使ってみよう！

では実際に Renesas VA を使ってみましょう。

web ブラウザから Renesas の Smart Analog の「Smart Analog Web シミュレータ」のページにアクセスしてください。

- Renesas VA：
  http://japan.renesas.com/smart_analog/va
  Renesas の TOP ページ → 製品情報 → Smart Analog → Smart Analog Web シミュレータ

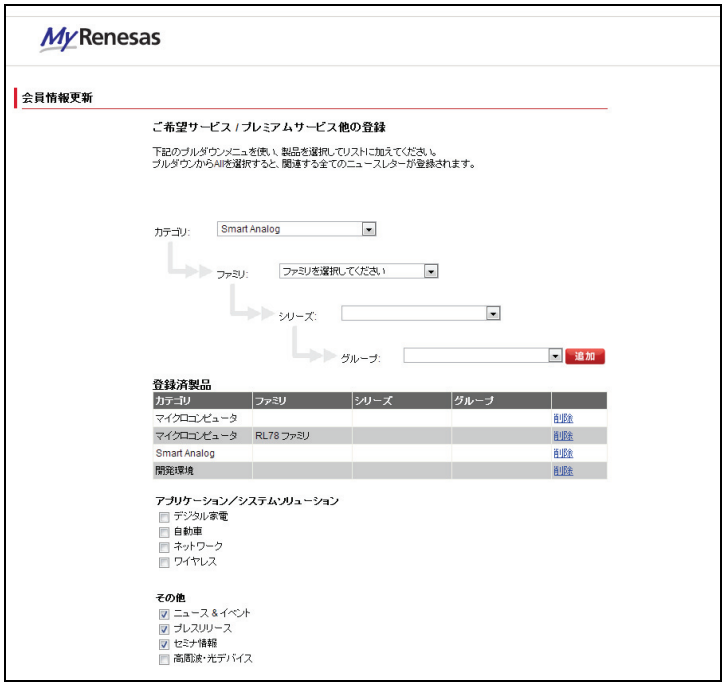
画面右上にある「すぐにお試し >>」をクリックしてください。

免責事項画面の最下部の「同意する」をクリックしてください。

なお、次の画面で My Renesas の ID とパスワードの入力画面となります。

Renesas VA を利用するには My Renesas の ID とパスワードが必要ですので、まだ ID をお持ちでない方は「My Reneas のご登録がまだのお客様」をクリックして、ID を新規登録(無料)してください。

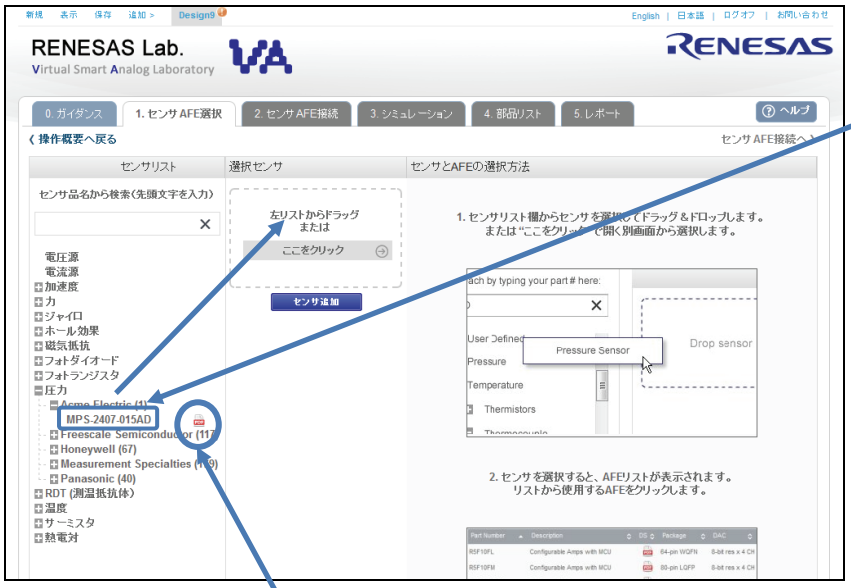
【注】ご登録の際、「ご希望サービス」でカテゴリから「Smart Analog」を選択してリスト追加すると、
Renesas VA、SA-Designer 等、Smart Analog に関連するニュースレターが登録されます。



ここをクリック

My Renesas の ID、パスワードを入力すると Renesas VA の画面となります。

右上の「シミュレーションスタート」をクリックしてください。



センサリストから

「圧力」
→「Acme Electric」
→「MPS-2407-015AD」

を選択して
選択センサへドラッグ

Part #　　: MPS-2407-015AD
Manufacturer　　: Acme Electric
Description　　: Pressure Sensor
Pressure Range (Pa) : 0 to 103k

ここをクリックするとセンサのデータシートをダウンロード可能

始めに Smart Analog に接続して使うセンサを選択します。

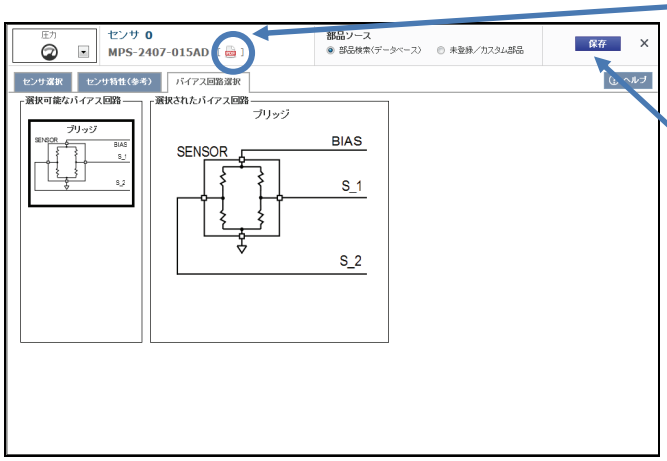
今回は圧力センサを使用しますので、センサリストから「圧力」を選択して、具体的型番を指定します。

センサリストから「圧力」を選択すると、Renesas VA に登録済みの圧力センサのベンダ一覧が表示されます。この中から「Acme Electric」を選択し、今回の対象として選んだメトロダインマイクロシステム社製圧力センサ「MPS-2407-015AD」を選択します。選択した「MPS-2407-015AD」を選択センサへドラッグします。

ここをクリックすると
バイアス回路選択画面が
表示されます。

次にセンサと Smart Analog IC を接続する方法を選択します。

選択センサから「Sensor Circuit アイコン」をクリックしてバイアス回路選択画面を表示します。



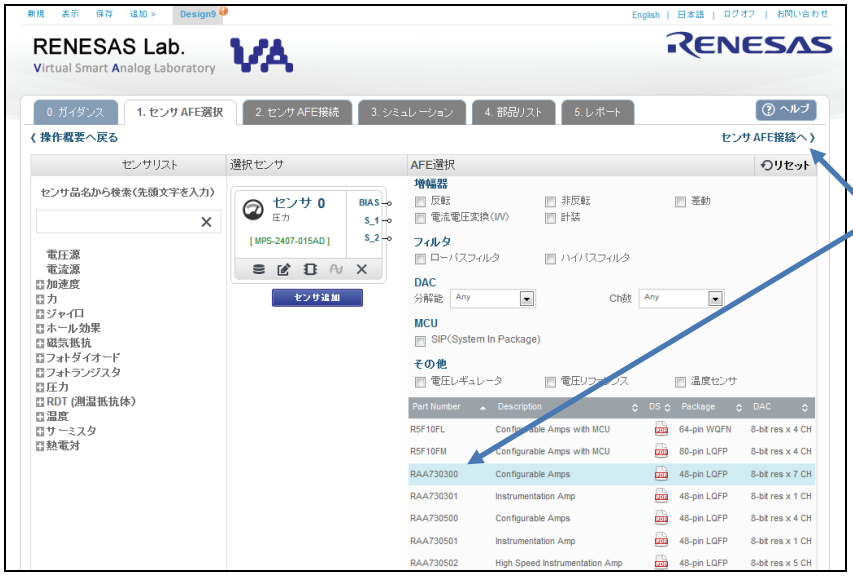
ここをクリックして
センサのデータシートをダウンロード

ロードバイアス回路が複数ある場合は
選択可能なバイアス回路一覧から選択し
右上の「保存」ボタンをクリック

画面左側上の「PDF アイコン」(MPS-2407-015AD の文字の隣)は MPS-2407-015AD のデータシートへのリンクです。クリックしてデータシートをダウンロードしておいてください。

「MPS-2407-015AD」と Smart Analog を接続する方法(バイアス回路)は今回はブリッジのみなので選択する必要はありませんが、複数ある場合は選択し保存ボタンをクリックします。

センサの選択・バイアス回路設定を確認して次に進みます。

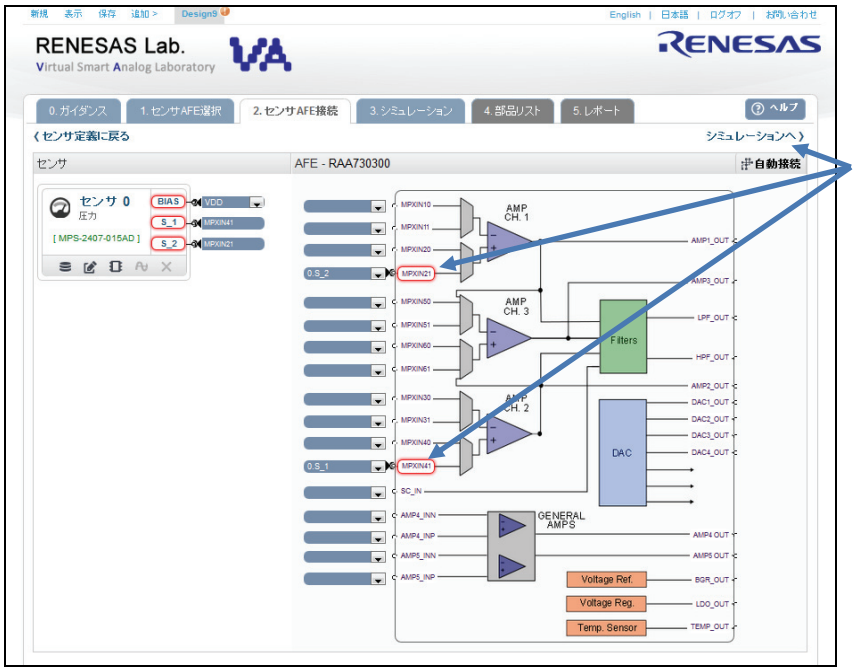


Part Number に「RAA730300」を選択し、
右上の「センサ AFE 接続へ>」ボタンを
クリック

シミュレーションで使用する Smart Analog の種類を選択します。

現在 Renesas VA では Smart Analog MCU(AFE と MCU を SiP で 1 つのパッケージに収めたもの)と Smart Analog IC(AFE 単体)をサポートしています。今回は Smart Analog IC(AFE 単体)を搭載した評価ボードを使いますので、Smart Analog IC(RAA730300)を選択します。

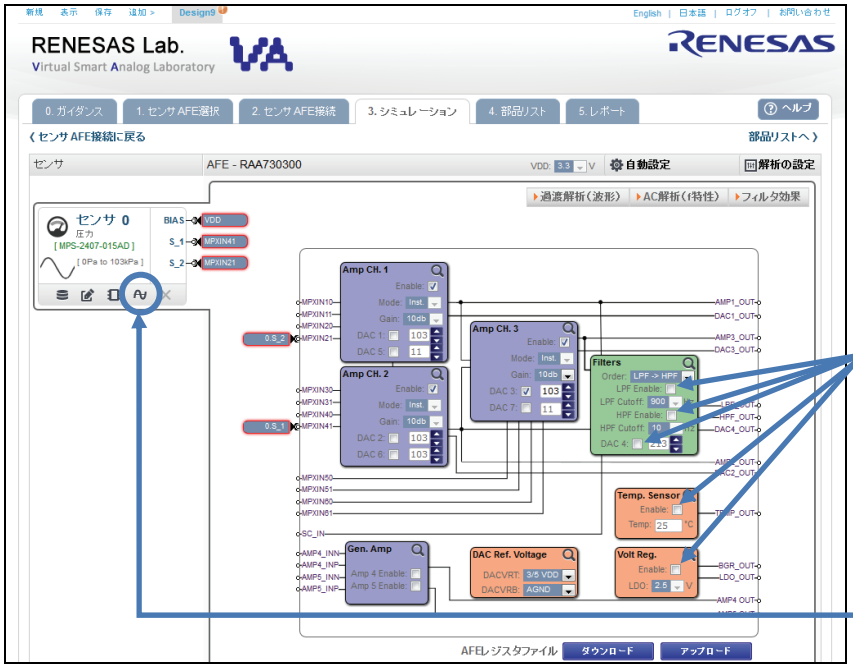
選択後、右上の「センサ AFE 接続へ>」をクリックしてください。



MPXIN21 端子にセンサ入力「0.S_2」を
MPXIN41 端子にセンサ入力「0.S_1」を
設定して右上の「シミュレーションへ＞」
ボタンをクリック

センサと Smart Analog IC との接続画面です。

ここで設定したセンサ - Smart Analog IC の接続情報に沿って、この後評価ボードで実際にセンサを接続します。今回は MPXIN21 と MPXIN41 にセンサを接続してください。（自動接続機能によりデフォルトでこれらに接続されている状態となります。）



Filters(LPF, HPF, DAC 4)、Temp Sensor、
Volt Reg.のチェックボックスを
非選択にして、各ブロックを OFF にする

ここをクリックすると
入力信号選択画面が
表示されます。

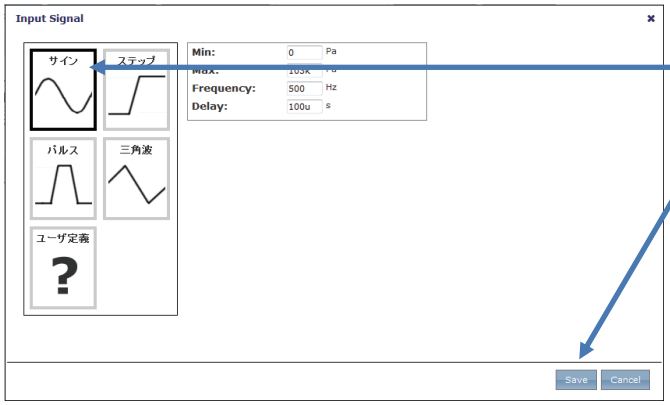
この画面にて Smart Analog 内部の回路の設定、センサへ入力する外部信号の設定、シミュレーションの種類などを設定します。

Smart Analog IC の回路定数は既にデフォルト値が入力されていて、このままシミュレーションすることができます。しかしながら今回は最も分かり易い構成とするため、AMP1,AMP2,AMP3 を使用した計装アンプ

のみを使用します。デフォルトで「ON」に設定されている「Filter (LPF, HPF, DAC 4)」、「Temp Sensor」、「Volt Reg.」の「Enable」も OFF にしてください。

次にセンサへ入力する信号を選択します。

センサから「Sensor Input アイコン」をクリックして入力信号選択画面を表示します。



Input Signal は「サイン」を選択
Min　　：0Pa　　(デフォルト)
Max　　：103kPa (デフォルト)
Frequency：500 Hz (デフォルト)
Delay　　：100us　(デフォルト)
上記設定後、右下の「Save」ボタンをクリック

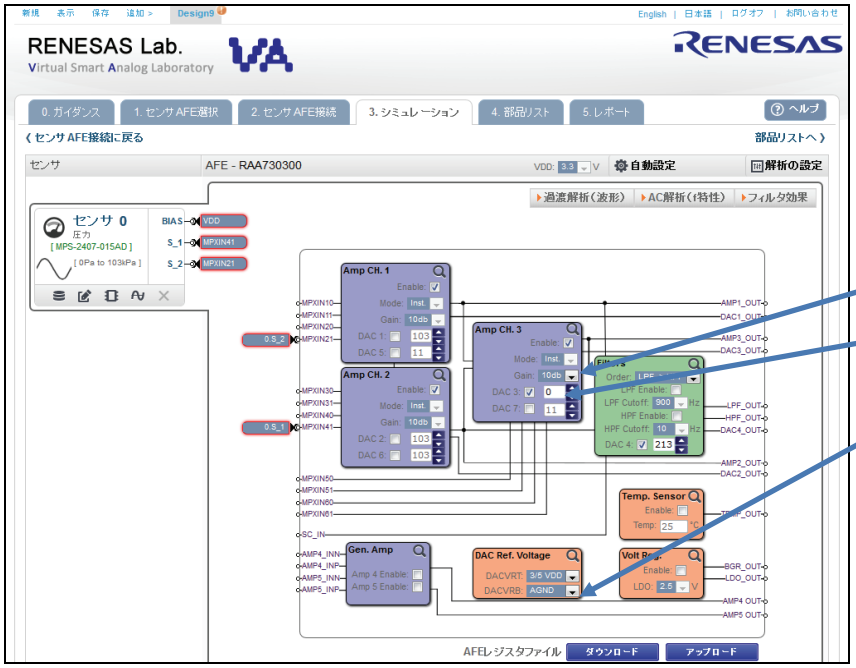
まずはサイン波で確認をしてみましょう。

＜psi とパスカル[Pa]の関係 ＞

今回のセンサのデータシートより 15psi の時、センサ出力は 130mV(Vdc:2V)である。

ここで 1psi=68.94757hPa なのでこのセンサの感度 S とすると

S＝130mV/(15×68.94757hPa)＝0.125699[mV/hPa]　となります。



10dB に設定(デフォルト)

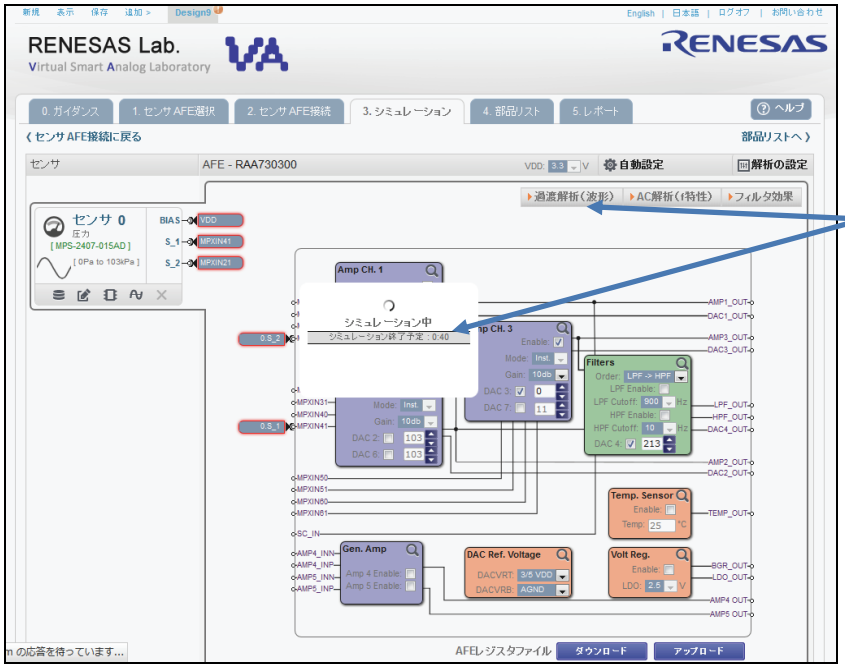
DAC3 の設定値を「0」に設定

DAC Ref. Voltage の DACVRB を AGND に設定(デフォルト)

最後に Smart Analog の AMP のゲインとバイアス電圧回路の設定をします。

計装アンプの Gain は Renesas VA が自動で最適化した数値(10dB)をそのまま使用し、バイアス電圧を設定する DAC3 の設定値を 0 にします。

さらに DAC のリファレンス電圧を設定する「DAC Ref Voltage」の下限側「DACVRB」を AGND に設定します。

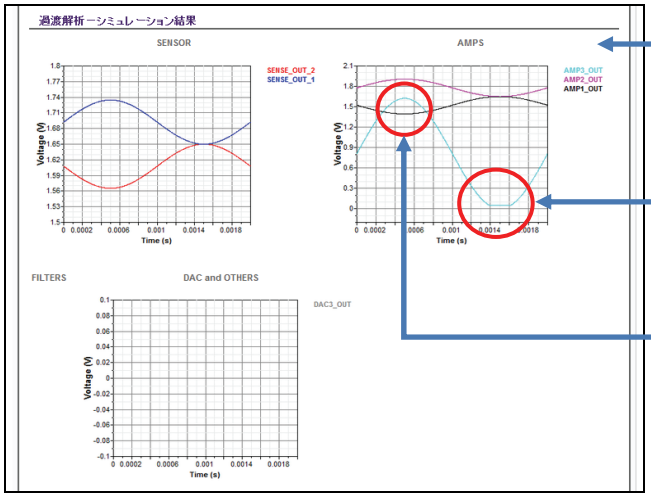


「過渡解析(波形)」ボタンをクリック

シミュレーションが開始され
終了予定時間が表示

ここまで設定が終わったら実際にシミュレーションを開始します。

上側中央の「過渡解析(波形)」ボタンをクリックするとシミュレーションが開始されます。



回路画面下側にシミュレーション波形が
表示される

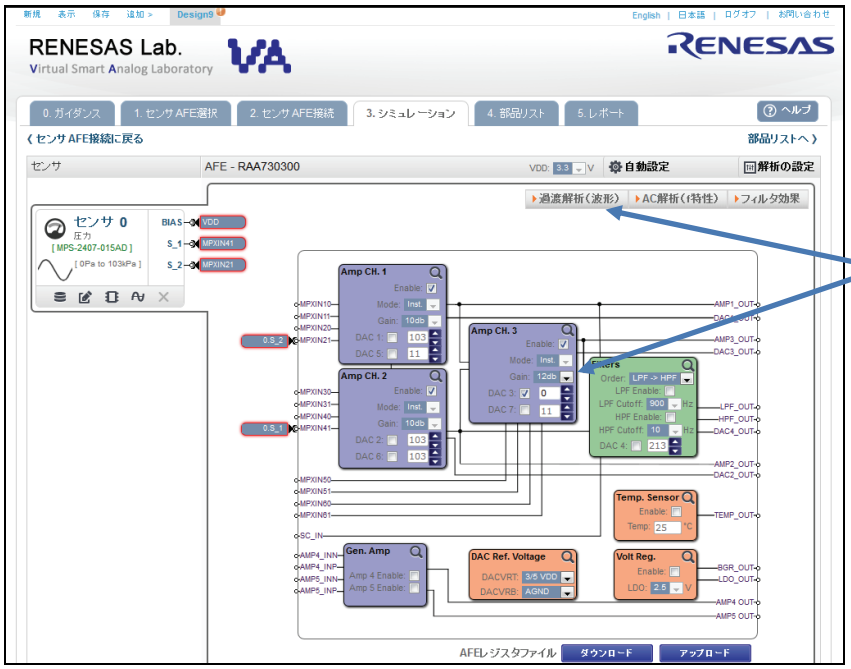
0V 以下の信号が出力されず
「クリップ」している事を確認

AMP3(水色)の出力が約 1.6V と
なっている

シミュレーションが終了すると「過渡解析(波形)」ボタンのすぐ下にシミュレーション結果のリンク(Result)、また回路画面下側にシミュレーション結果の波形が表示されます。

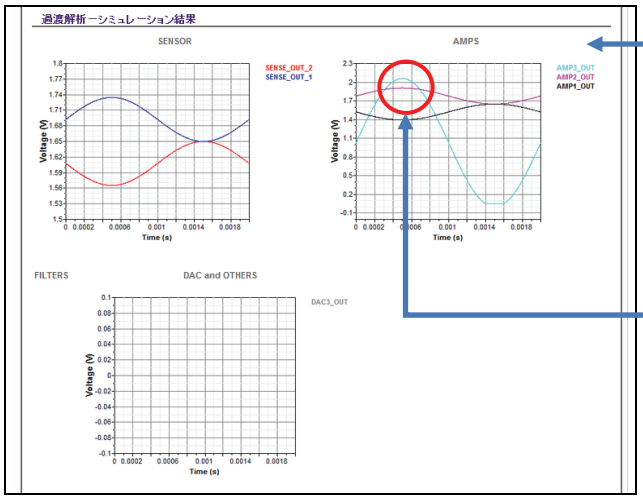
シミュレーションの結果から、AMP3 の出力の下側が「クリップ」されていることが分かります。
これは AMP3 のバイアス電圧を 0V に設定したため、AMP3 の動作点が 0V になっている、という理由です。

この時の AMP3 出力が約 1.6V と小さいので、ゲインを少し上げてみます。



AMP3 の Gain を 12dB に設定する
過渡解析ボタンを再クリック

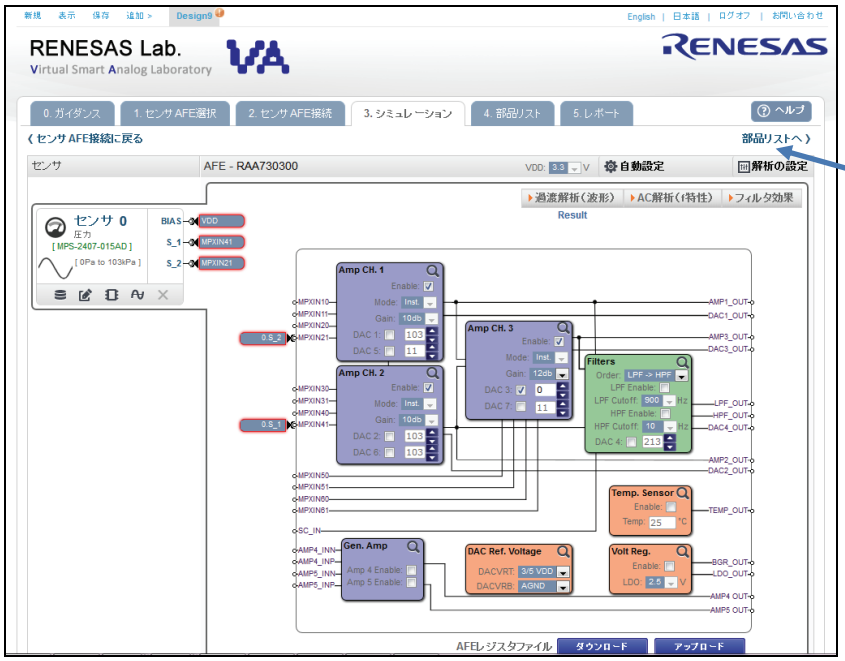
計装アンプの Gain を 12dB に設定してもう一度「過渡解析」ボタンをクリックしてシミュレーションします。



シミュレーション波形を確認する

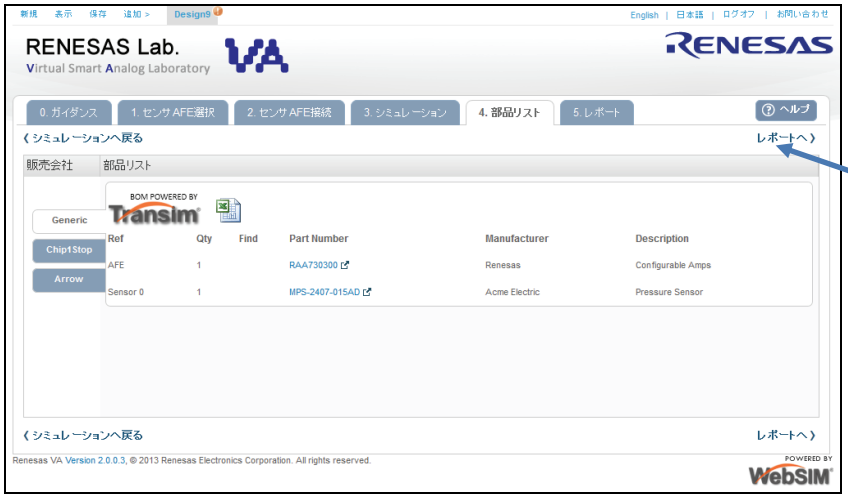
AMP3(水色)の出力が約 2.0V となっている

シミュレーション結果から、AMP3 の出力は約 2.0V まで増幅されていることが確認できます。
十分な出力が得られている事が確認できましたので、このパラメータを使う事とします。



右上の「部品リストへ＞」ボタンをクリック

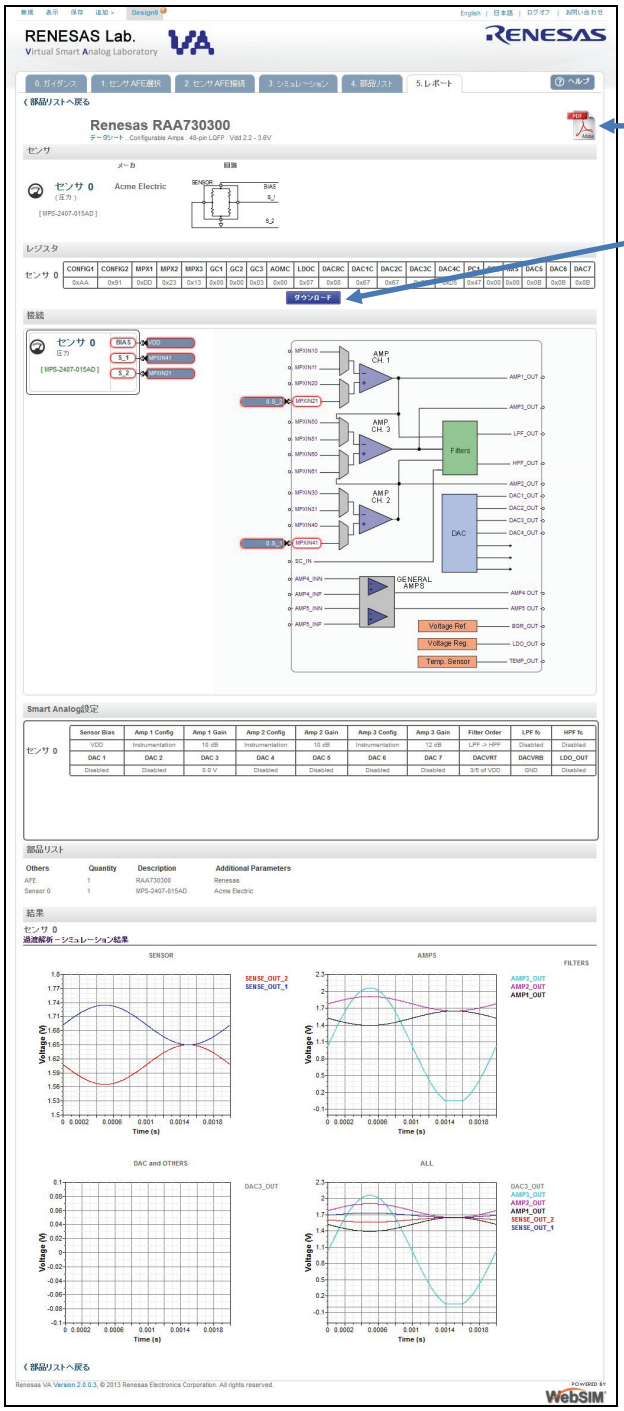
右上の「部品リストへ＞」ボタンをクリックして次の画面へ進んでください。



右上の「レポートへ＞」ボタンをクリック

この画面は今回のシミュレーションで使用した部品リストの確認画面です。

右上の「レポートへ＞」ボタンをクリックして次の画面へ進んでください。



クリックしてこの画面の pdf を保存

クリックして
Smart Analog 回路パラメータ
（AFE レジスタファイル）を保存

今回のシミュレーションの結果が表示されます。

右上の「pdf アイコン」をクリックするとこの画面の pdf「Renesas_RAA730300.pdf」がダウンロードできます。

画面上側の「レジスタ」の項目の「ダウンロード」ボタンをクリックして、
「Renesas.Register.RAA730300.Sensor-XX_XX_XXXX.ini」をダウンロードして PC に保存してください。

このファイルが今回シミュレーションした Smart Analog のアナログ回路設定値のパラメータ（AFE レジスタファイル）となります。

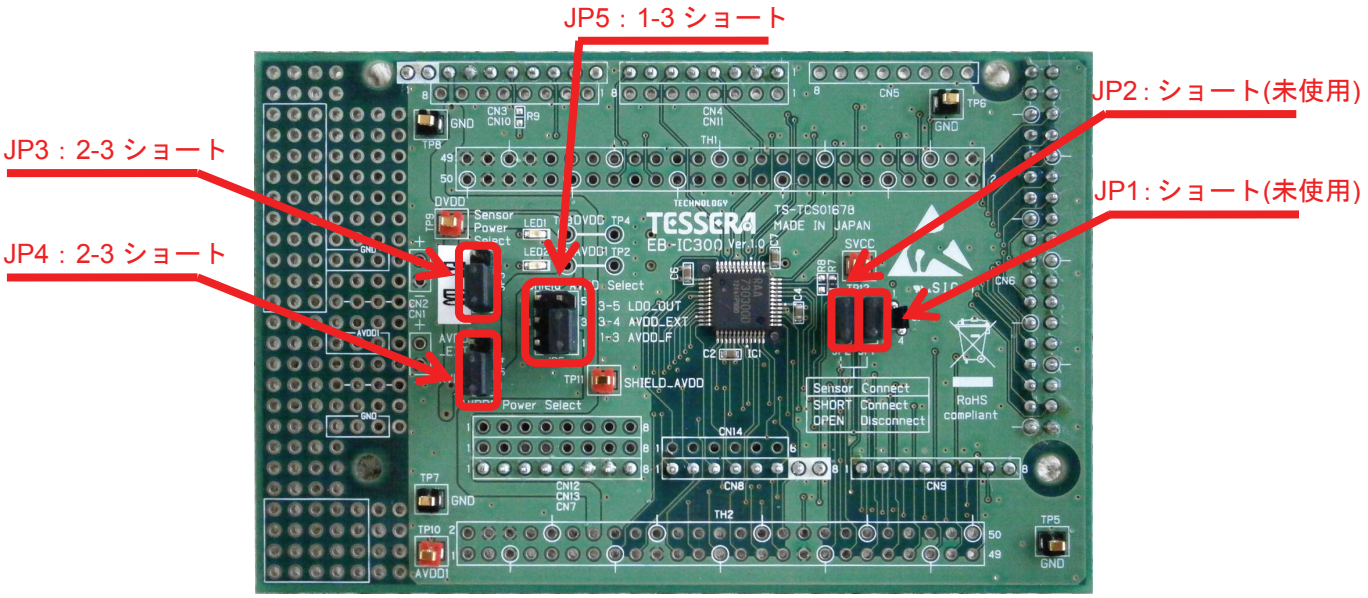
シミュレーションは以上で終了です。

次に評価ボードでの開発に入ります。

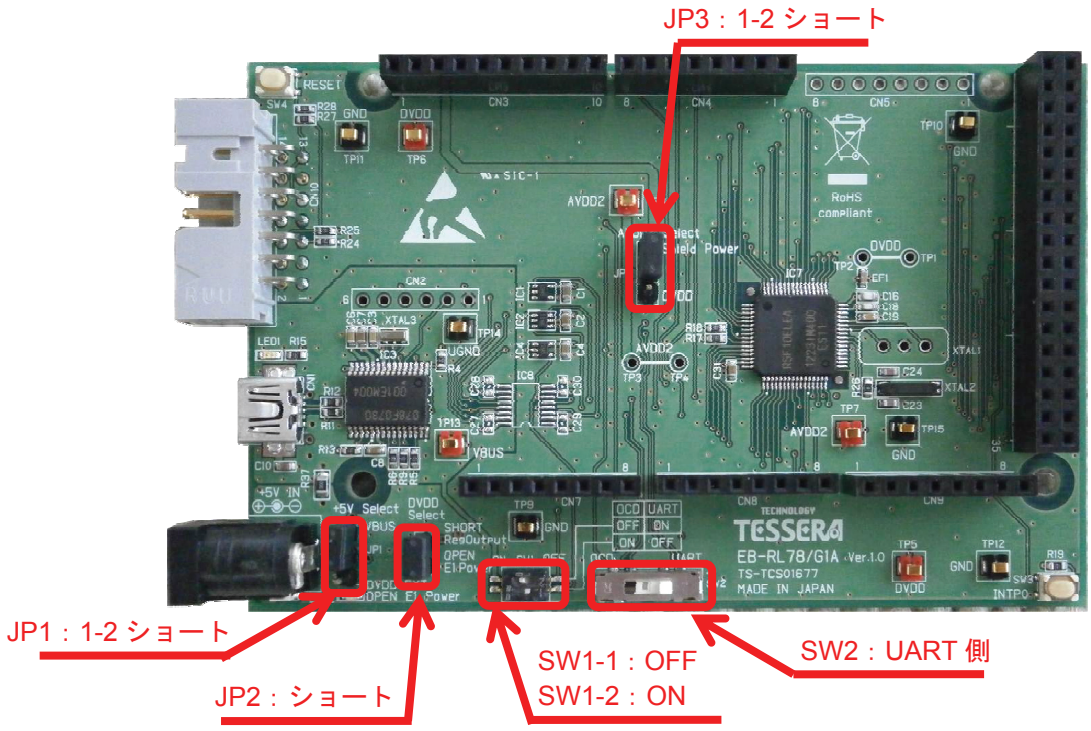
6.　センサ、LED 回路を結線しよう

センサと LED の回路をブレッドボードに作り、評価ボードに取り付けましょう。また、本資料と併せて、TSA-IC300 のユーザーズマニュアル、MPS-2400 シリーズのデータシートもご参照ください。

6.1　　評価ボードのショートピン、ＳＷを設定する



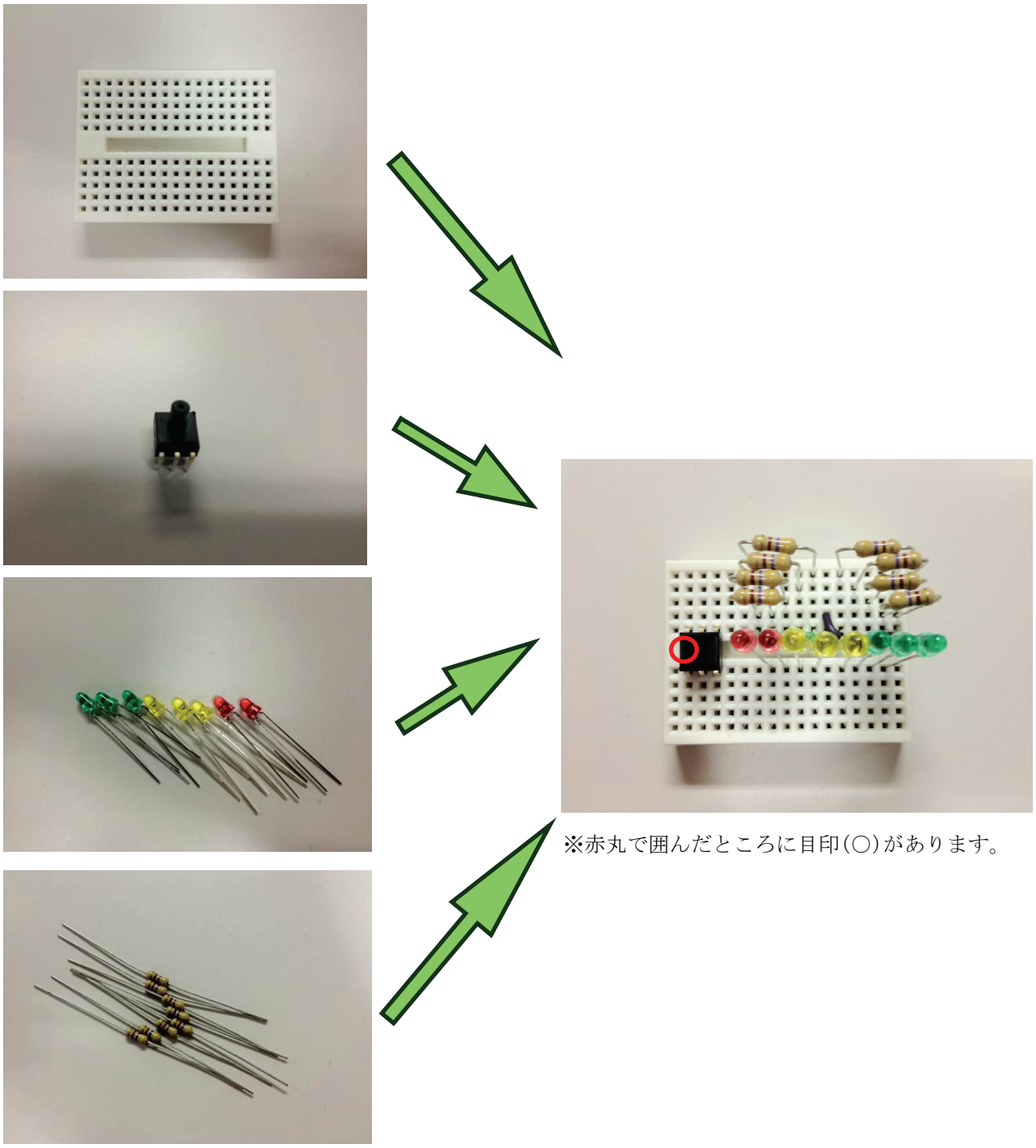
EB-IC300 設定



EB-RL78/G1A 設定

6.2　ブレッドボードにセンサ、LED 回路を組む

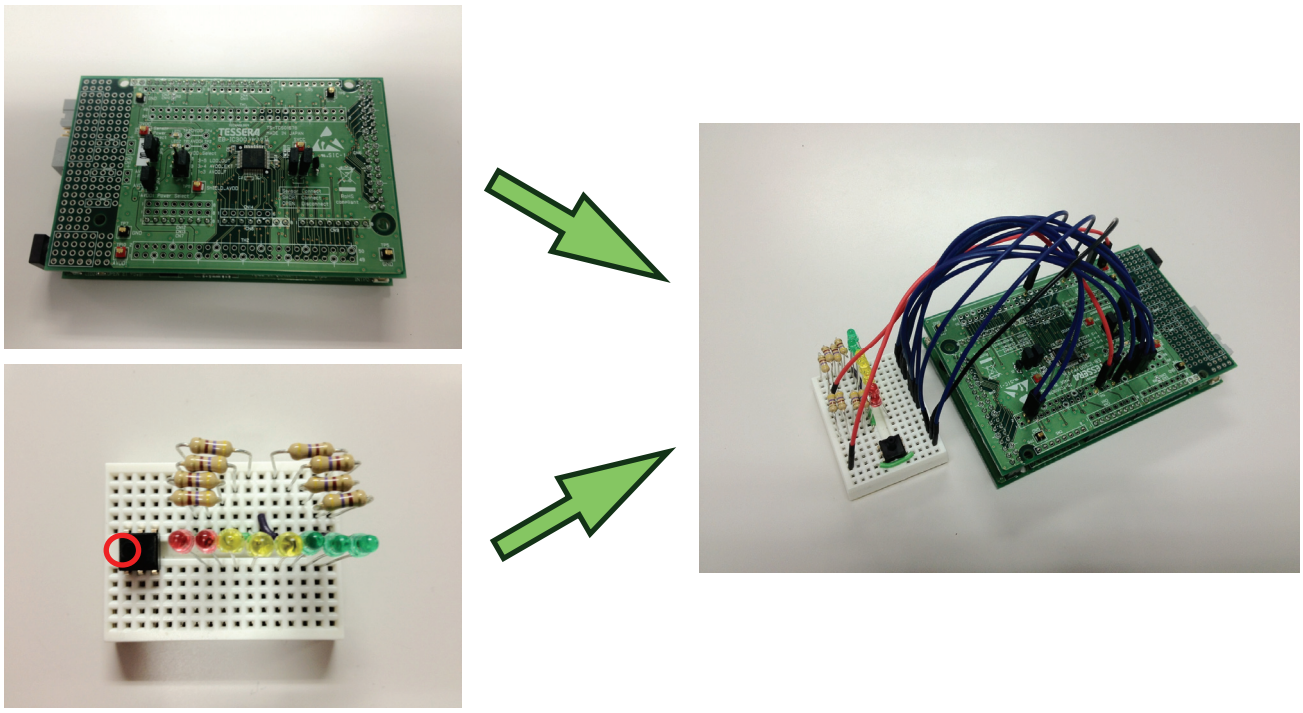
「ブレッドボード」に「MPS-2407-015AD」+「LED 回路」を構成してください。



※赤丸で囲んだところに目印(○)があります。

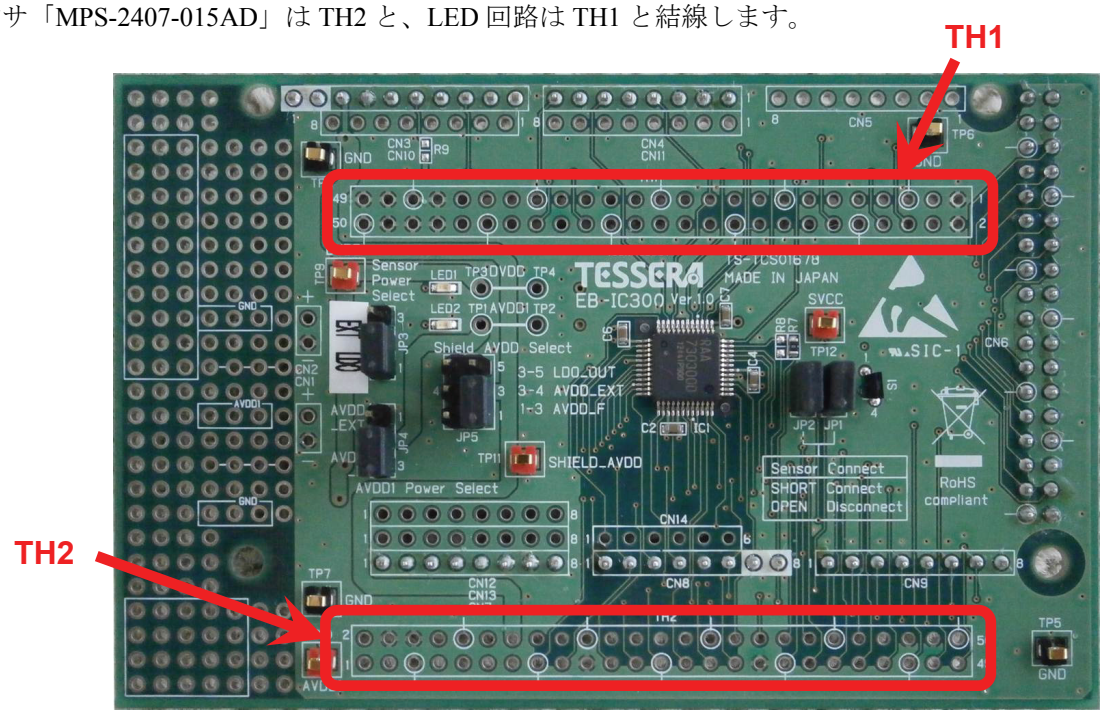
6.3　出来上がったブレッドボードを評価ボードに結線する

出来上がったブレットボードと評価ボードを結線してください。



ブレッドボードとの結線には、EB-IC300 の TH1 と TH2 を使用します。

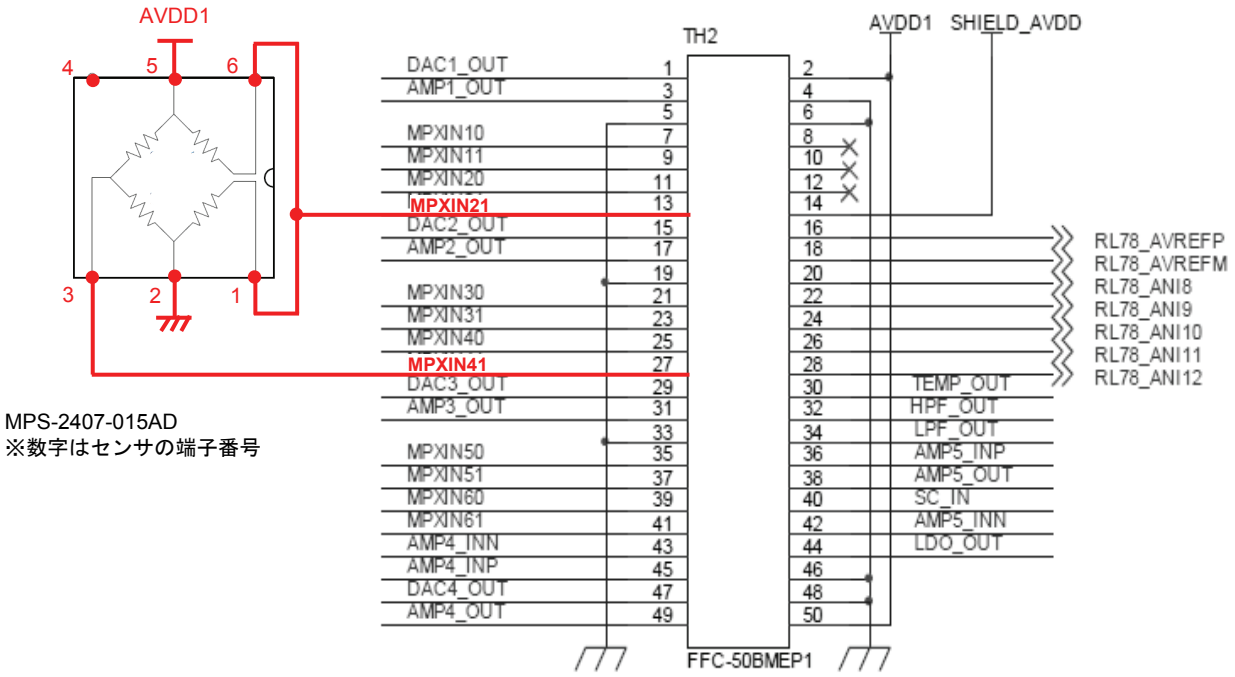
センサ「MPS-2407-015AD」は TH2 と、LED 回路は TH1 と結線します。



EB-IC300 の TH1 と TH2

センサ「MPS-2407-015AD」と TH2 は以下のように結線してください。

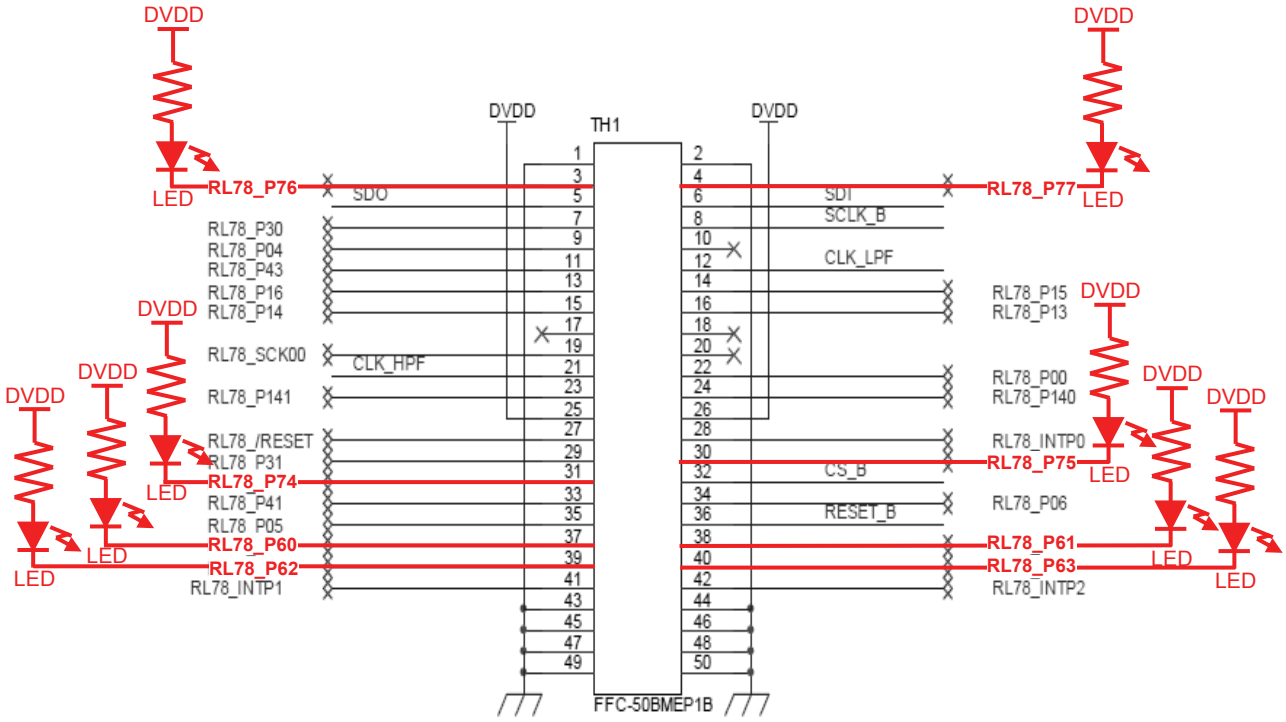
※本資料では、センサ「MPS-2407-015AD」の端子番号 1～6 を下図のように処理します。



センサ「MPS-2407-015AD」と TH2 との結線図

LED 回路と TH1 は以下のように結線してください。

※本資料では、P60, 61, 62 を緑色 LED、P63, 74, 75 を黄色 LED、P76,77 を赤色 LED に接続します。



LED 回路と TH1 との結線図

以上で評価ボードの準備は完了です。

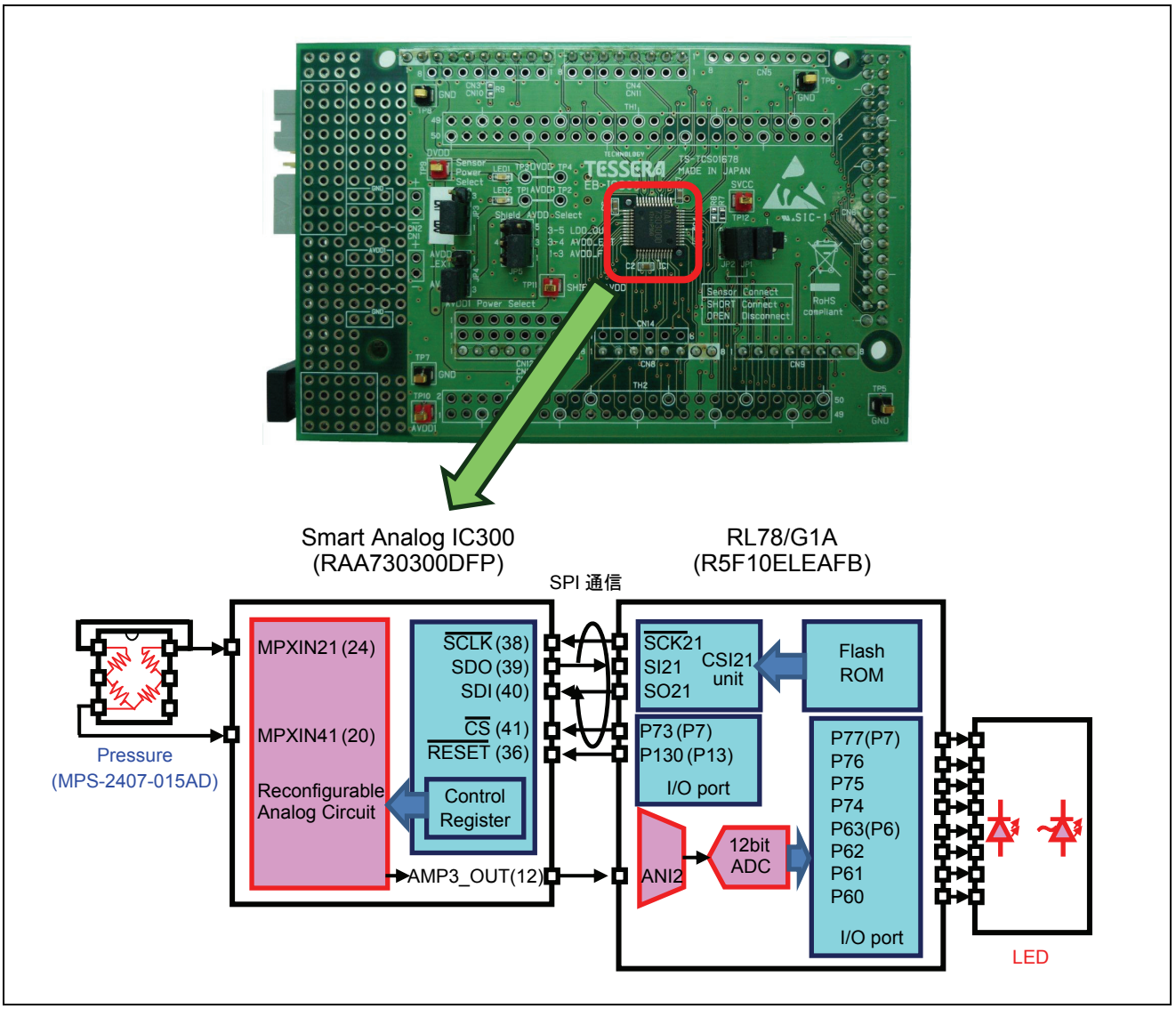
次にこのボードを使ってプログラムを作ります。

7. SA-Designer で AFE のパラメータを取り込んで、CubeSuite+でプログラムを作ってみよう！

5.で作った Renesas VA での Smart Analog IC の AFE レジスタファイル、6.で作ったセンサ、LED 回路と接続した評価ボードを使って、評価ボード上で動かすためのプログラムを作ってみましょう。

7.1 評価ボードに搭載されている IC とマイコンの接続状態とこのパートで作るプログラムの内容

実際にプログラムを開発する前に、評価ボードに搭載されている Smart Analog IC とマイコンの構成と端子情報について説明します。



Smart Analog IC 300(RAA730300DFP)と RL78/G1A(R5F10ELEAFB)との端子接続と内部回路は上の図のような関係になっています。

Smart Analog IC 300 と RL78/G1A は SPI で接続されており、Smart Analog IC300 は RL78/G1A から SPI で制御されて動作します。

5.で作った AFE レジスタファイルには Smart Analog IC300 のレジスタ設定値が格納されています。このレジスタ設定値を RL78/G1A の Flash ROM に予め保存しておき、RL78/G1A から SPI 経由で Smart Analog IC300 へ転送して、Smart Analog IC300 を動作させるようにプログラムを作ります。

また、Smart Analog IC300 を介してセンサ信号を RL78/G1A の ADC で受け取り、A/D コンバートした結果に応じて LED ポートを制御して LED を光らせる部分のプログラムも作ります。

7.2　SA-Designer をインストールしよう！

プログラムを組む前にまずは SA-Designer をインストールしましょう。

7.2.1　そもそも SA-Designer とは？

SA-Designer は、Smart Analog 製品のアナログフロントエンド回路を設計して、その回路データを C ソースコードとして生成するツールです。

従来、センサ動作に合わせたアナログフロントエンド回路の部品調達や部品変更に伴ってハードウェアの準備に時間がかかっていましたが、Smart Analog と SA-Designer であればそれらが不要となり、SA-Designer による PC 画面でのマウス操作だけで簡単に回路設計をすることができるようになります。

また、センサの選定に利用できる Web シミュレータ「Renesas VA」の AFE レジスタファイルをインポートできるほか、ユーザプログラム統合開発環境「CubeSuite+」との連携もできるため、センサの選定から、回路設計、およびプログラム開発までをすべてソフトウェアで完結でき、効率よくシステム開発を進めることができます。

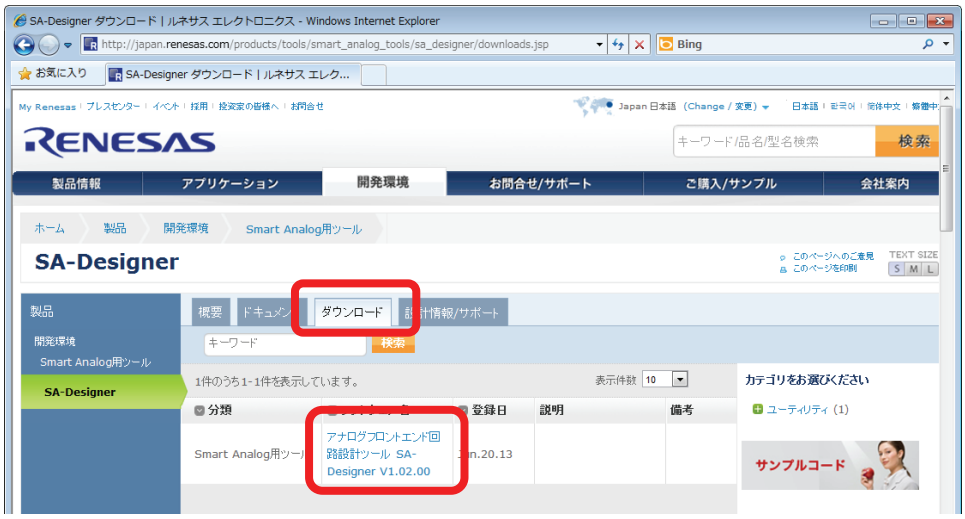
SA-Designer のサポートする機能

- 回路ブロック図スタイルの GUI による回路設計・カスタマイズ
- 作成した回路データ（レジスタ値等）の一覧表示、保存、復元
- Web シミュレータ「Renesas VA」で作成した回路デザイン(AFE レジスタファイル)の読み込み
- 回路設計用の C ソースファイル生成
- 統合開発環境 CubeSuite+へのプロジェクト登録、 CubeSuite+でのビルド/ダウンロード/デバッグの実行
- チュートリアル搭載

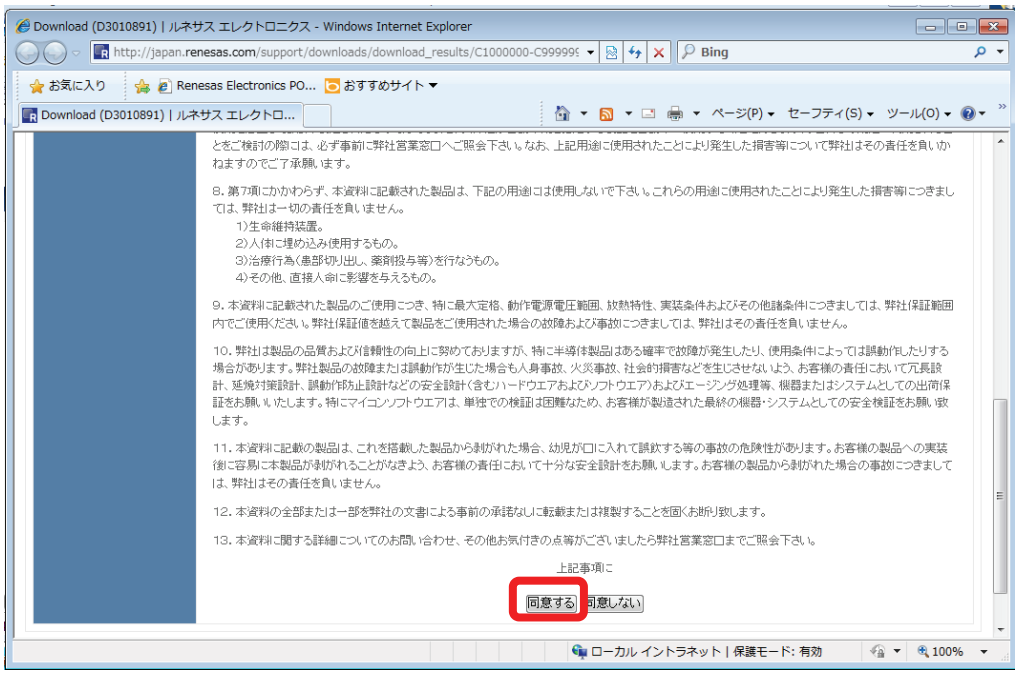
7.2.2　SA-Designer のインストール

web ブラウザで Renesas の開発環境の「Smart Analog 用ツール」の「SA-Designer」ページにアクセスしてください。

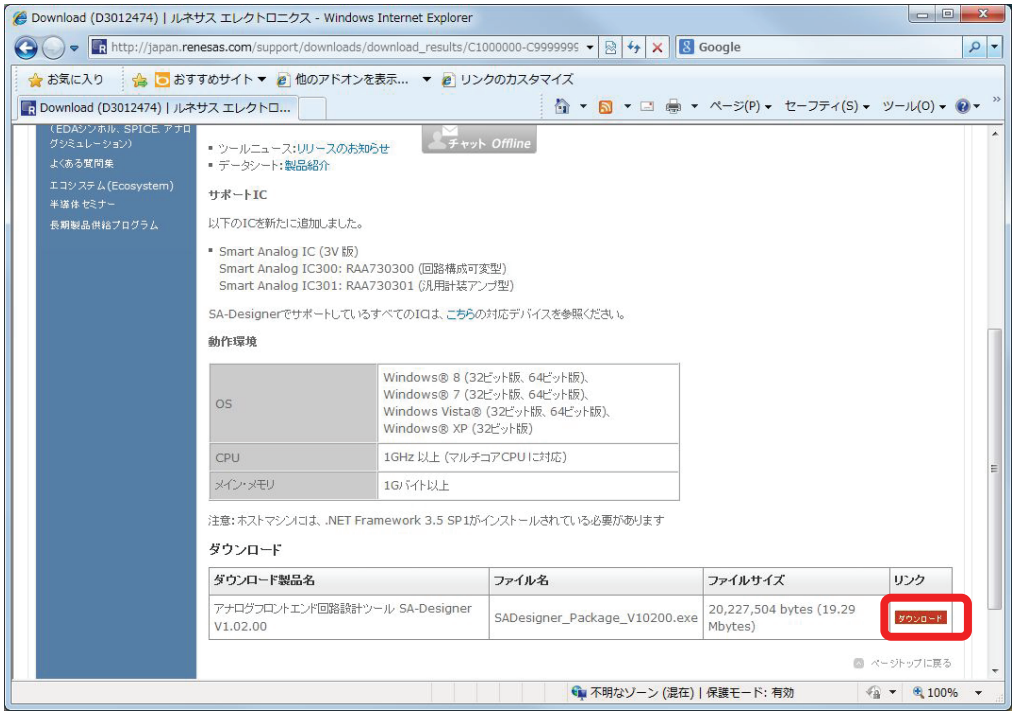
- SA-Designer
  http://japan.renesas.com/sa_designer
  Renesas の TOP ページ → 開発環境 → Smart Analog 用ツール→ SA-Designer



「ダウンロード」タブの「アナログフロントエンド回路設計ツール SA-Designer Vx.xx.xx」をクリックしてください。



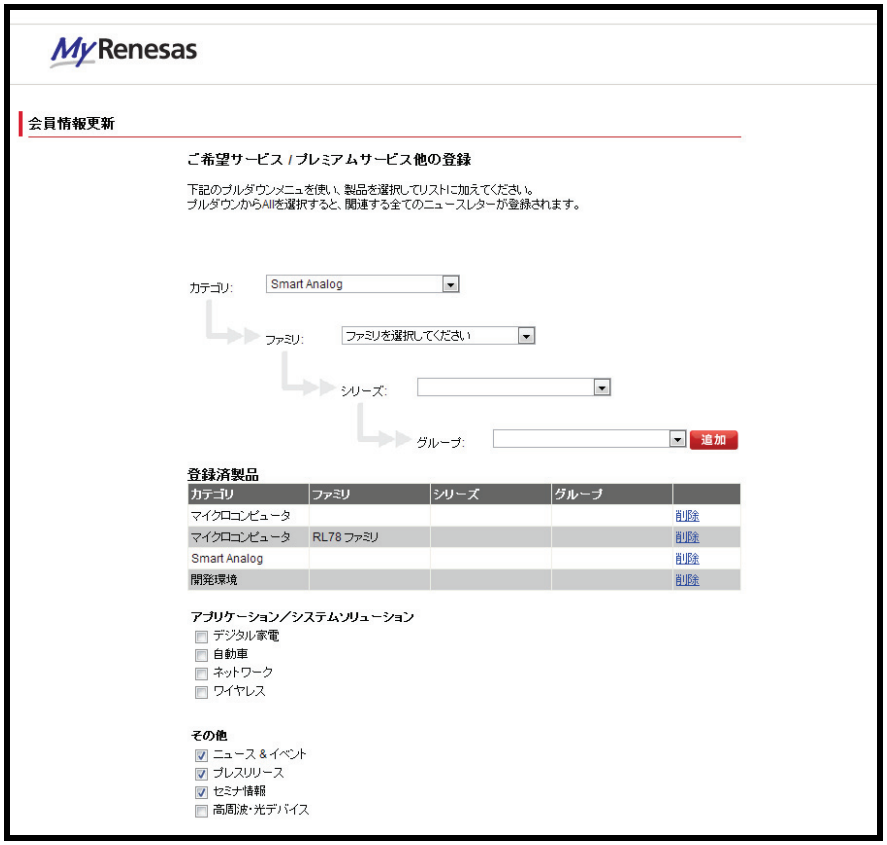
免責事項画面が表示されますので「同意する」をクリックしてください。



画面右下の「ダウンロード」をクリックしてファイルをパソコンに保存してください。

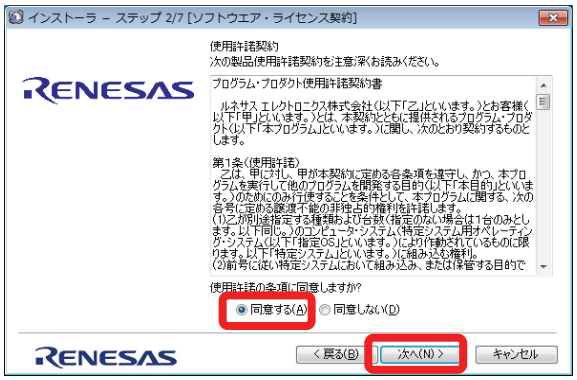
この後、My Renesas の ID とパスワードの入力画面が表示されますので、ID とパスワードを入力してください。まだ ID をお持ちでない方は「My Reneas のご登録がまだのお客様」をクリックして、ID を新規登録(無料)してください。

【注】　ご登録の際、「ご希望サービス」でカテゴリから「Smart Analog」を選択してリスト追加すると、Renesas VA、SA-Designer 等、Smart Analog に関連するニュースレターが登録されます。


MyRenesas
会員情報更新
ご希望サービス / プレミアムサービス他の登録
下記のプルダウンメニュを使い、製品を選択してリストに加えてください。
プルダウンからAllを選択すると、関連する全てのニュースレターが登録されます。
カテゴリ:
Smart Analog
ファミリ:
ファミリを選択してください
シリーズ:
グループ:
追加
登録済製品
カテゴリ
ファミリ
シリーズ
グループ
マイクロコンピュータ
削除
マイクロコンピュータ
RL78 ファミリ
削除
Smart Analog
削除
開発環境
削除
アプリケーション/システムソリューション
デジタル家電
自動車
ネットワーク
ワイヤレス
その他
ニュース & イベント
プレスリリース
セミナ情報
高周波・光デバイス

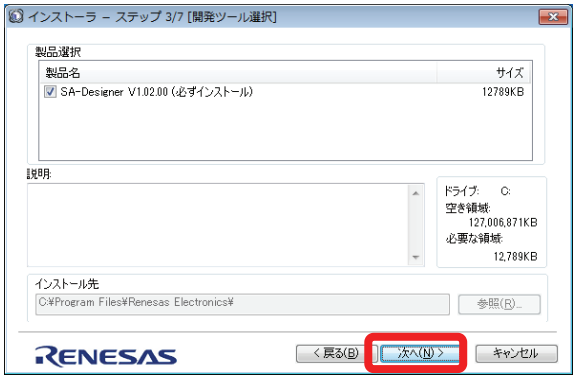
ダウンロードが完了したら、ダウンロードしたファイル「SADesigner_Package_VXXXXX.exe」をダブルクリックしてください。インストーラが起動します。

「次へ」をクリックしてください。

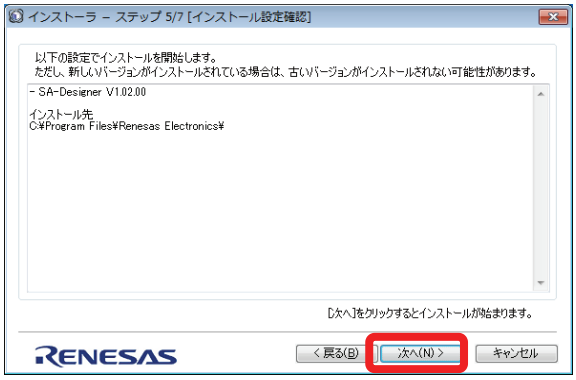


本ソフトウェアの使用許諾契約書が表示されます。

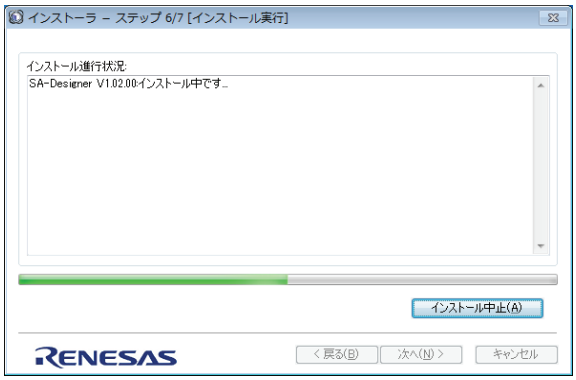
「同意する」をチェックして、「次へ」をクリックしてください。



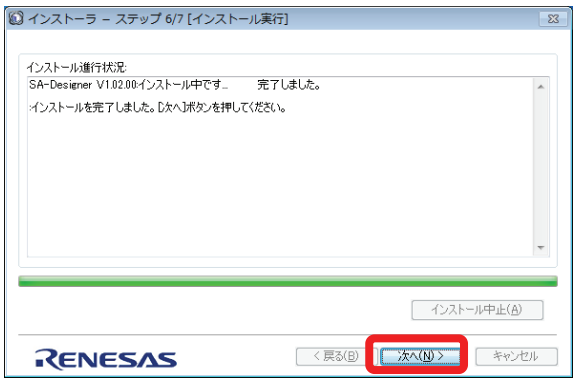
SA-Designer のインストール先が表示されますので、内容を確認して「次へ」をクリックしてください。



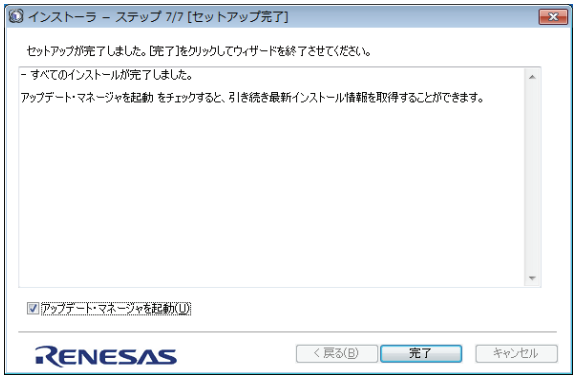
「次へ」をクリックしてください。



インストールが開始されます。



「次へ」をクリックしてください。



「完了」をクリックしてください。これで SA-Designer のインストールは終了です。

7.3　CubeSuite+をインストールしよう！

CubeSuite+のインストールを行います。

既に CubeSuite+をインストール済みの方はここは読み飛ばして、6.4 に進んでください。

【注】 Ver1.02 未満をお使いの方は最新版へアップグレードしてください。
SA-Designer と CubeSuite+の連動は Ver1.02 以上が必要です。

7.3.1　CubeSuite+のインストール方法について

CubeSuite+の機能やサポート環境については、下記ページを参照ください。

- 統合開発環境 CubeSuite+
  http://japan.renesas.com/cubesuite+
  Renesas の TOP ページ → 開発環境 → 統合開発環境(IDE) → 統合開発環境 CubeSuite+

インストールについては、上記ページの「ダウンロードタブ」から「【無償評価版】統合開発環境 CubeSuite+ Vx.xx.xx」をダウンロードしてインストールしてください。

【注】 Ver1.02 以上の最新版をインストールしてください。

7.4　SA-Designer を起動する

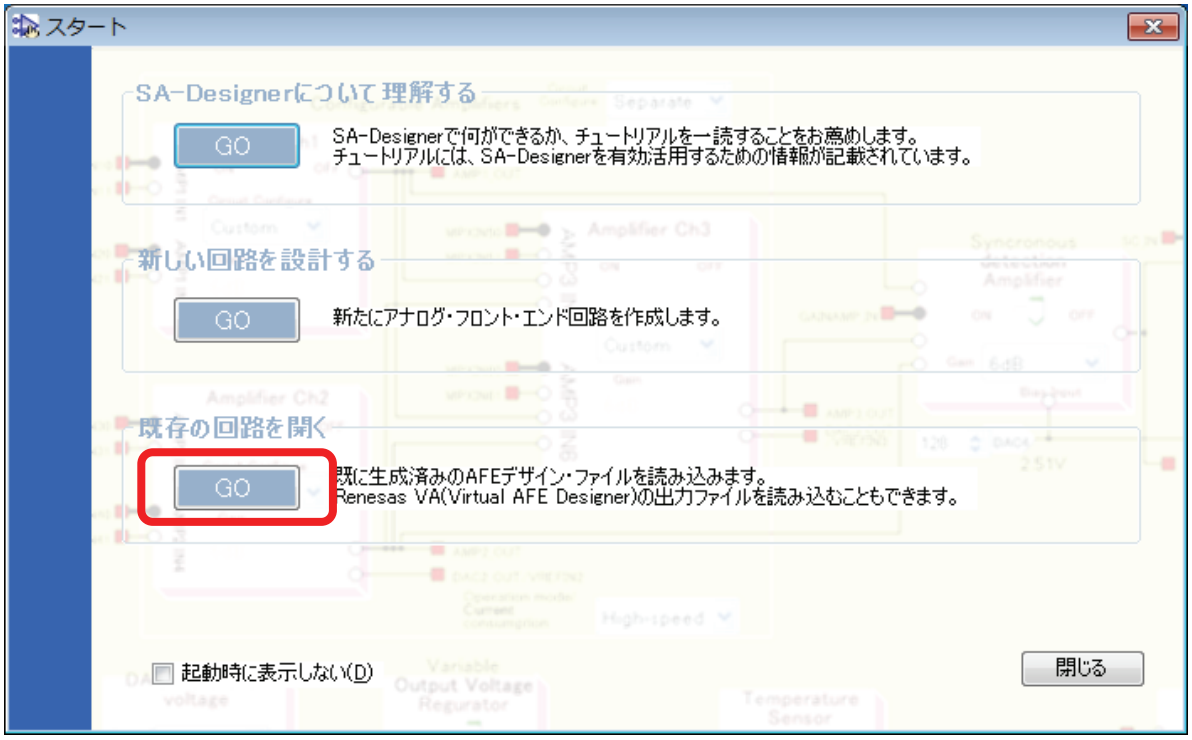
では実際にプログラムを作ってみましょう。まずは SA-Desinger を起動させます。

Windows のスタートメニューから SA-Designer を起動します。

【注】 ver1.02.00(Smart Analog IC300 シリーズ対応版)以上の最新版をインストールしてください。

[Renesas Electronics Utilities]→[スマート・アナログ・ツール]→[SA-Designer]

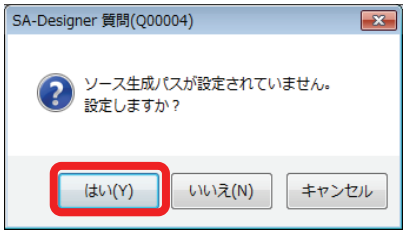
SA-Designer のスタート画面で、「既存の回路を開く」の GO ボタンをクリックしてください。



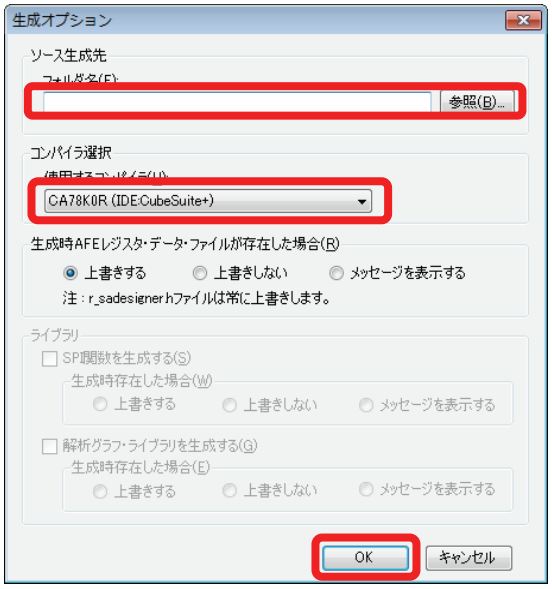
7.5　Renesas VA で作成した AFE レジスタファイル(.ini ファイル)を読み込ませる

Renesas VA でシミュレーションした Pressure (MPS-2407-015AD)の AFE レジスタファイル（Renesas.Register.RAA730300.Sensor-XX_XX_XXXX.ini ファイル）を開いてください。

起動初回は生成パスの指定を聞かれますので、任意の場所を設定してください。

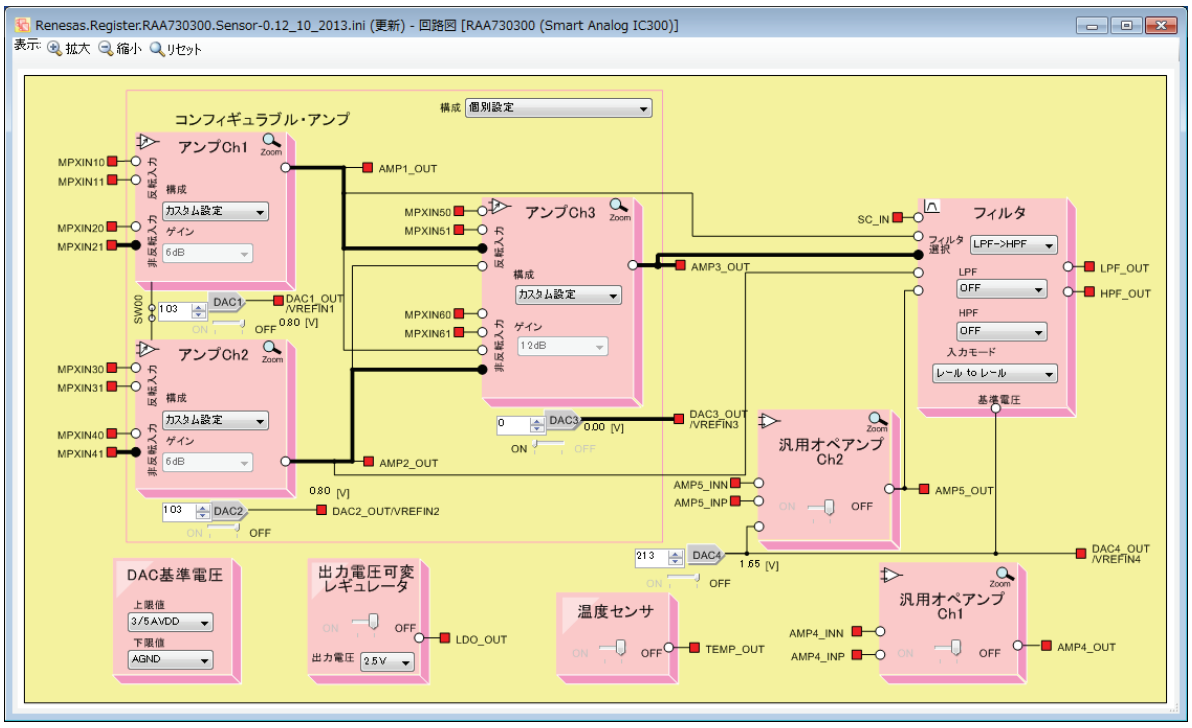


「はい」をクリックしてください。



ソース生成先には任意の場所を設定してください。使用するコンパイラには「CA78K0R (IDE: CubeSuite+)」を選択して OK ボタンをクリックしてください。

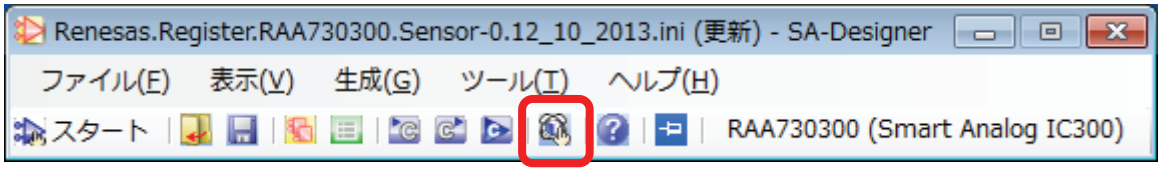
SA-Designer の回路設計画面が以下のように開きます。



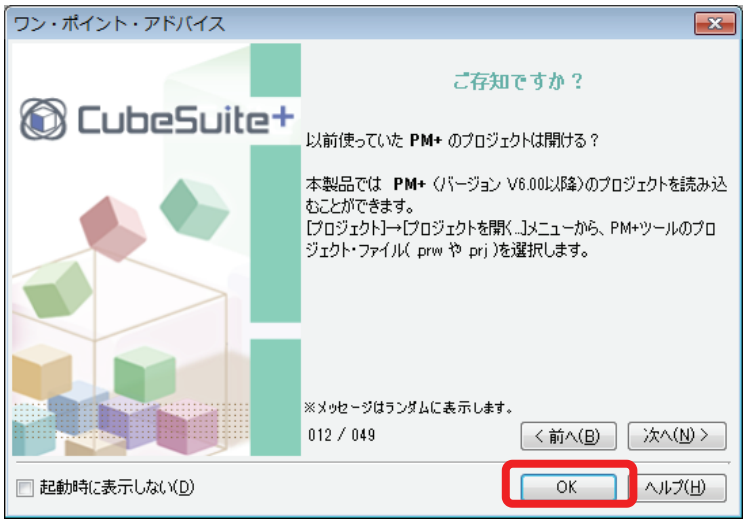
7.6　CubeSuite+を起動する

SA-Designer を CubeSuite+と連携させます。

以下の赤枠で囲ったボタンをクリックして CubeSuite+を起動してください。

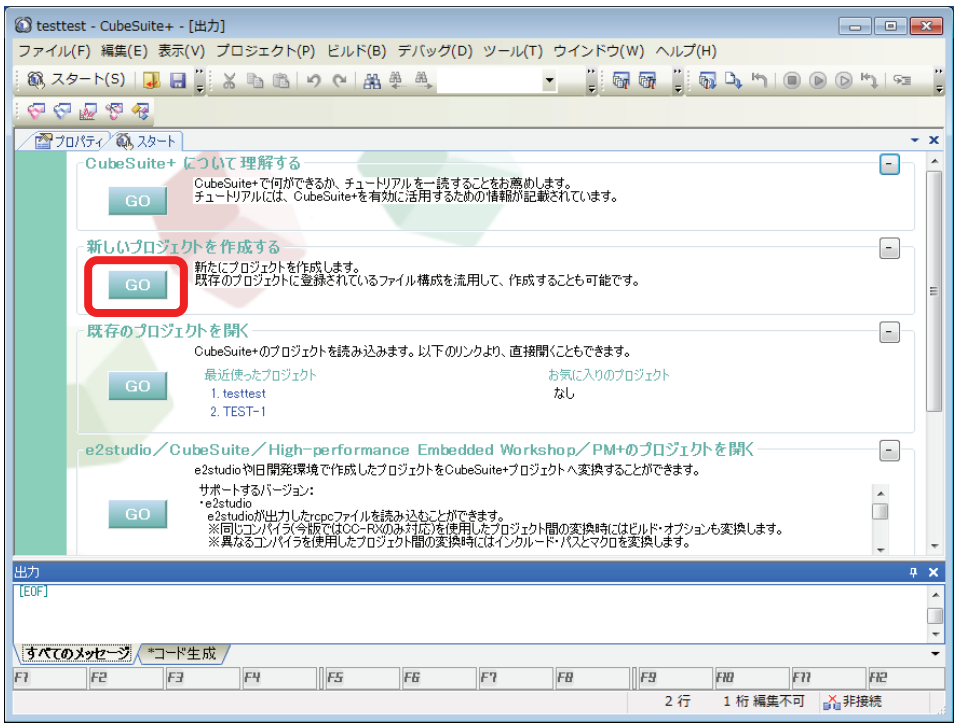


今回のプログラム用のプロジェクトを作成します。

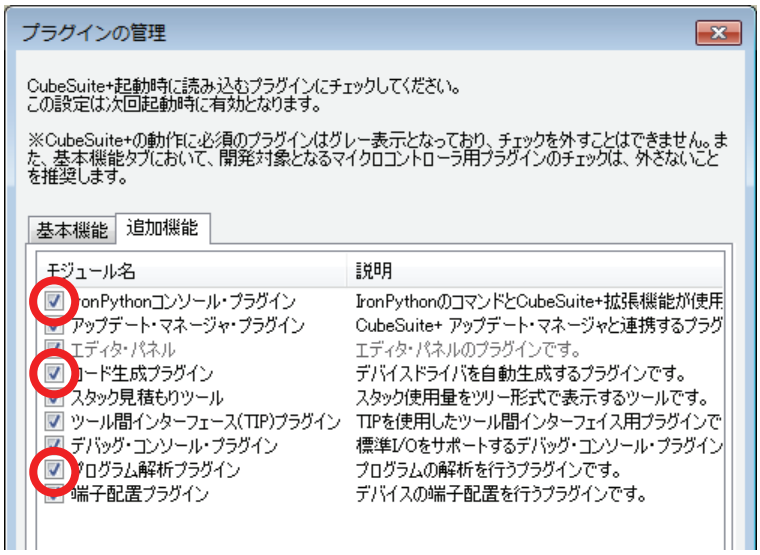


ＯＫボタンを押してください。

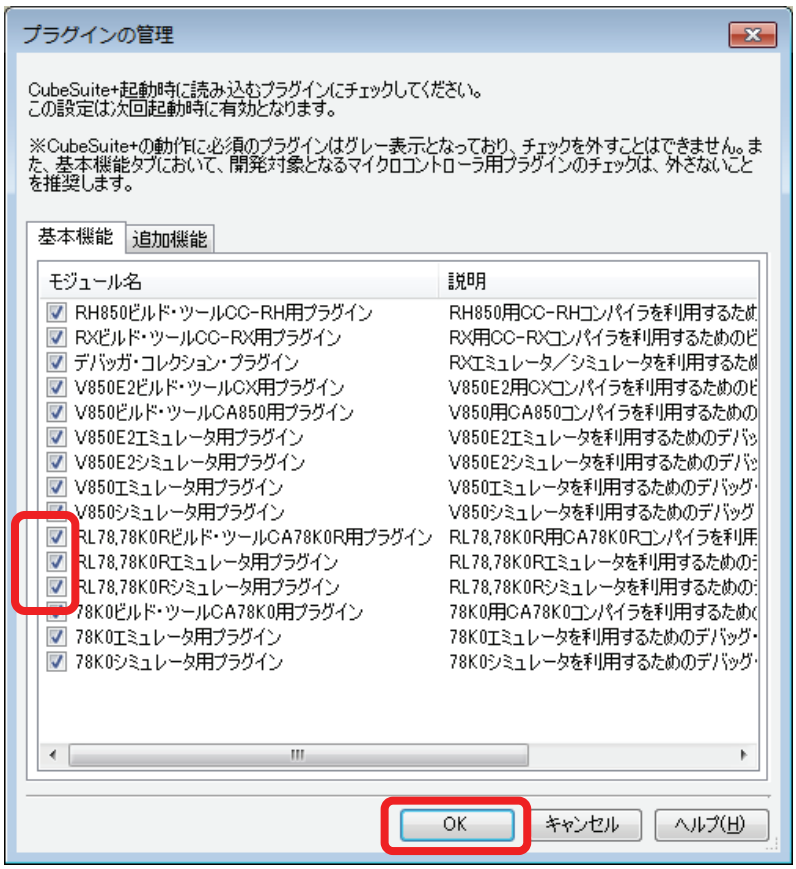
スタート画面から「新しいプロジェクトを作成する」の GO ボタンをクリックしてください。



【注】 CubeSuite+の起動時に、上記スタート画面が表示されない場合はプラグインの設定が原因です。
「ツール」メニューから「プラグインの管理」を選択し、下記の通り設定してください。
スタート画面が表示される方は、次頁の「プロジェクト作成」画面へ進んでください。



CubeSuite+の起動時に読み込むプラグインをチェックします。「追加機能」タブから、「IronPython コンソール・プラグイン」、「コード生成プラグイン」、「プログラム解析プラグイン」にチェックしてください。

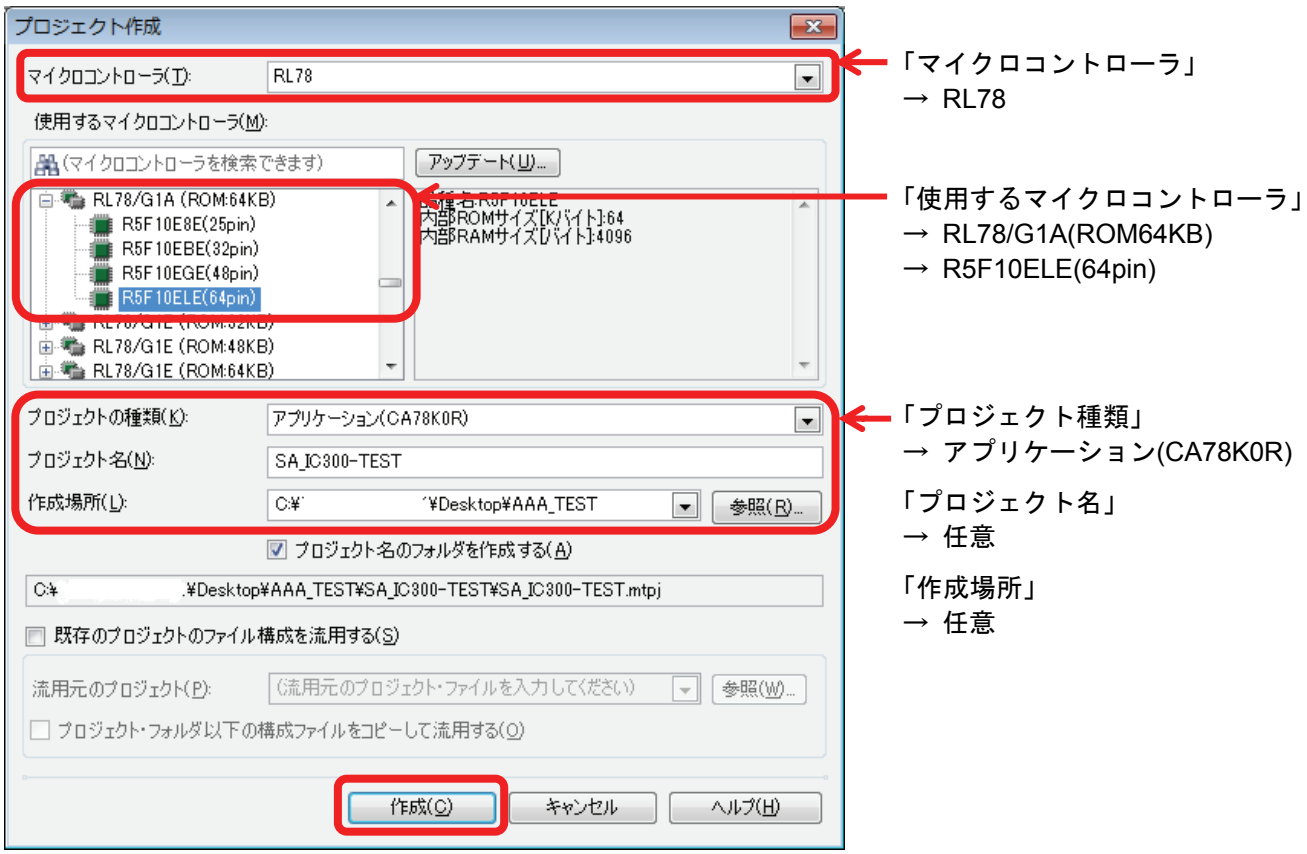


「基本機能」タブから、「RL78,78K0R ビルド・ツール CA78K0R 用プラグイン」、「RL78,78K0R エミュレータ用プラグイン」、「RL78,78K0R シミュレータ用プラグイン」にチェックしてください。

OK をクリックしてください。

スタート画面の「新しいプロジェクトを作成する」の GO ボタンをクリック後、以下のように設定してください。

RL78/G1A を選択してコード生成を行います。



「マイクロコントローラ」
→ RL78

「使用するマイクロコントローラ」
→ RL78/G1A(ROM64KB)
→ R5F10ELE(64pin)

「プロジェクト種類」
→ アプリケーション(CA78K0R)

「プロジェクト名」
→ 任意

「作成場所」
→ 任意

上記を設定したら、作成ボタンをクリックしてください。

7.7　コード生成の設定をする

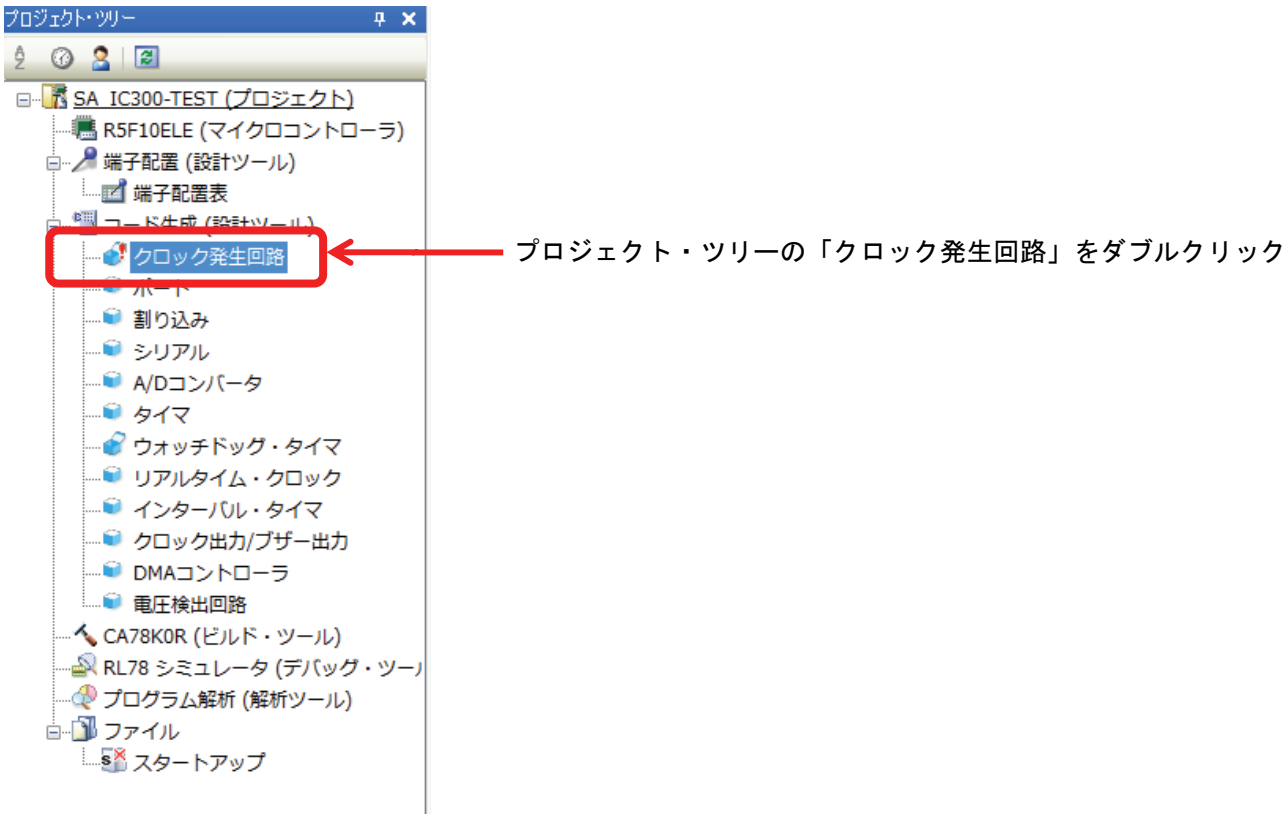
プロジェクトを作成したら、実際にマイコンを動作させるプログラムを作ります。

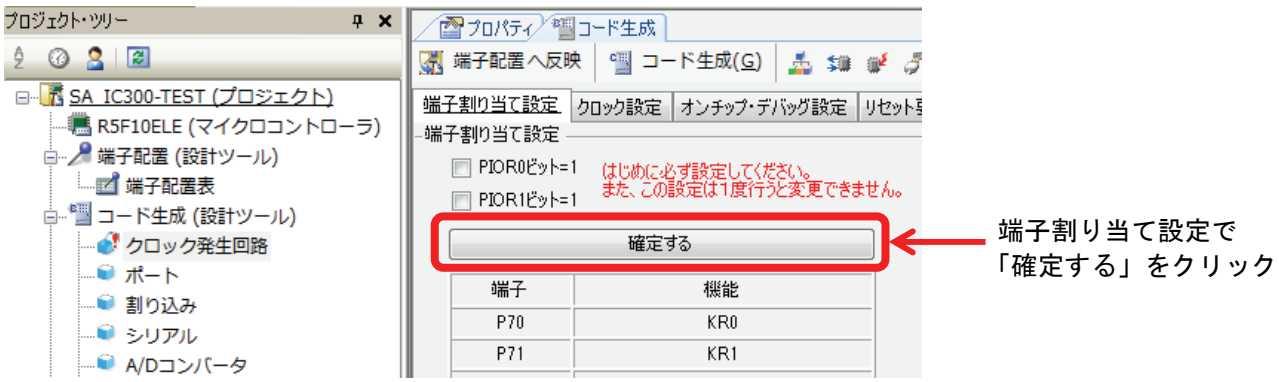
今回は、GUI 画面で指定するだけでマイコンの初期設定を簡単にコード生成する方法でプログラムを作ります。

プロジェクト・ツリーから設定したい機能をダブルクリックして設定していきます。

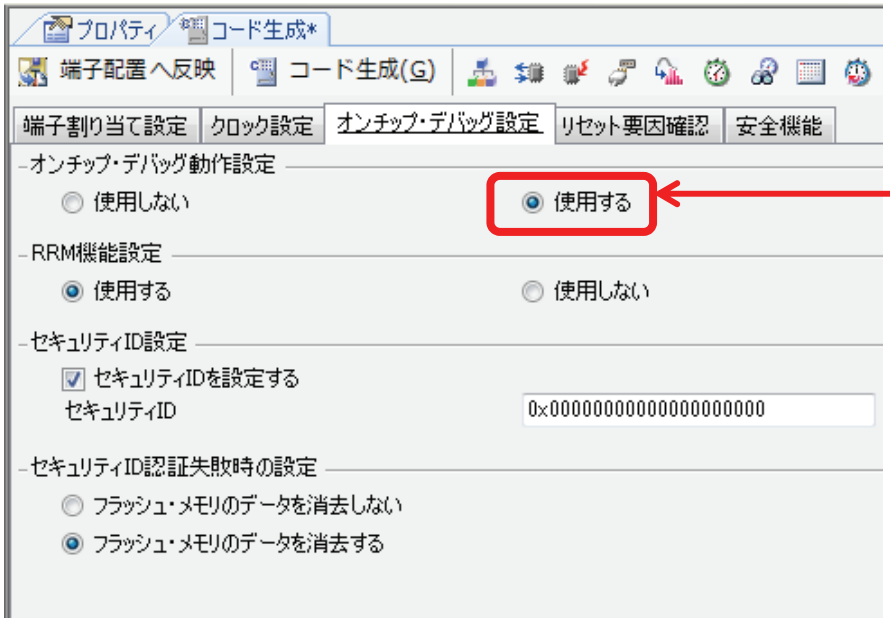
以下の手順にて各機能を設定していきます。

7.7.1　クロック発生回路の端子割り当て設定



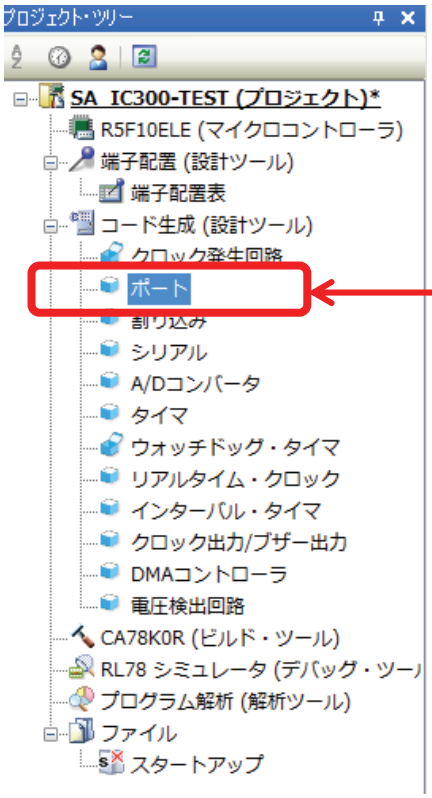


クロック発生回路のオンチップ・デバッグ設定



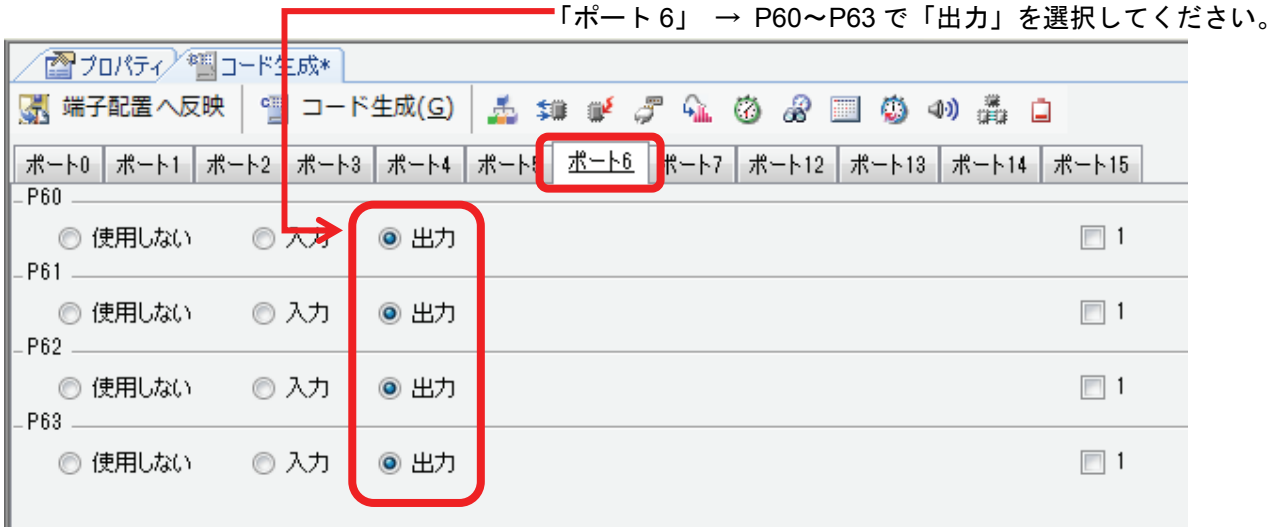
「オンチップ・デバッグ動作設定」
→使用する

7.7.2　ポートの端子割り当て設定



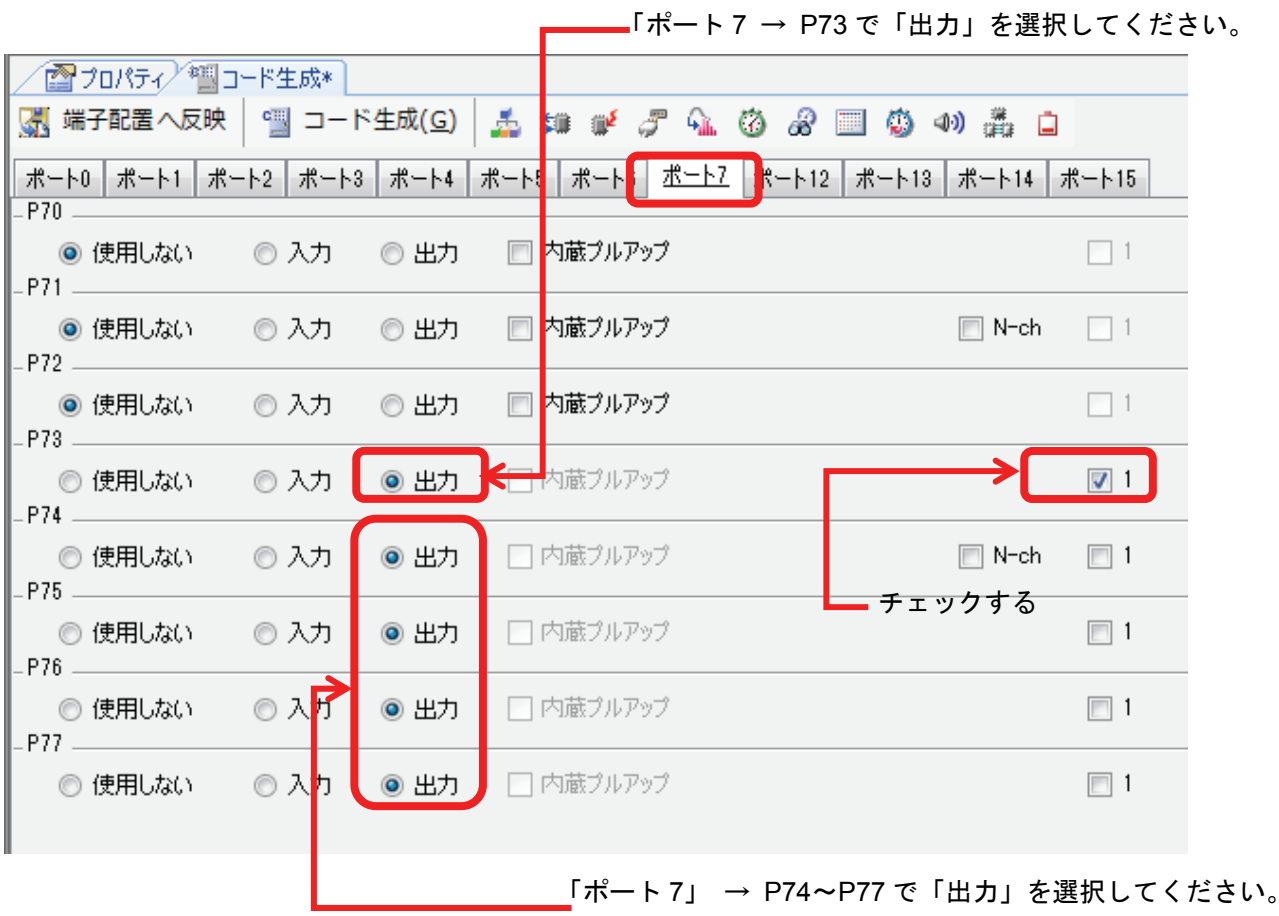
プロジェクト・ツリーの「ポート」をダブルクリック

「ポート 6」を設定します。ポート 6 はセンサの挙動を受けて点滅する LED を制御します。

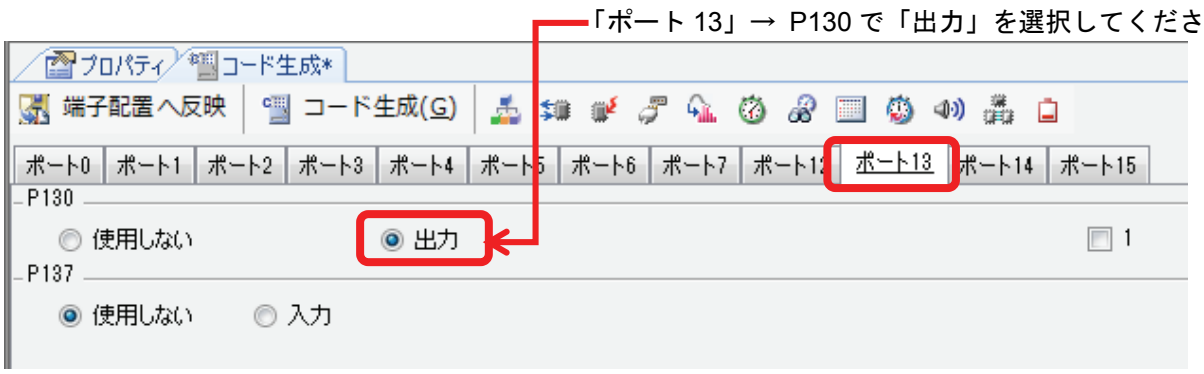


「ポート 7」を設定します。ポート 73 は Smart Analog IC の「CS」端子とポート 74～77 はセンサの挙動を受けて点滅する LED を制御します。

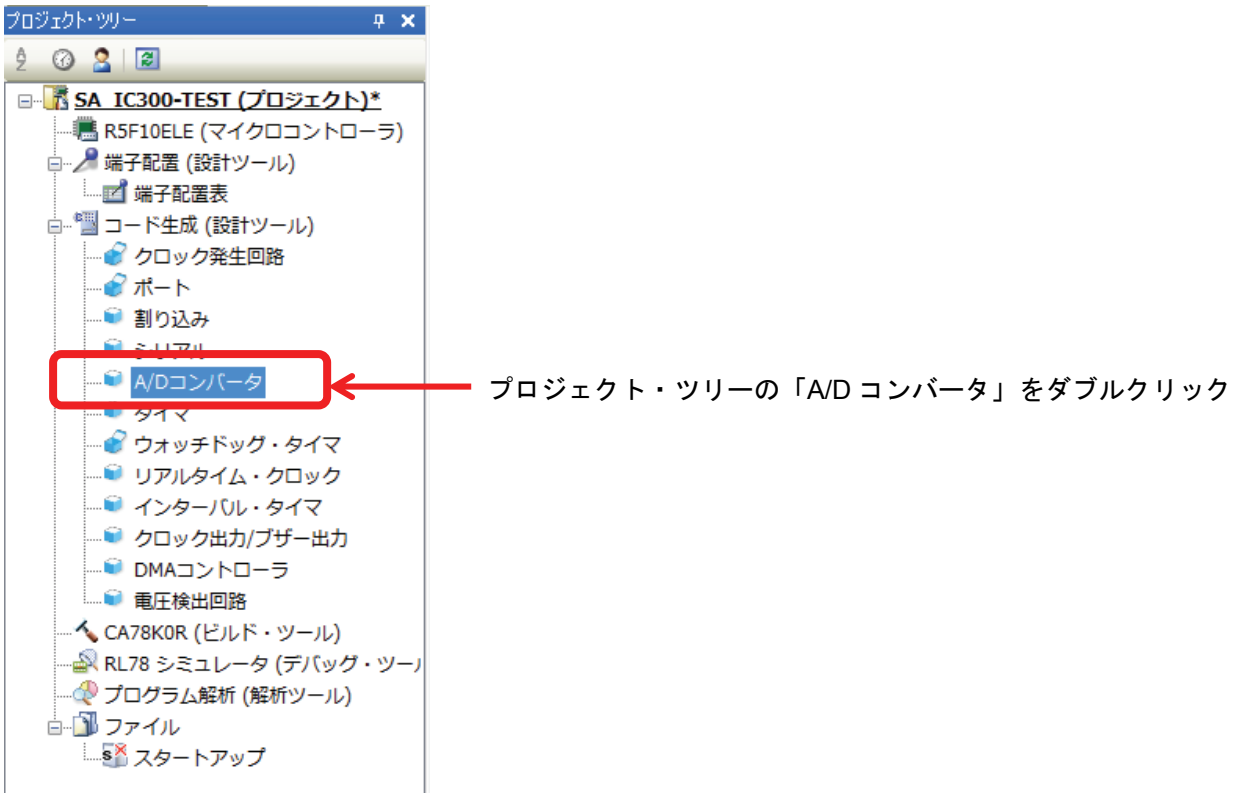
このポートは「Active High」で使用しますので、ポート設定の右端の「1」にチェックを入れてください。

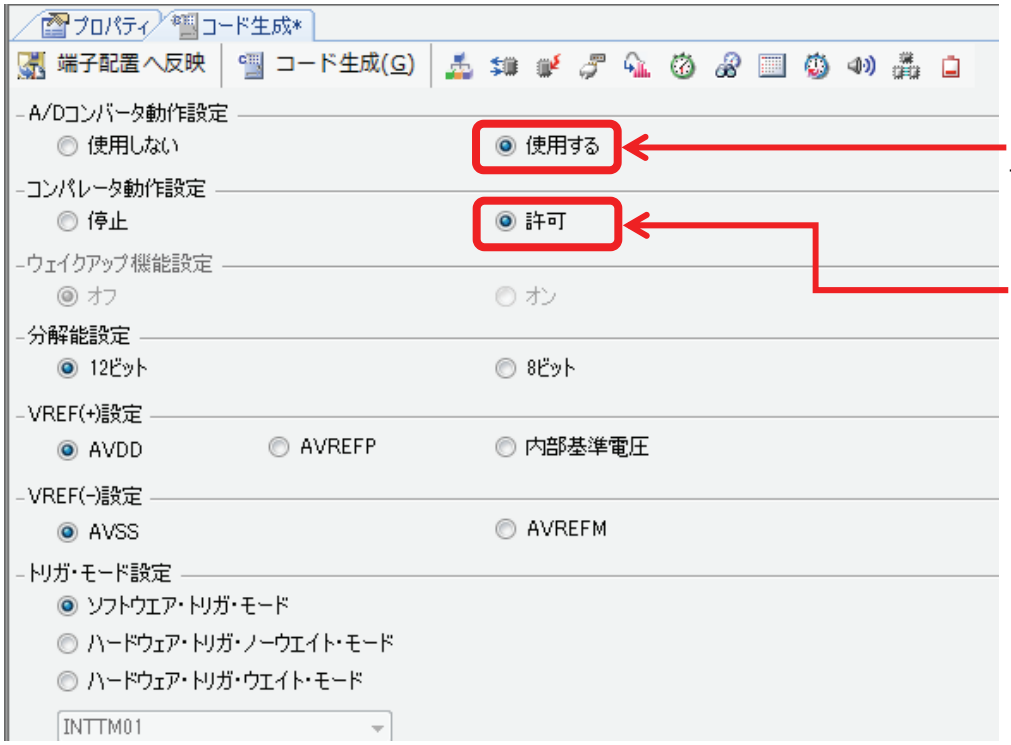


「ポート 13」を設定します。ポート 13 は Smart Analog IC の「RESET」端子を制御します。



7.7.3　A/D コンバータの設定





「A/D コンバータ動作設定」
→ 使用する

「コンパレータ動作設定」
→ 許可

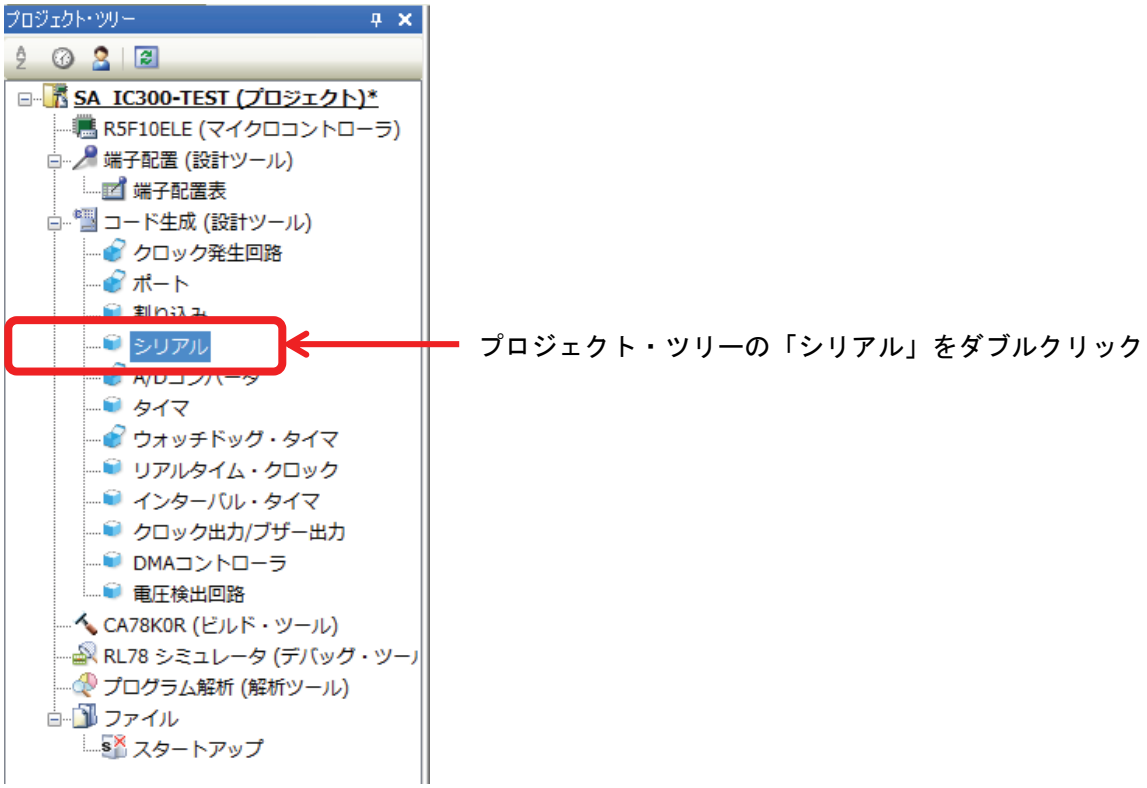
スクロールしてください

「動作モード設定」
→ 全てのチェックを OFF

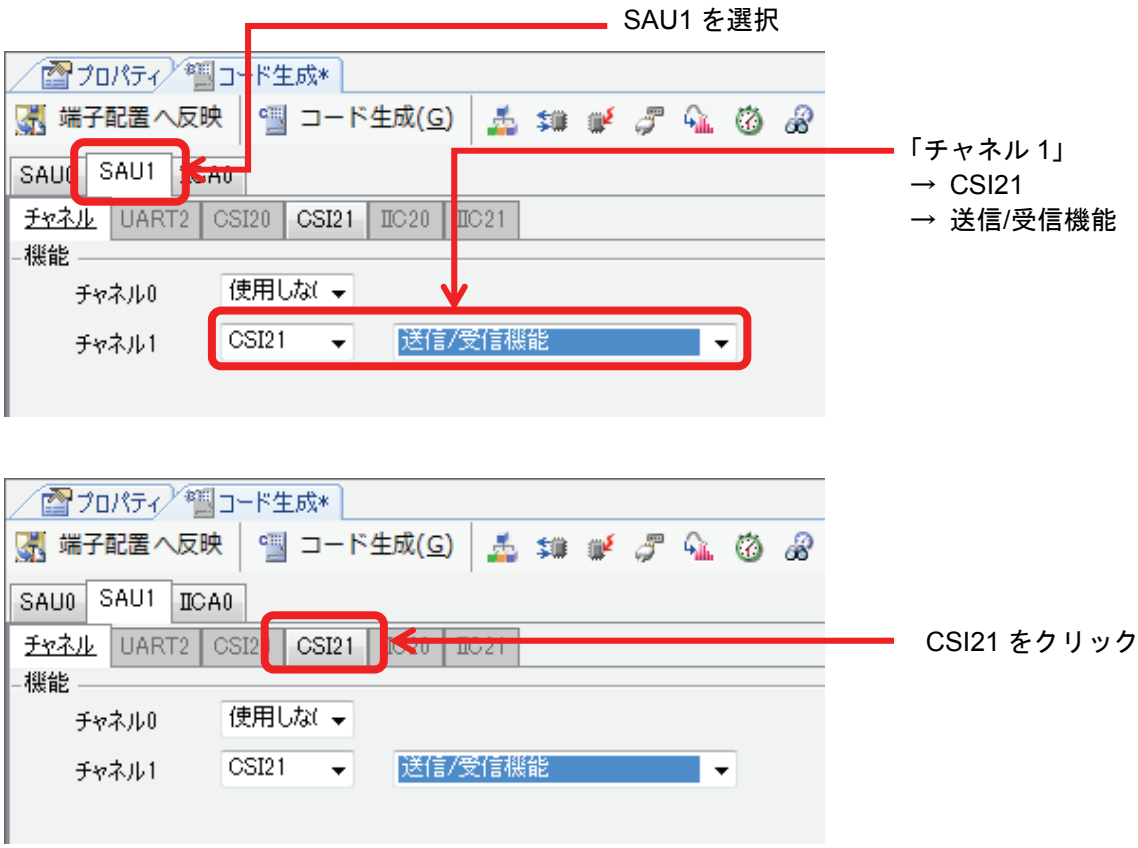
「変換開始チャネル設定」
→ ANI2 を選択

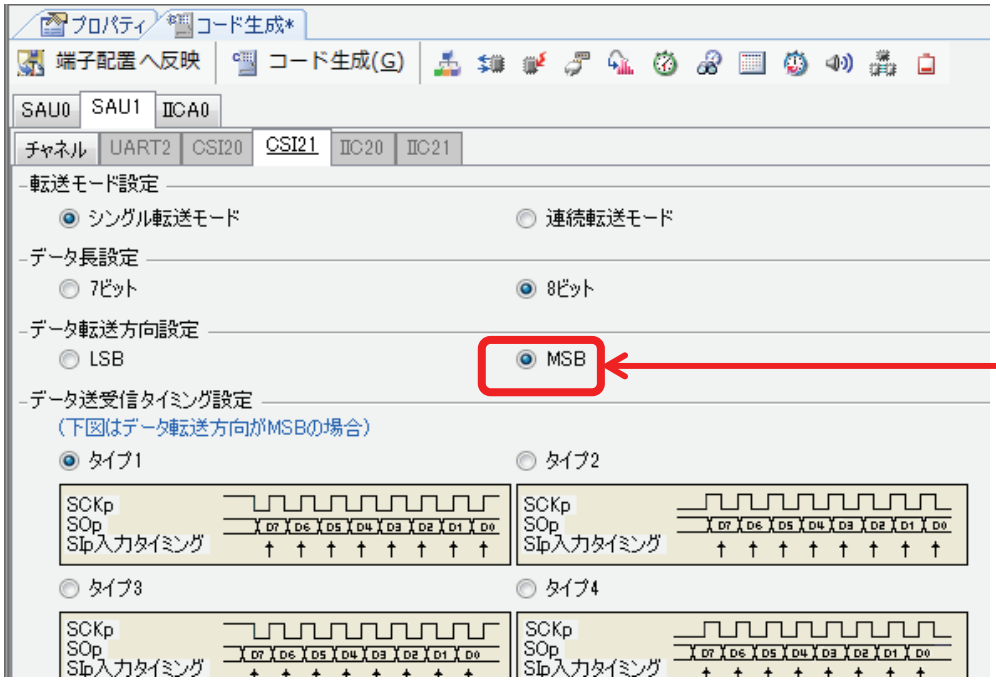
AD コンバータの端子設定については、今回のプログラムは ANI2 のみを使用しますので、ANI16-ANI30 は全て OFF にしてください。

7.7.4　シリアルの設定



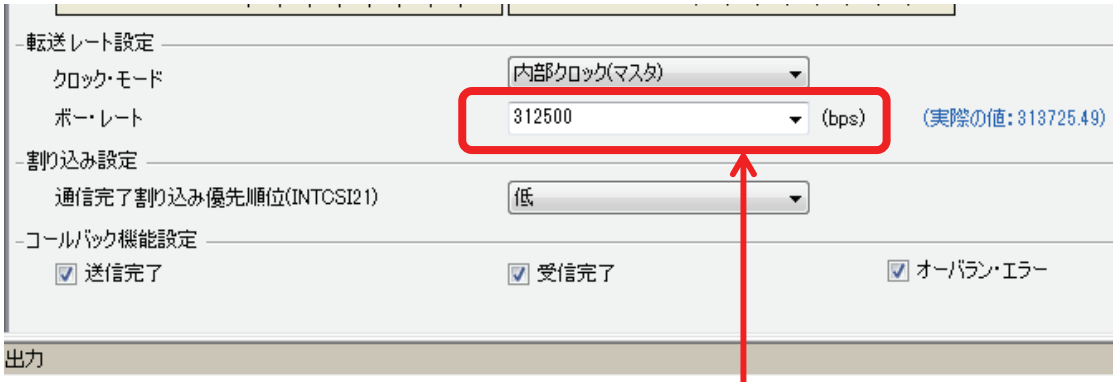
シリアルユニットは「CSI21」を使用しますので、CSI21 を選択して各種設定をします





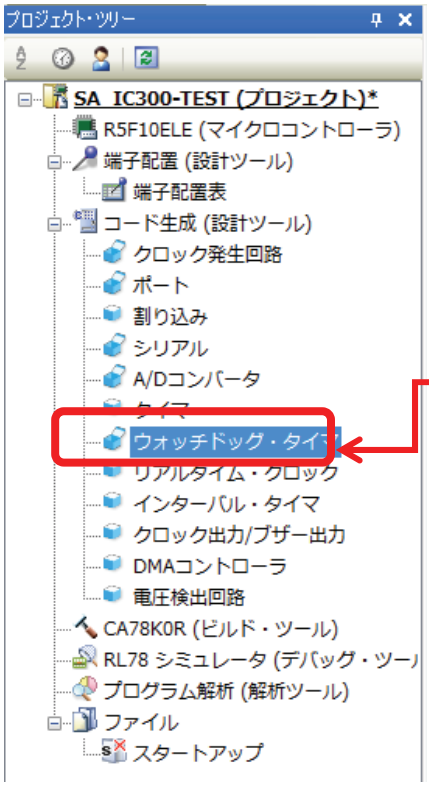
「データ転送方向設定」
→ MSB

スクロールしてください

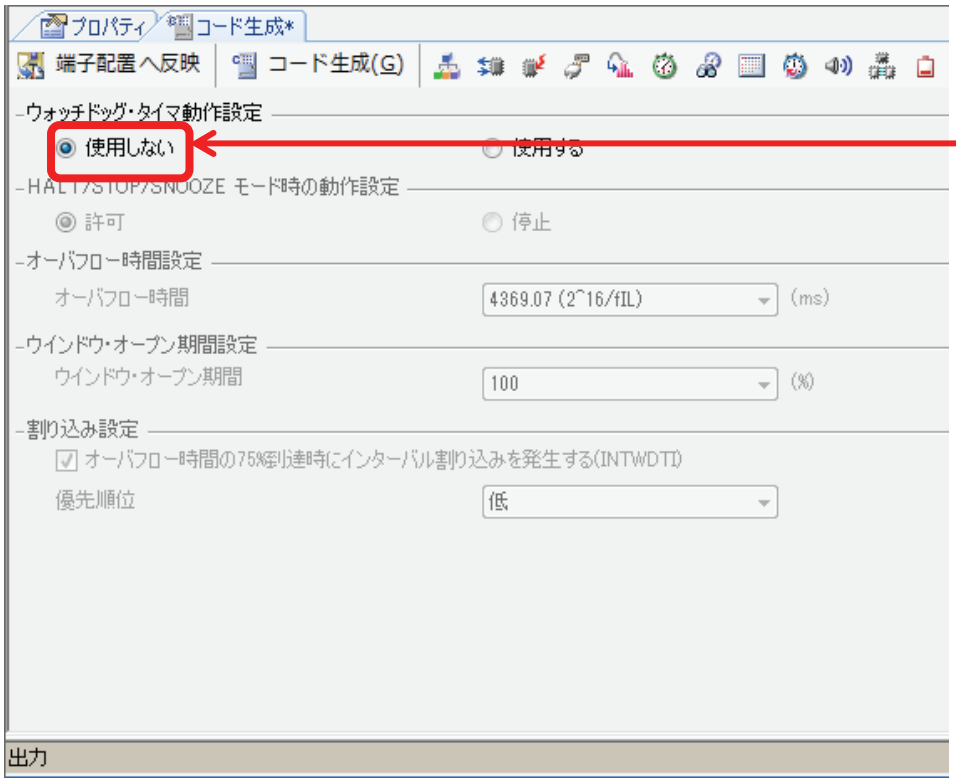


「転送レート設定」「ボー・レート」
→ 312500

7.7.5　ウォッチドックタイマの設定



プロジェクト・ツリーの「ウォッチドック・タイマ」をダブルクリック

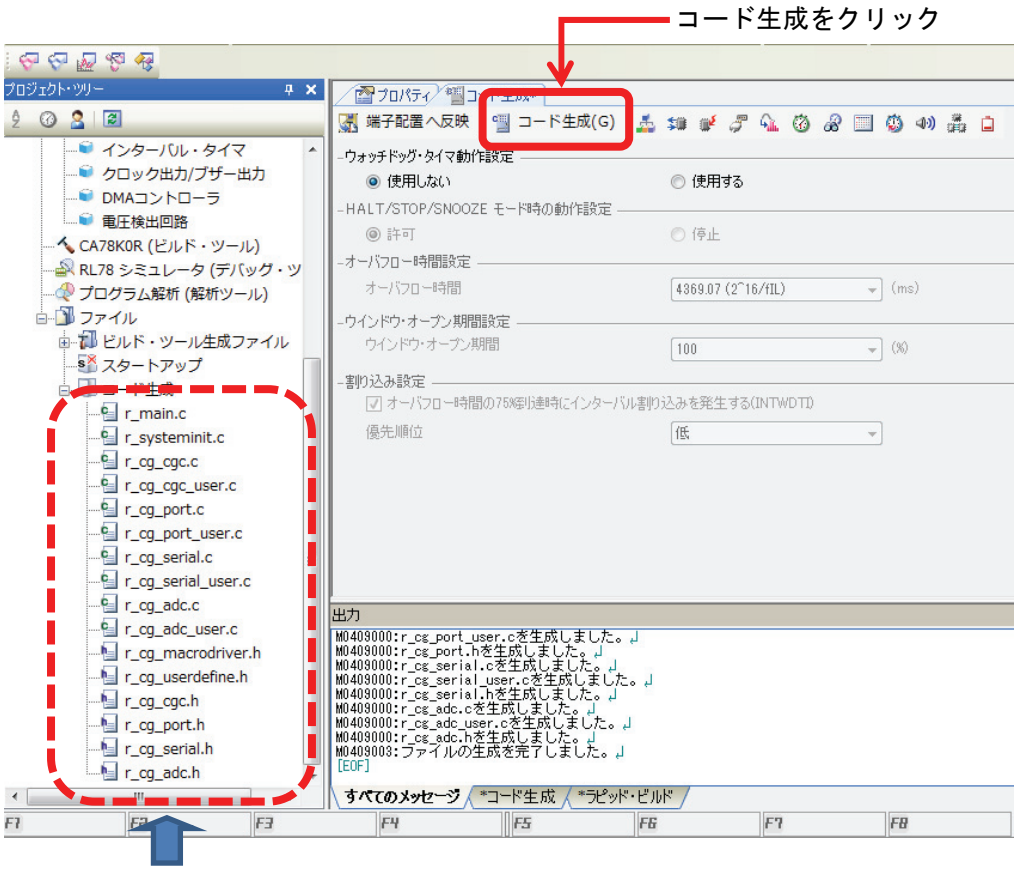


「ウォッチドッグ・タイマ動作設定」
→ 使用しない

7.8　生成実行

最後にコード生成ボタンをクリックしてコードを生成します。

プロジェクト・ツリーの「ファイル」の中に C ソースやヘッダーが出力されます。



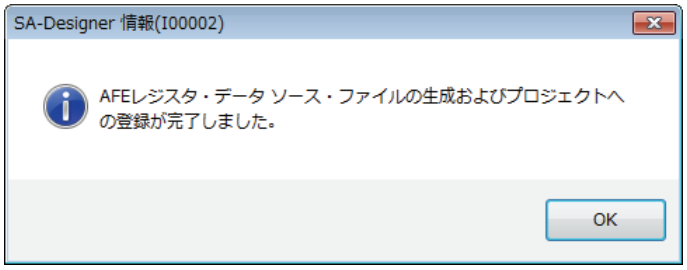
マイコン初期化用のコードが生成されます。
また各種機能を制御する API も生成されます。

7.8.1　回路パラメータの C ソース生成と登録

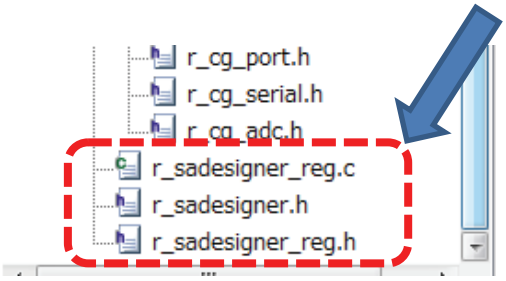
ここで一旦 SA-Designer に戻ります。

SA-Designer で Smart Analog IC の回路パラメータの C ソースを生成し、生成した C ソースを Cube Suite+に自動登録します。

下記の赤枠のアイコンをクリックしてください。



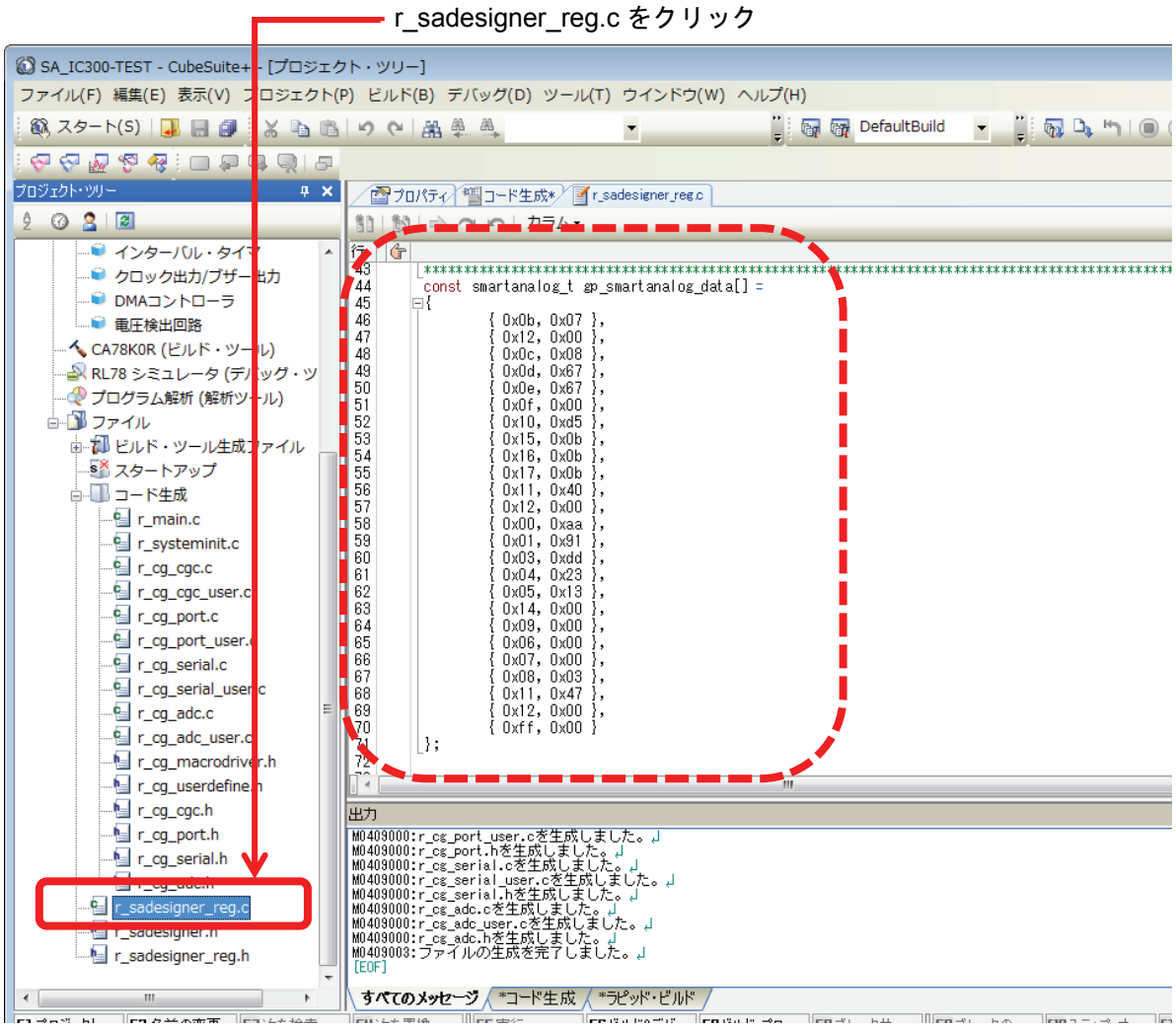
「OK」をクリックしてください。



Cube Suite+のプロジェクトツリーに生成した
回路パラメータの C ソースファイルが登録されます。

7.9　SA-Designer が生成したファイルの補足

Smart Analog IC の回路パラメータは CubeSuite+上では構造体変数(const)で定義されます。



回路パラメータは{アドレス, データ}で定義された構造体変数にて生成されます。

最後の{0xff, 0x00}はデリミタ用の値です。

7.10　プログラムを編集する

7.10.1　r_main.c の編集

プロジェクト・ツリーにある「r_main.c」を開いてください。

下記の青文字の部分を追加してください。

```
/***********************************************************************************
***************************************
Includes
***********************************************************************************
***************************************/
#include "r_cg_macrodriver.h"
#include "r_cg_cgc.h"
#include "r_cg_port.h"
#include "r_cg_serial.h"
#include "r_cg_adc.h"
/* Start user code for include. Do not edit comment generated here */
#include "r_sadesigner.h"
/* End user code. Do not edit comment generated here */
#include "r_cg_userdefine.h"

/***********************************************************************************
***************************************
Global variables and functions
***********************************************************************************
***************************************/
/* Start user code for global. Do not edit comment generated here */
void R_SAIC_Write(smartanalog_t * const p_saic_data);
void R_SAIC_Create(void);
extern const smartanalog_t gp_smartanalog_data[];
/* End user code. Do not edit comment generated here */
void R_MAIN_UserInit(void);

/***********************************************************************************
***************************************
* Function Name: main
* Description  : This function implements main function.
* Arguments    : None
* Return Value : None
***********************************************************************************
***************************************/
void main(void)
{
    R_MAIN_UserInit();
    /* Start user code. Do not edit comment generated here */
    R_CSI21_Start();
    R_SAIC_Create();
    R_ADC_Start();

    while (1U)
    {
        ;
    }
    /* End user code. Do not edit comment generated here */
}

/***********************************************************************************
***************************************
* Function Name: R_MAIN_UserInit
* Description  : This function adds user code before implementing main function.
* Arguments    : None
* Return Value : None
***********************************************************************************
***************************************/
```

SA-Designer 生成ソースをインクルード

SPI 関数の宣言、変数の広域宣言

SPI、Smart Analog IC 初期設定、A/D のスタート

```
void R_MAIN_UserInit(void)
{
    /* Start user code. Do not edit comment generated here */
    EI();
    /* End user code. Do not edit comment generated here */
}

/* Start user code for adding. Do not edit comment generated here */
void R_SAIC_Create(void)
{
    volatile uint16_t w_count;
    /* Analog IC Reset */
    P13.0 = 0U;
    /* Change the waiting time according to the system */
    for (w_count = 0U; w_count < 0x1A; w_count++)
    {
        NOP();
    }
    /* Analog IC Reset release */
    P13.0 = 1U;
    R_SAIC_Write(gp_smartanalog_data);
}

void R_SAIC_Write(smartanalog_t * const p_saic_data)
{
    uint8_t adrs;
    uint8_t dat;
    uint8_t wait;
    smartanalog_t *p_saic_write;
    PMC3 = 0xfc;
    CSIMK21 = 1U;    /* mask INTCSI21 */

    p_saic_write = p_saic_data;
    while (p_saic_write->address != 0xff)
    {
        P7.3 = 0U;                  /* SAIC CS=L */
        /* SA stable waiting time (tSKA) */
        for (wait = 0U; wait < 1U; wait++)
        {
            NOP();
        }
        adrs = (p_saic_write->address & 0x7f) | 0x80;   /* 0x80 is data write mode
*/
        SIO21 = adrs;                               /* send SAIC Address data  */
        while (CSIIF21 == 0U);                       /* wait for CSI send      */
        CSIIF21 = 0U;
        dat = p_saic_write->data;
        SIO21 = dat;                /* send SAIC Regiseter Data    */
        while (CSIIF21 == 0U);      /* wait for CSI send           */
        CSIIF21 = 0;

        /* SA Stable waiting time (tKSA) */
        for (wait = 0U; wait < 1U; wait++)
        {
            NOP();
        }
        P7.3 = 1U;                  /* SAIC CS=H */
        /* SA Stable waiting time (tSHA) */
```

Smart Analog IC リセット

Smart Analog IC リセット解除、
設定関数コール

デリミタ検出

チップセレクト

アドレス送信

データ送信

チップセレクト解除

```
        for (wait = 0U; wait < 1U; wait++)
        {
            NOP();
        }
        p_saic_write++;
    }
}
/* End user code. Do not edit comment generated here */
```

チップセレクト解除

ここまで記入したら一旦「r_main.c」を保存します。

「r_main.c」の main 関数は、プログラム実行時に呼び出されます。main 関数内では、以下の処理を行っています。

1. ユーザー定義初期設定 (今回は修正不要)
2. CSI21 初期設定 (RL78/G1A と Smart Analog IC300 の通信設定)
3. Smart Analog IC300 のリセット
4. Smart Analog IC300 の初期設定 (回路パラメータ設定)
5. A/D コンバータ動作開始
6. 無限ループ

7.10.2　r_cg_adc_user.c の編集

プロジェクト・ツリーにある「r_cg_adc_user.c」を開いてください。

下記の青文字の部分を追加してください。

```
/***********************************************************************************
***************************************
Global variables and functions
***********************************************************************************
***************************************/
/* Start user code for global. Do not edit comment generated here */
uint16_t    g_tmp = 0;                                          A/D 変換結果取得用変数の定義
/* End user code. Do not edit comment generated here */

/***********************************************************************************
***************************************
* Function Name: r_adc_interrupt
* Description  : This function is INTAD interrupt service routine.
* Arguments    : None
* Return Value : None
***********************************************************************************
***************************************/
__interrupt static void r_adc_interrupt(void)
{
    /* Start user code. Do not edit comment generated here */
  #define STEP 14                                              感度を指定
                                                               (10hPa=約 10mV に対応する A/D 変換結果値)

  static int16_t initial_count = 0;                            起動後の時間管理用カウンタ
  static int16_t count = 0;                                    時間管理用カウンタ
  static uint16_t initial_atmosphere_adc;                      リセット解除してから 3 分後の気圧に対する A/D 変換結果
  int16_t differential_adc;                                    initial_atmosphere_adc と現在の A/D 変換結果の差
  int16_t atmosphere_level;                                    現在の気圧が 9 段階レベルゲージのどのレベルか計算した結果

  uint8_t LED_data_U = P7;                                     Port6, Port7 の値取得 (LED 点灯用)
  uint8_t LED_data_L = P6;

  if (initial_count < 181)                                     約 1 秒 * 180 回 = 約 3 分待ち
  {
   if (count > 18519)                                          54us * 18519 回 = 約 1 秒待ち
   {
      initial_count++;
      count = 0;
   }
  }
  else if (initial_count == 181)                               約 3 分後
  {
   R_ADC_Get_Result(&initial_atmosphere_adc);                  リセット解除してから 3 分後の
   initial_count++;                                            A/D 変換結果を大気圧の初期値とする
   count = 0;
  }
  else                                                         大気圧の初期値 initial_atmosphere_adc 取得して以降の処理
  {
   if (count > 1851)                                           約 100ms 間隔で LED 更新 (54us * 1851 回 = 約 100ms)
   {
      R_ADC_Get_Result(&g_tmp);                                A/D 変換結果を取得
      differential_adc = g_tmp - initial_atmosphere_adc;       初期 A/D 値と今回の A/D 値を比較
      atmosphere_level = differential_adc / STEP;              気圧のレベルを算出

      if (atmosphere_level > 2)                                気圧のレベルに応じて LED 値設定
      {
         LED_data_U = 0x00 | (LED_data_U & 0x0F);
         LED_data_L = 0x00 | (LED_data_L & 0xF0);
      }
```

```
        else if (atmosphere_level == 2)
        {
           LED_data_U = 0x80 | (LED_data_U & 0x0F);
           LED_data_L = 0x00 | (LED_data_L & 0xF0);
        }
        else if (atmosphere_level == 1)
        {
           LED_data_U = 0xC0 | (LED_data_U & 0x0F);
           LED_data_L = 0x00 | (LED_data_L & 0xF0);
        }
        else if (atmosphere_level == 0)
        {
           LED_data_U = 0xE0 | (LED_data_U & 0x0F);
           LED_data_L = 0x00 | (LED_data_L & 0xF0);
        }
        else if (atmosphere_level == -1)
        {
           LED_data_U = 0xF0 | (LED_data_U & 0x0F);
           LED_data_L = 0x00 | (LED_data_L & 0xF0);
        }
        else if (atmosphere_level == -2)
        {
           LED_data_U = 0xF0 | (LED_data_U & 0x0F);
           LED_data_L = 0x08 | (LED_data_L & 0xF0);
        }
        else if (atmosphere_level == -3)
        {
           LED_data_U = 0xF0 | (LED_data_U & 0x0F);
           LED_data_L = 0x0C | (LED_data_L & 0xF0);
        }
        else if (atmosphere_level == -4)
        {
           LED_data_U = 0xF0 | (LED_data_U & 0x0F);
           LED_data_L = 0x0E | (LED_data_L & 0xF0);
        }
        else
        {
           LED_data_U = 0xF0 | (LED_data_U & 0x0F);
           LED_data_L = 0x0F | (LED_data_L & 0xF0);
        }

        P7 = LED_data_U;
        P6 = LED_data_L;

        count = 0;
      }
    }
    count++;

    /* End user code. Do not edit comment generated here */
}

/* Start user code for adding. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
```

気圧のレベルに応じて LED 値を設定

3 以上：全て点灯
2：P60,P61,P62,P63,P74,P75,P76 点灯
1：P60,P61,P62,P63,P74,P75 点灯
0：P60,P61,P62,P63,P74 点灯
-1：P60,P61,P62,P63 点灯
-2：P60,P61,P62 点灯
-3：P60,P61 点灯
-4：P60 点灯
-5 以下：全て消灯

LED 更新

ここまで記入したら「r_cg_adc_user.c」を保存します。

「r_cg_adc_user.c」の r_adc_interrupt 関数は、A/D 変換処理終了の割り込み時に呼び出されます。r_adc_interrupt 関数内では、以下の処理を行っています。

1. Smart Analog IC300 の初期設定後、約 3 分待ち
   (センサから Smart Analog IC を経ての出力が安定するまで待つ)
2. 約 3 分後に取得した A/D 変換結果を、initial_atmosphere_adc に保存
3. 以降、100ms ごとに以下を繰り返す
   3.1 A/D 変換結果を取得
   3.2 取得した A/D 変換結果と initial_atmosphere_adc の差分を算出し、感度(*1)で割ることで気圧のレベルを算出
   3.3 算出した気圧のレベルに応じて LED 値を設定(*2)

*1：感度は 10hPa としています。ここでは計算するにあたり単位を揃えるため、10hPa に対応する A/D 変換値を感度(STEP)に指定しています。今回は実測の結果より 10hPa に対応する Smart Analog IC300 の出力電圧は約 10mV であったため STEP の値を 14 としています。A/D 変換結果(ADCR レジスタの値)は以下の式より算出しています。

$ADCR = INT ((V_{AIN} / AV_{REF}) * 4096 + 0.5)$　　(V_{AIN} = 10mV, AV_{REF} =3.0V として算出)

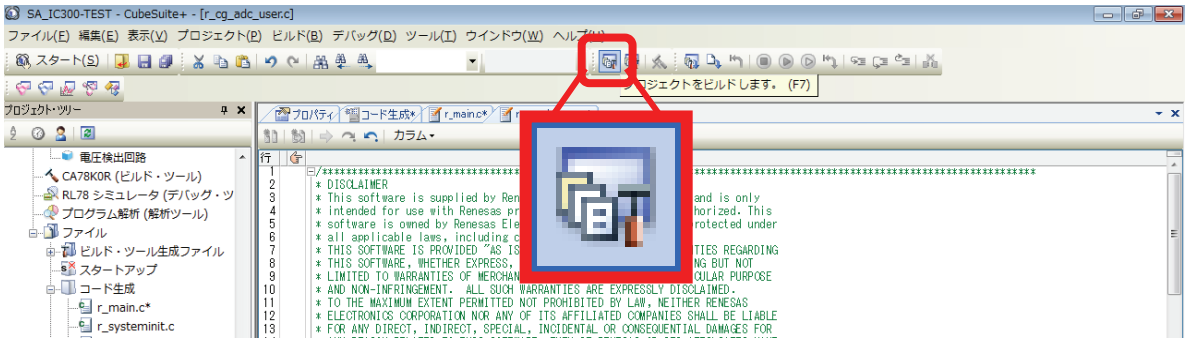
*2 : LED 値は下記に基づいて設定しています。（P：測定開始 3 分後の気圧、A：測定開始 3 分後の電圧）

気圧レベル	気圧(hPa)	A/D 入力電圧(mV)		点灯する LED と対応する Port （○：点灯、－：消灯）							
		MIN	MAX	P77	P76	P75	P74	P63	P62	P61	P60
				赤	赤	黄	黄	黄	緑	緑	緑
3 以上	P+30	A + 30.0	－	○	○	○	○	○	○	○	○
2	P+20	A + 20.0	A + 29.9	－	○	○	○	○	○	○	○
1	P+10	A + 10.0	A + 19.9	－	－	○	○	○	○	○	○
0	P	A	A + 9.9	－	－	－	○	○	○	○	○
-1	P-10	A - 10.0	A - 0.1	－	－	－	－	○	○	○	○
-2	P-20	A - 20.0	A - 10.1	－	－	－	－	－	○	○	○
-3	P-30	A - 30.0	A - 20.1	－	－	－	－	－	－	○	○
-4	P-40	A - 40.0	A - 30.1	－	－	－	－	－	－	－	○
-5 以下	P-50	－	A - 40.1	－	－	－	－	－	－	－	－

7.11　プログラムをビルドする

上記のプログラムの編集が終わったら、ビルドボタンをクリックしてプログラム(ロードモジュール)を生成します。

下記の赤枠で囲ったボタンをクリックしてください。

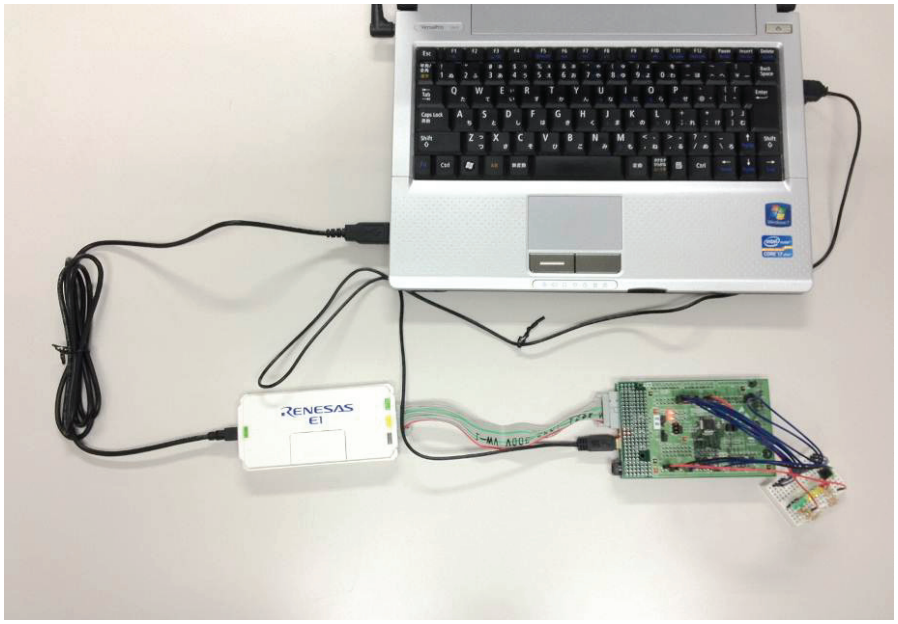


ここでエラーが出力されなかったら、プログラムは無事完成しています。

もし、エラー等が表示される場合は、エラーメッセージを確認し、7.7や7.10で設定した内容を確認してみてください。

7.12　プログラム実行環境を作る(エミュレータと接続する)

評価ボード、E1 エミュレータ、パソコンを接続してください。



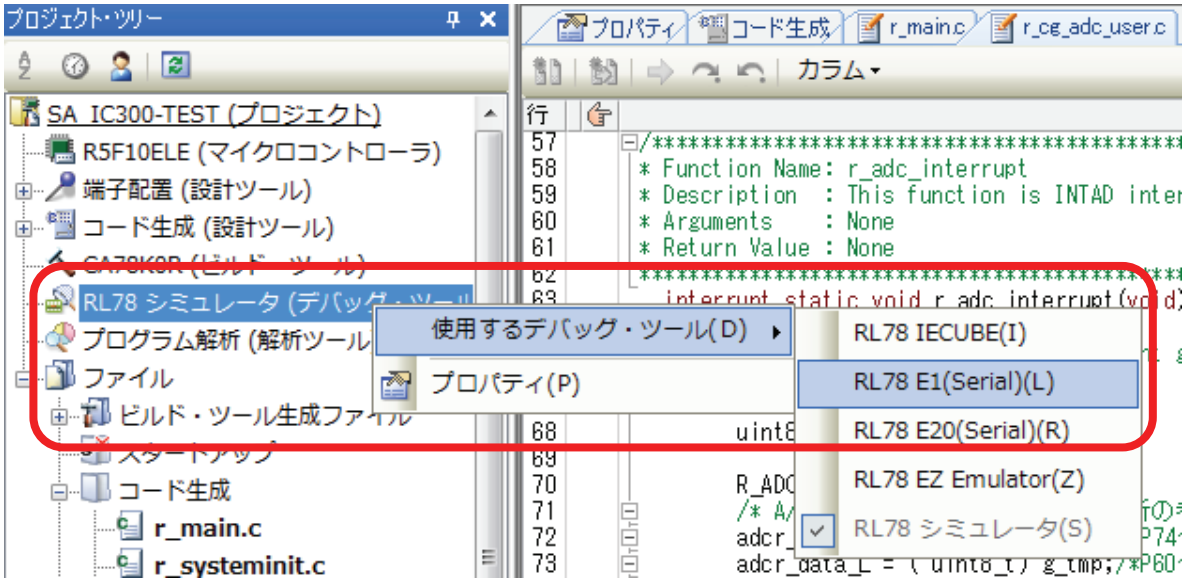
USB ケーブルを 2 つ用意してください。

- 1 つは E1 へ接続
- 1 つは評価ボード(TSA-IC 300)へ接続

E1 と評価ボード(TSA-IC 300)を E1 付属のケーブルで接続してください。

7.12.1　デバッグ・ツールを設定する

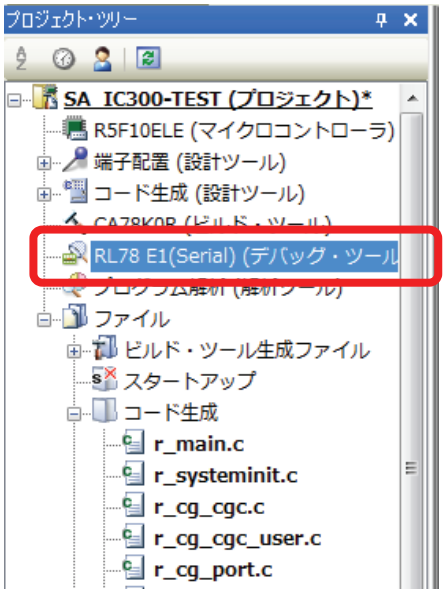
デバッグ・ツールの設定をします。



プロジェクト・ツリーの「RL78 シミュレータ(デバッグ・ツール)」を右クリック

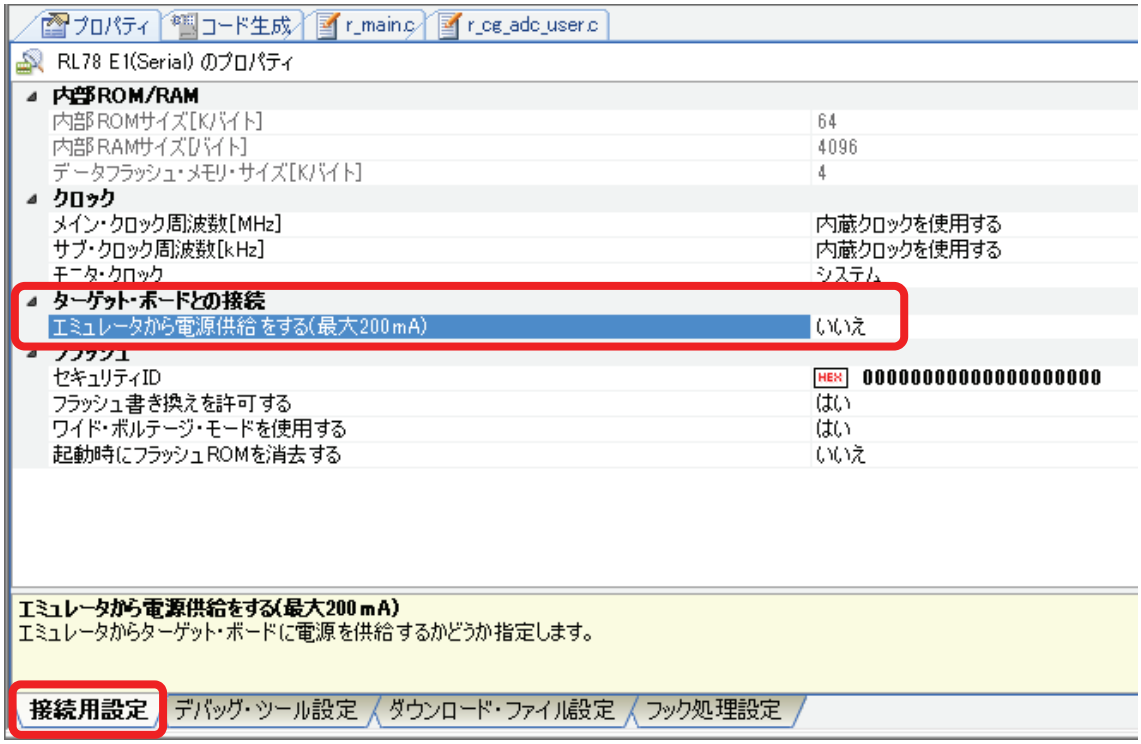
→ 「使用するデバッグ・ツール」 → 「RL78 E1(Serial)(L)」を選択

続いて、E1 エミュレータを接続する際の電源設定、プログラム実行中にメモリ空間を読み込めるようにする設定をします。

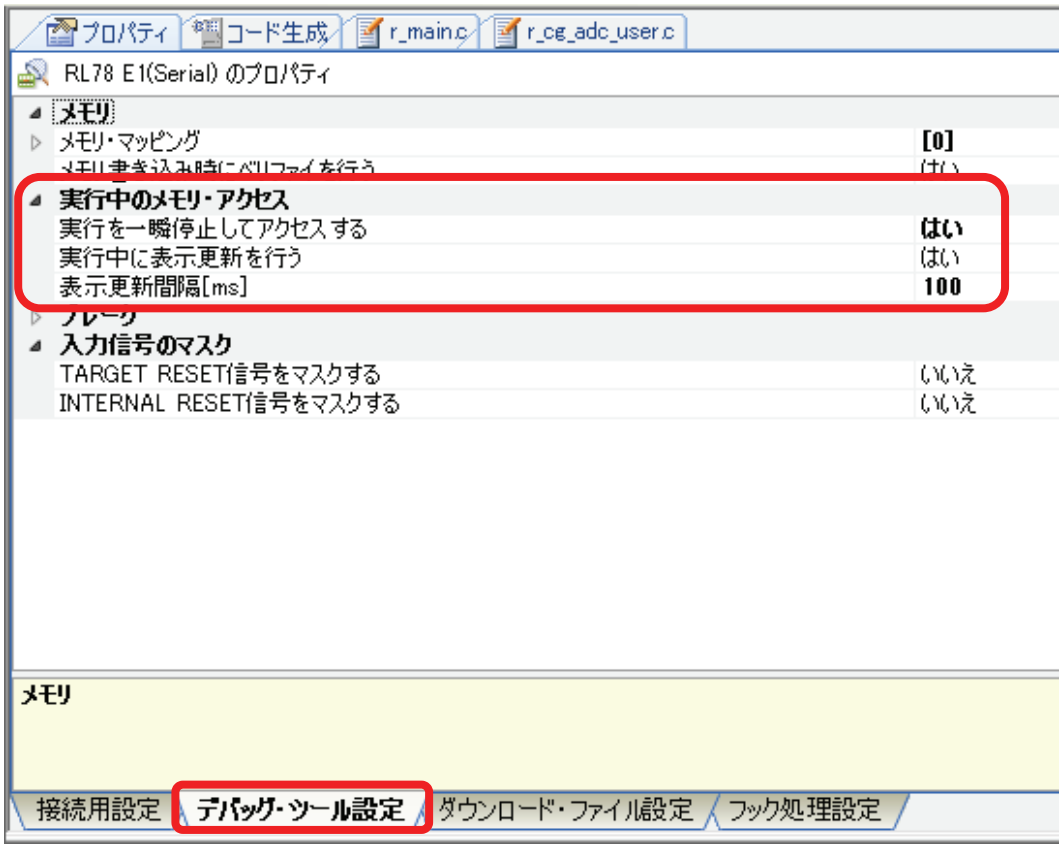


プロジェクト・ツリーの「RL78 E1(Serial)(デバッグ・ツール)」をダブルクリック

続いて以下を設定してください。



「接続用設定」タブの「ターゲット・ボードとの接続」 → エミュレータから電源供給をする(最大 200mA) → いいえ



「デバッグ・ツール設定」タブの「実行中のメモリ・アクセス」

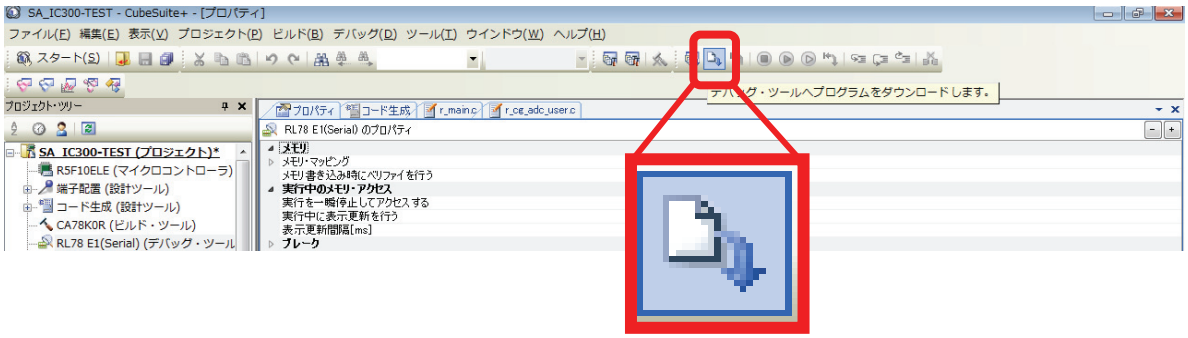
→ 実行を一瞬停止してアクセスする → はい

→ 表示更新間隔[ms] → 100

7.13　マイコンにプログラムをダウンロードする

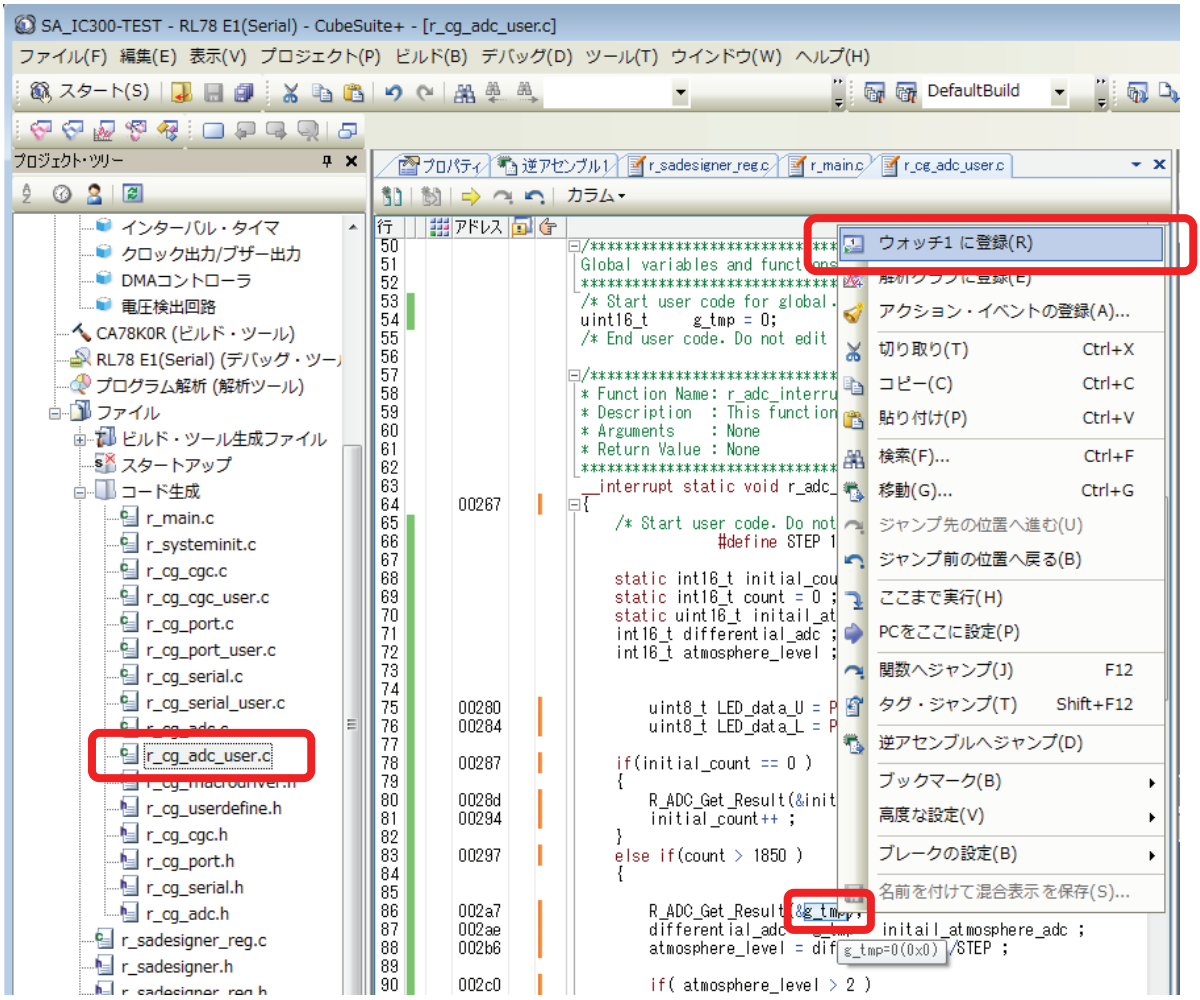
デバッグ・ツールの設定が終了したら、作ったプログラム(ロードモジュール)をマイコンにダウンロードします。

下記の赤枠で囲ったアイコンをクリックしてください。



7.14　A/D 変換で取得した値をモニターする

プログラムの中で A/D 変換にて取得した値は「g_tmp」に格納されます。そこで g_tmp を「ウォッチ登録」して動作をモニターできるようにします。



プロジェクト・ツリーの「r_cg_adc_user.c」をダブルクリック → 変数「g_tmp」の箇所をダブルクリックしてさらに右クリック → 「ウォッチ 1 に登録」

7.15　プログラムを実行してセンサーの動作を確認

では実際にマイコンにダウンロードしたプログラムを実行してセンサーの動作を確認します。

下記の赤枠で囲ったアイコンをクリックしてください。

先ほど「ウォッチ登録」した「g_tmp」の値が表示されます。

センサの動きに合わせて値が変化していることを確認してください。

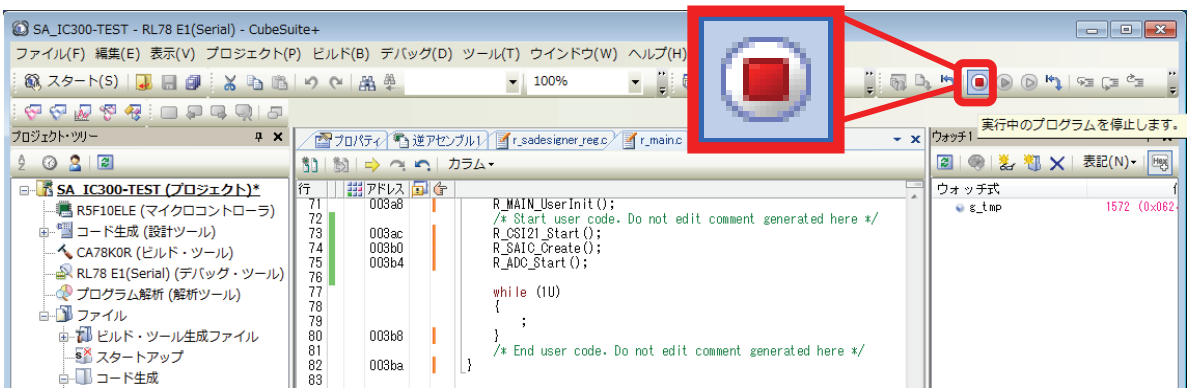
(センサの圧力をシリンジ等で変化させてみて下さい。)

なお、この時点で既に LED も点灯するので併せて確認してください。

ここで、センサの動きをより分かり易くするためにグラフ表示をさせてみます。

一旦プログラムを停止します。

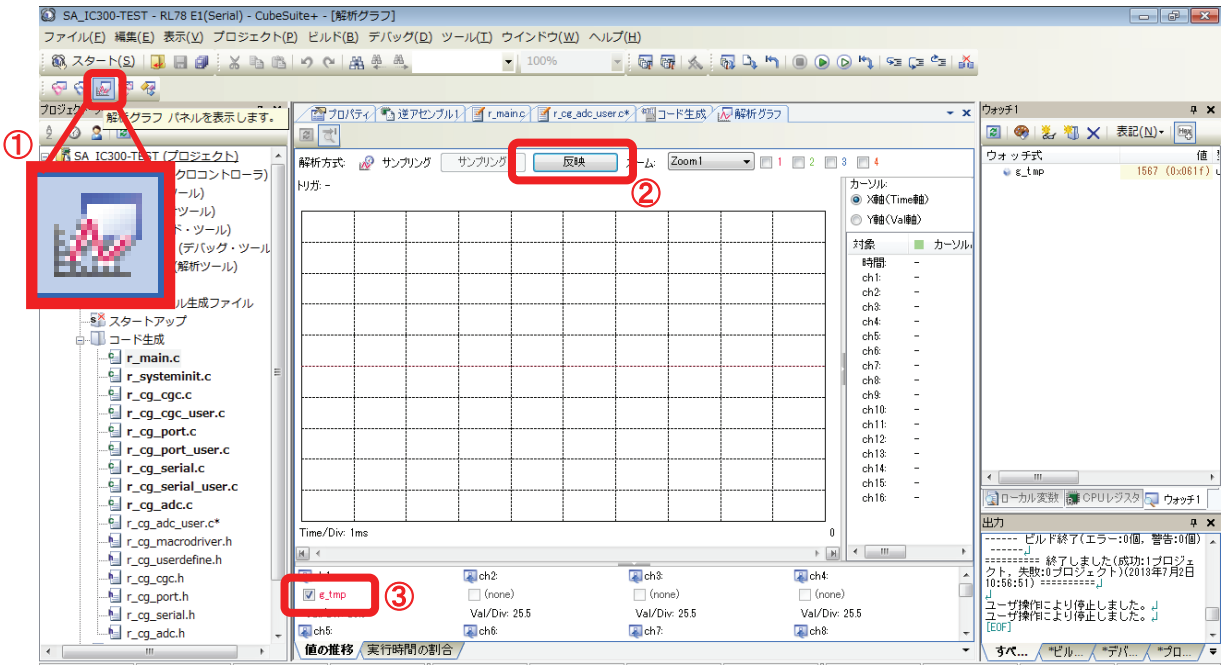
下記の赤枠で囲ったアイコンをクリックしてください。



7.16　解析グラフパネルでセンサの動きをグラフ表示する

センサの動きをグラフ表示させる設定をします。

下記の①～③の設定をしてください。

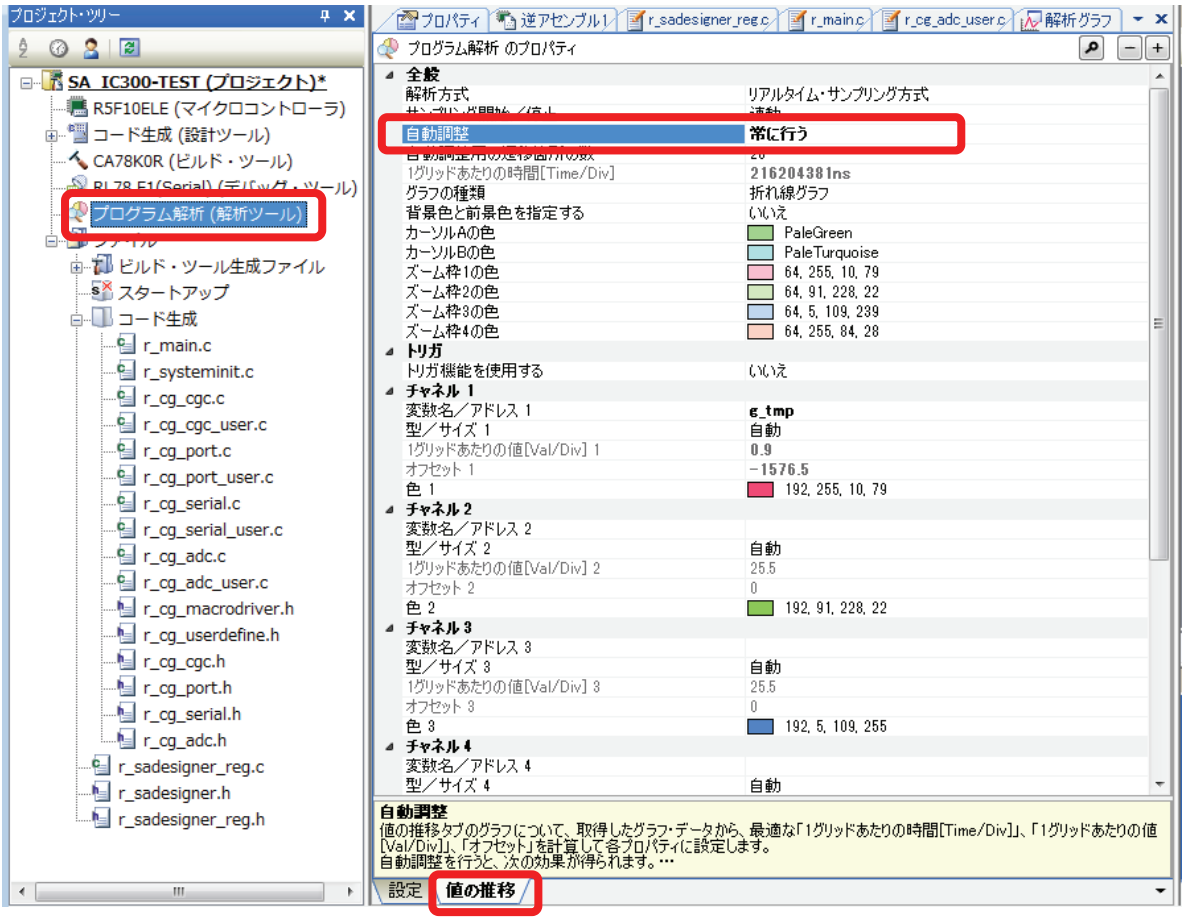


① 面左上のアイコンをクリックします。

② ラフウィンドウが表示されますので、その上の「反映」ボタンをクリックします。

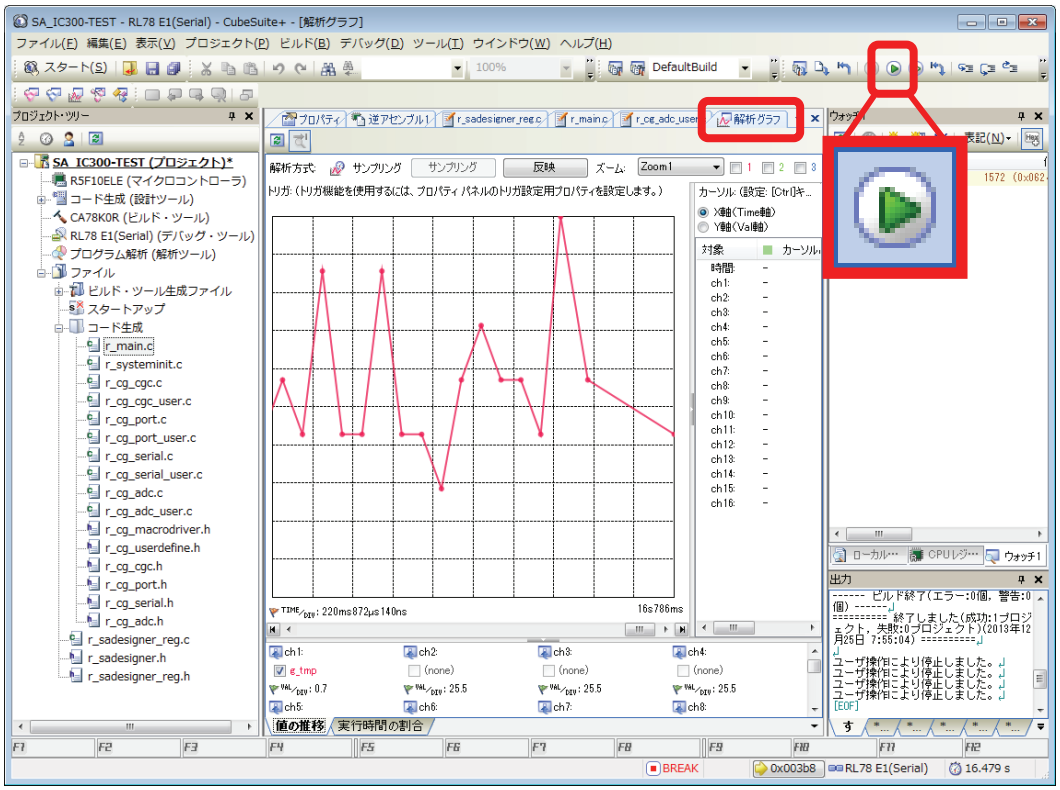
③ ラフ下画面に「g_tmp」が追加されたことを確認します。

次にプロジェクト・ツリーの「プログラム解析 (解析ツール)」をダブルクリックします。



「値の推移」タブの「全般」 → 「自動調整」 → 常に行う

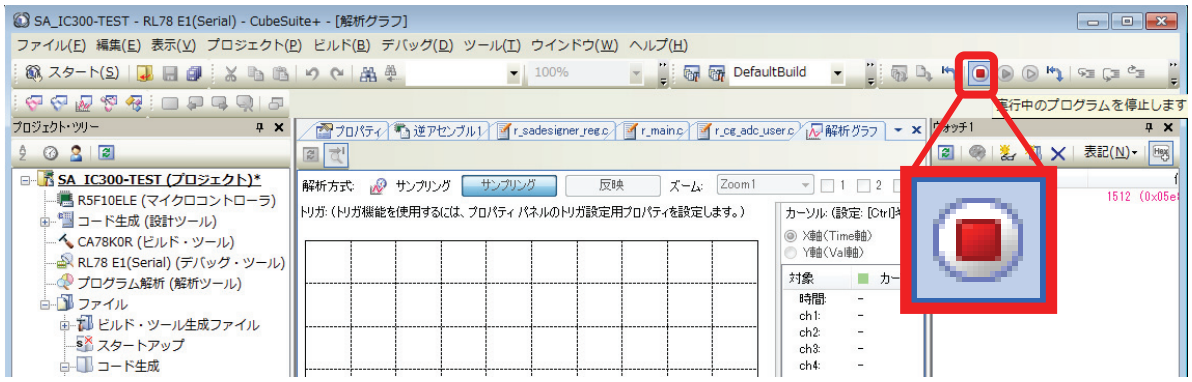
ここまでの設定が終わったら、下記赤枠のアイコンをクリックしてプログラムを再度動作させます。



「解析グラフ」のタブでセンサの動きが波形で表示されることを確認してください。
(先ほどと同様にセンサの圧力をシリンジ等で変化させてみて下さい。)

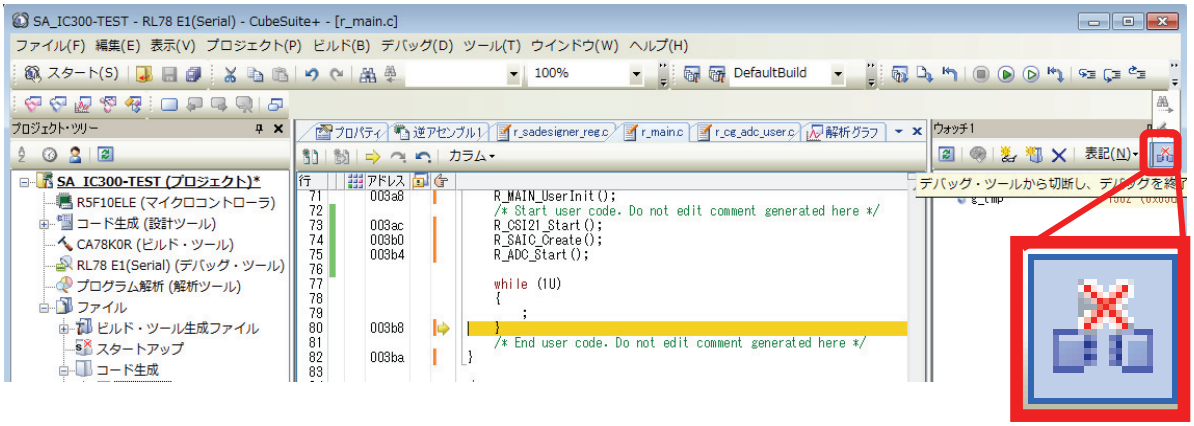
以上でプログラムは完成です。

プログラムを動作を停止させます。
下記の赤枠で囲ったアイコンをクリックしてください。

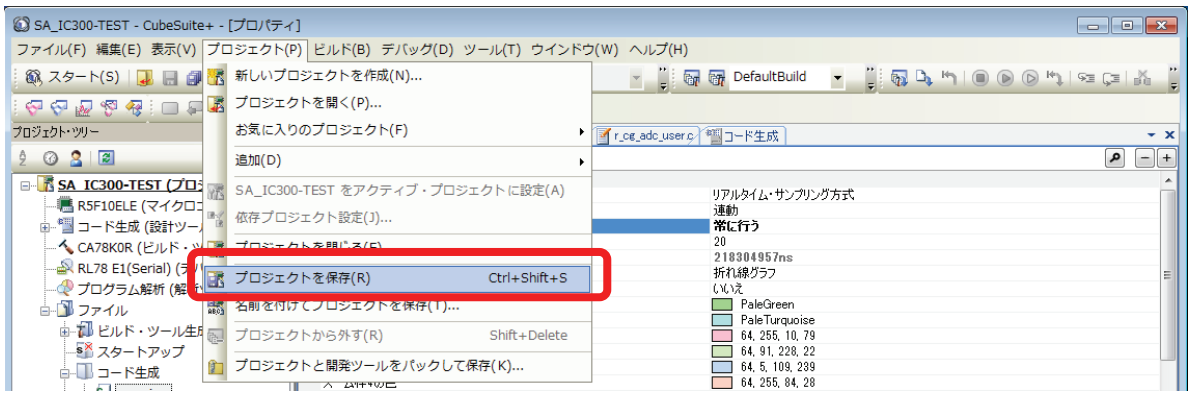


デバッグ・ツールを停止させて、デバッグを終了します

下記の赤枠のアイコンをクリックしてください。

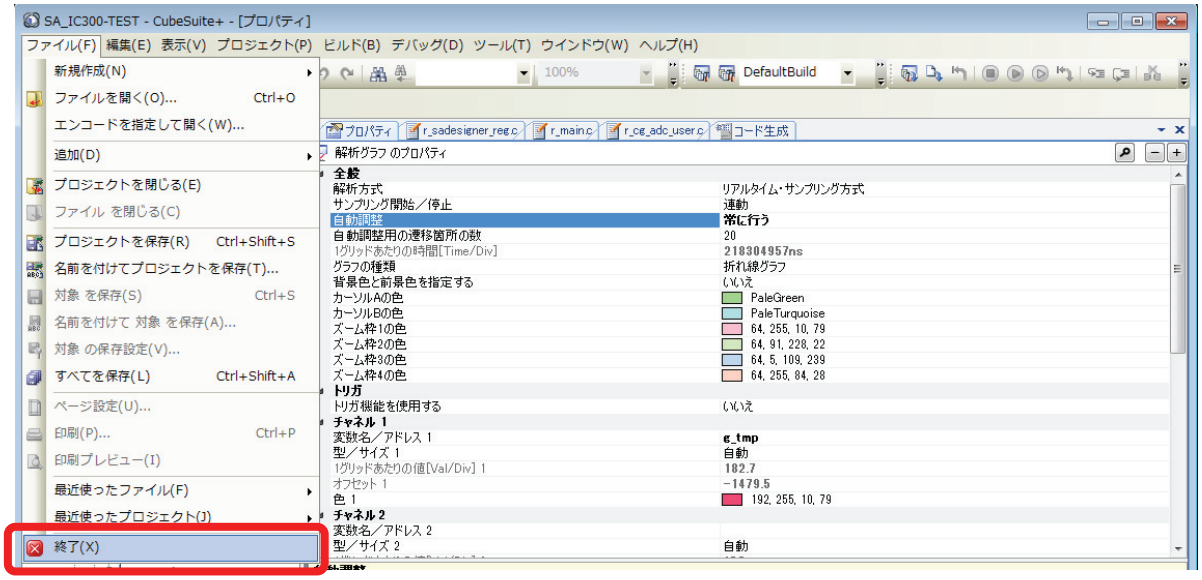


プロジェクトを保存して、CubeSuite+を終了します。



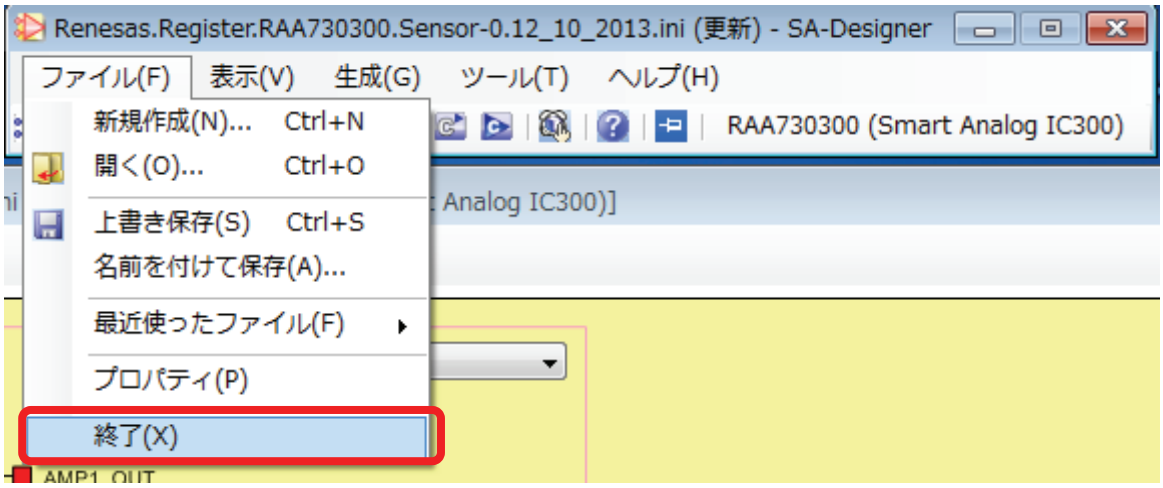
「プロジェクト」タブから「プロジェクトを保存」を選択下さい。

<参考> 次回 CubeSuite+を起動して本プロジェクトを呼び出せば、上記で設定した各内容が自動で呼び出されます。



「ファイル」タブから「終了」を選択して、CubeSuite+を終了させてください。

続いて SA-Designer を終了します。



「ファイル」タブから「終了」を選択して、SA-Designer も終了させてください。

今回使用した Renesas VA の AFE レジスタファイルの保存を聞かれますので、必要に応じて保存してください。

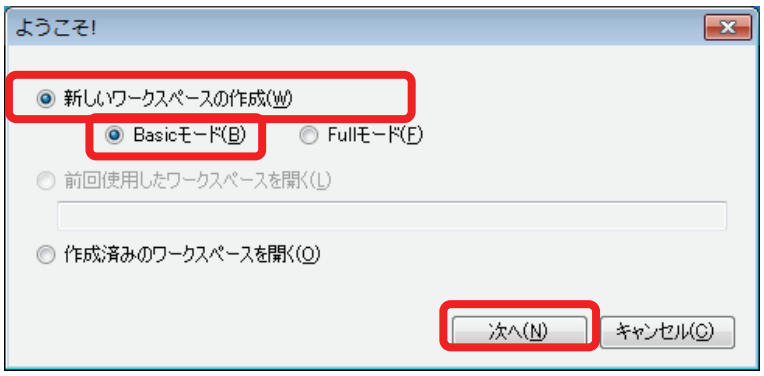
この後今回作成したプログラムをマイコンに書き込みますので、評価ボード、E1、PC の接続はそのままにしてください。

7.17　プログラムをマイコンに書込む

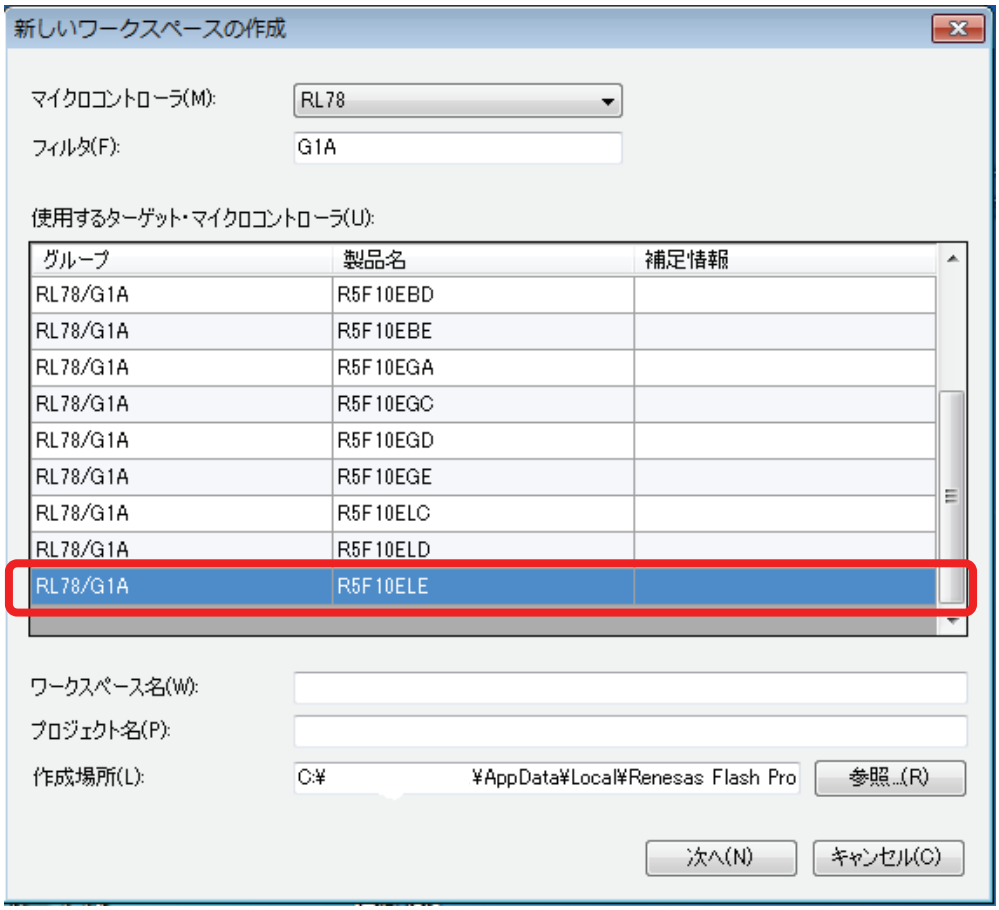
続いて完成したプログラムを RL78/G1A へ書き込んでみましょう。まずは書込みツールを起動させます。

Windows のスタートメニューから書込みツールを起動します。

[Renesas Electronics Utilities]→[書き込みツール]→[Renesas Flash Programmer]



「新しいワークスペースの作成」「Basic モード」を選択し次へボタンをクリックしてください。



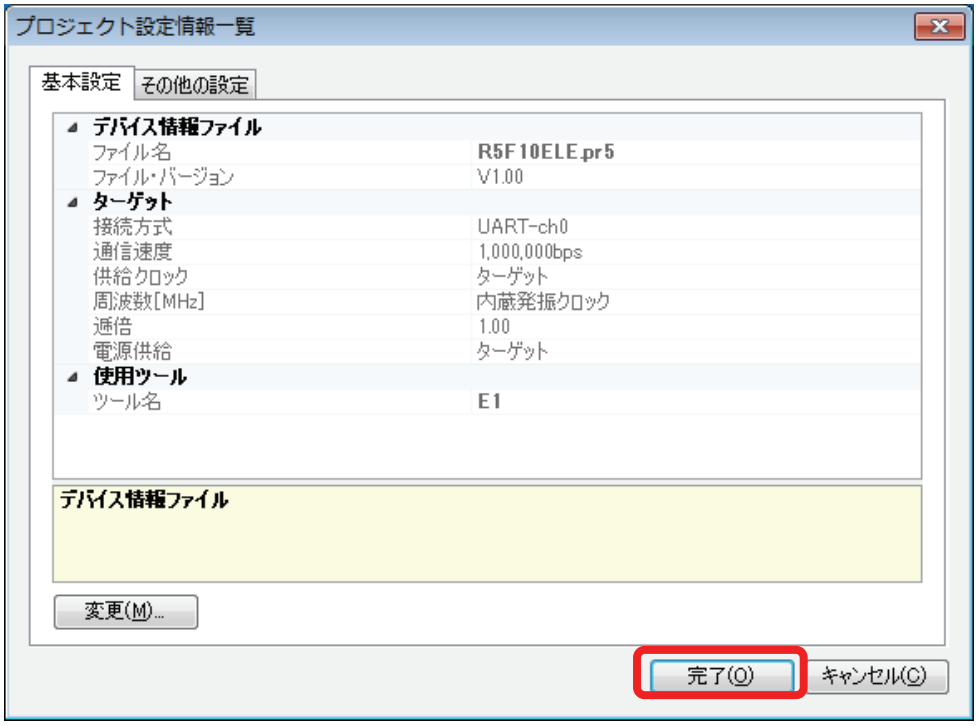
新しいワークスペースの作成画面でグループ：「RL78/G1A」製品名：「R5F10ELE」を選択し、任意の「ワークスペース名」を入力して、「次へ」ボタンをクリックしてください。

(プロジェクト名はワークスペース名と同じ名前が自動入力されます。)

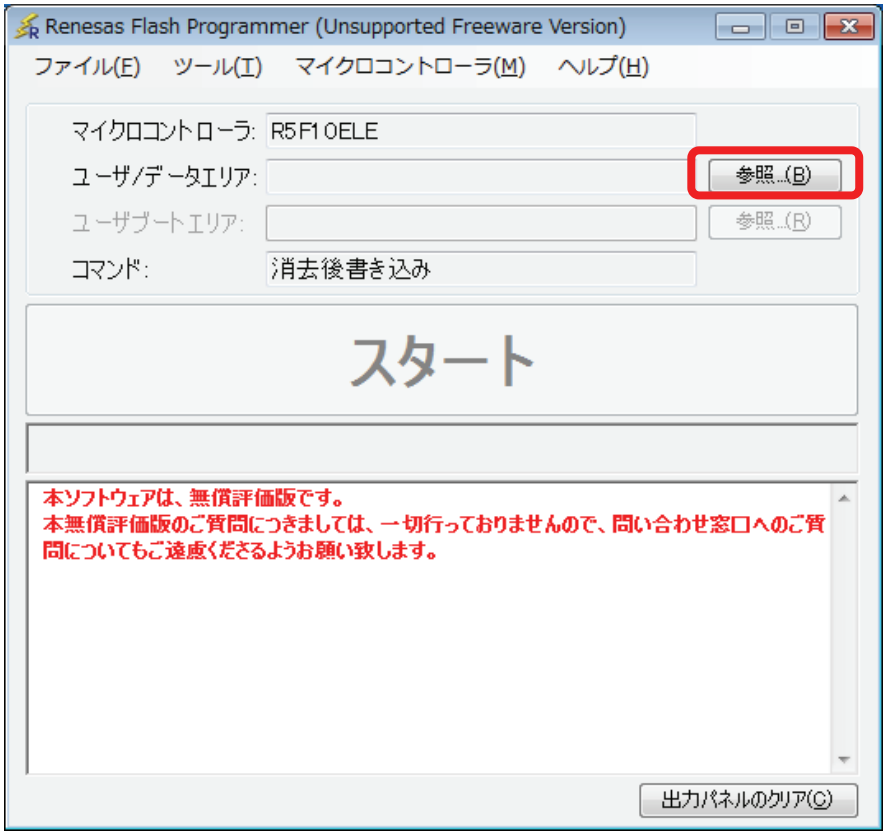
使用ツールが「E1」になっていることを確認して「次へ」をクリックしてください。

通信速度はデフォルトで選択された速度をそのままお使いください。「次へ」をクリックしてください。

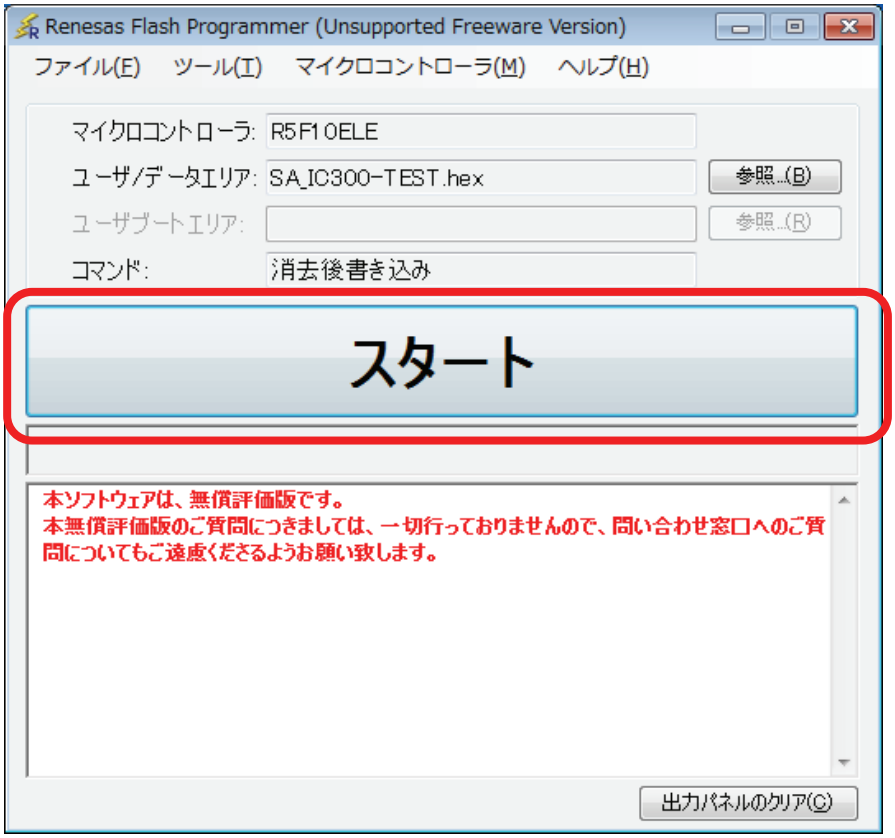
何も選択せず、「次へ」をクリックしてください。



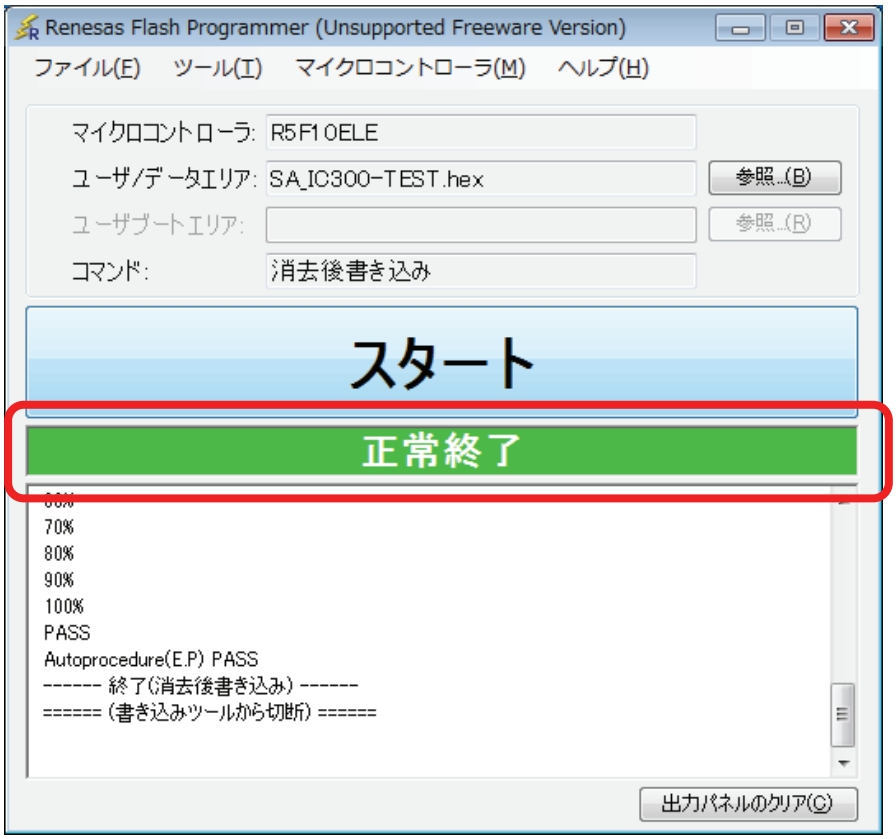
「完了」をクリックしてください。



「参照」ボタンをクリックして、今回作成した CubeSuite+のプロジェクトファイルの中にある「ファイル名.hex」を指定してください。



「スタート」ボタンを押すと書き込みが始まります。



「正常終了」が表示されたら、書込みは成功です。

「ファイル」タブから「終了」を選択して終了させてください。

以上でプログラムの書込みは完了です。

PC -> E1 -> 評価ボード(TSA-IC 300)の各配線を外してください。

PC -> 評価ボード(TSA-IC 300)の USB ケーブルも外してください。

以上で SA-Designer～CubeSuite+でのプログラム作成は完了です。

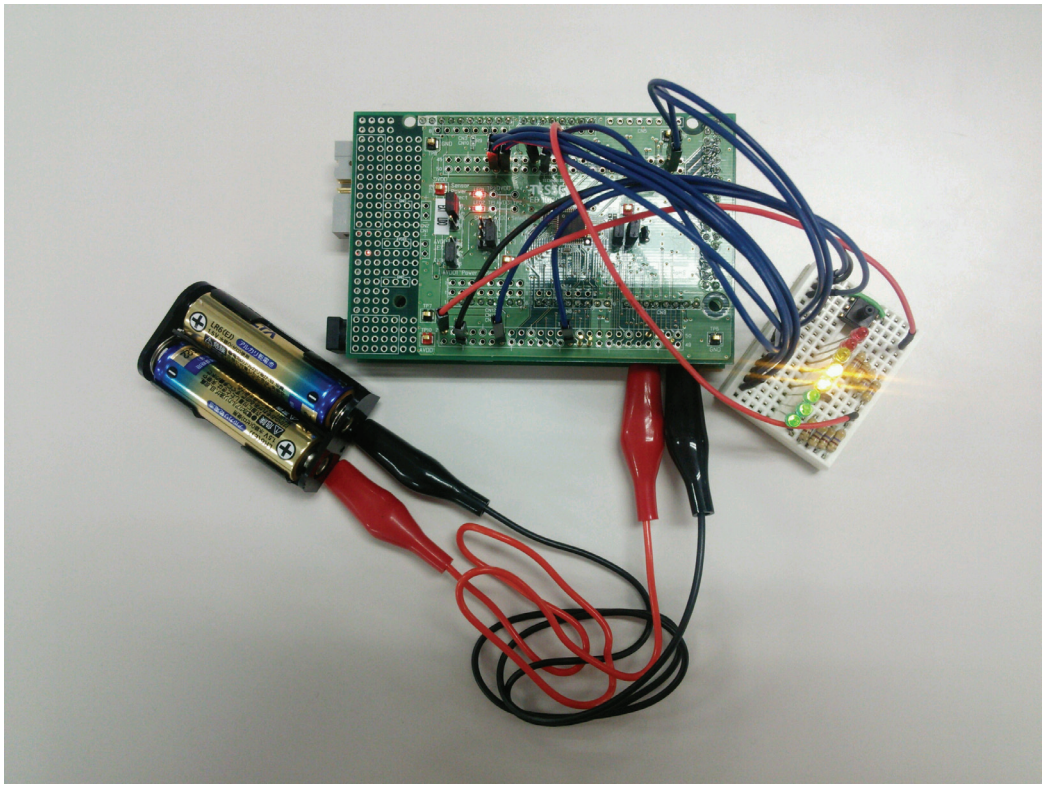
最後に動作を確認します。

8. 実機で動作確認してみよう！

プログラムが無事完成したら、実機で動作確認してみましょう。

評価ボード(TSA-IC 300)に電池ケースを接続し電池を入れて下さい。

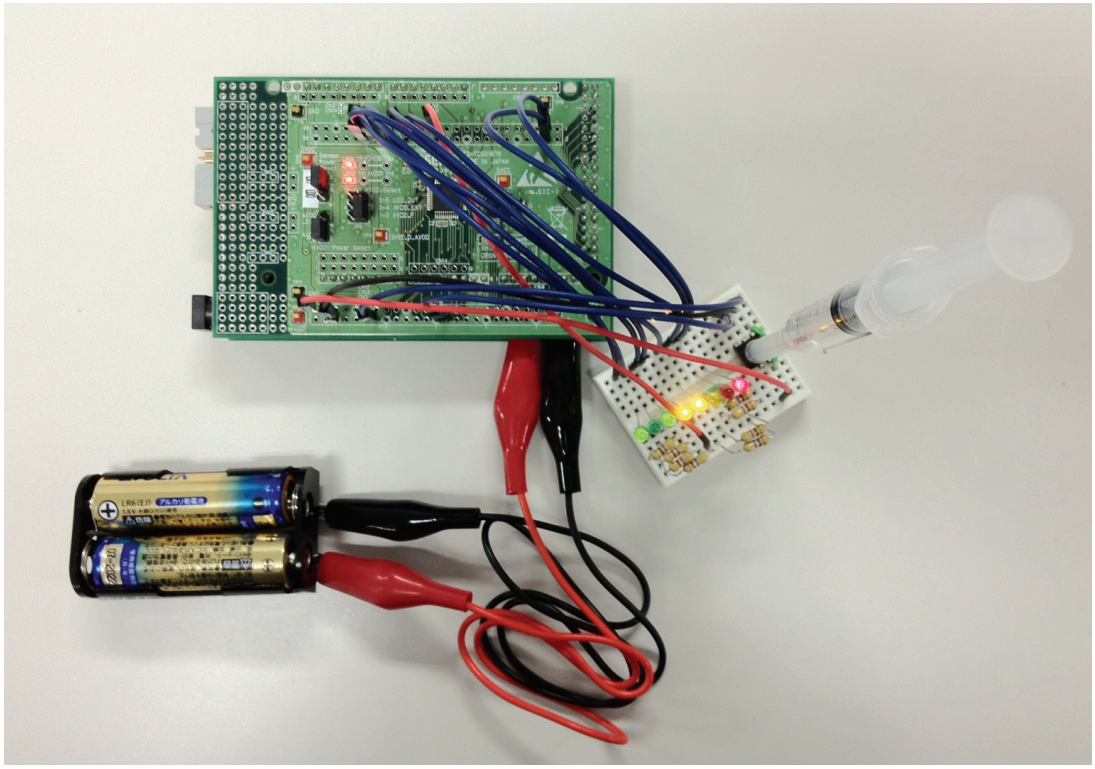
評価ボード(TSA-IC 300)にある RESET(SW4)スイッチを押して、評価ボード上の IC を一度リセットしてください。



リセット直後は、全ての LED が点灯します。そこから約 3 分後、緑色 LED×3 つに加え、緑色の LED 側から一つ目、二つ目までの黄色 LED が点灯した状態となります。

この状態での気圧を基準として、気圧が高くなると赤色側の LED が点灯、気圧が低くなると緑色側の LED が点灯します。

感度を 10hPa とした場合、ビルの 1 階から 15 階程度の高さの間で、LED×1 つが消灯、または点灯します。ビルの 1 階でリセットした場合、ビルの 15 階程度の高さまで上ると、基準の状態から黄色 LED が一つ消灯します。逆に、ビルの 15 階程度の高さでリセットし、1 階の高さまで下ると、基準の状態から赤色 LED 側に近い黄色 LED が一つ点灯します。



上の写真のように、スポイト（上写真は、目盛付きスポイト）などを利用し、センサが感知する気圧を強制的に変化させることによって、動作を確認することも可能です。

以上で Smart Analog の開発は完了です。

余力があれば、以下を試してみてください。

- Renesas VA でセンサのゲインを変更して感度を調整してみる。
- マイコンのプログラムを編集して気圧に対する感度を変更してみる。
- LED の個数を増やしてみる。
- LED の点灯する方法も変更してみる。

< うまく動かない方へ >

うまく動かない方は以下を確認してみてください。

- 7. の内容をもう一度確認して設定、プログラムの内容が間違っていないか？
- 評価ボードとセンサ、LED の接続を確認ください。

9.　最後に

この資料では、Smart Analog を初めてお使い頂く皆様を対象に、センサのシミュレーションからマイコンのプログラム開発、実機動作確認までを解説してきました。

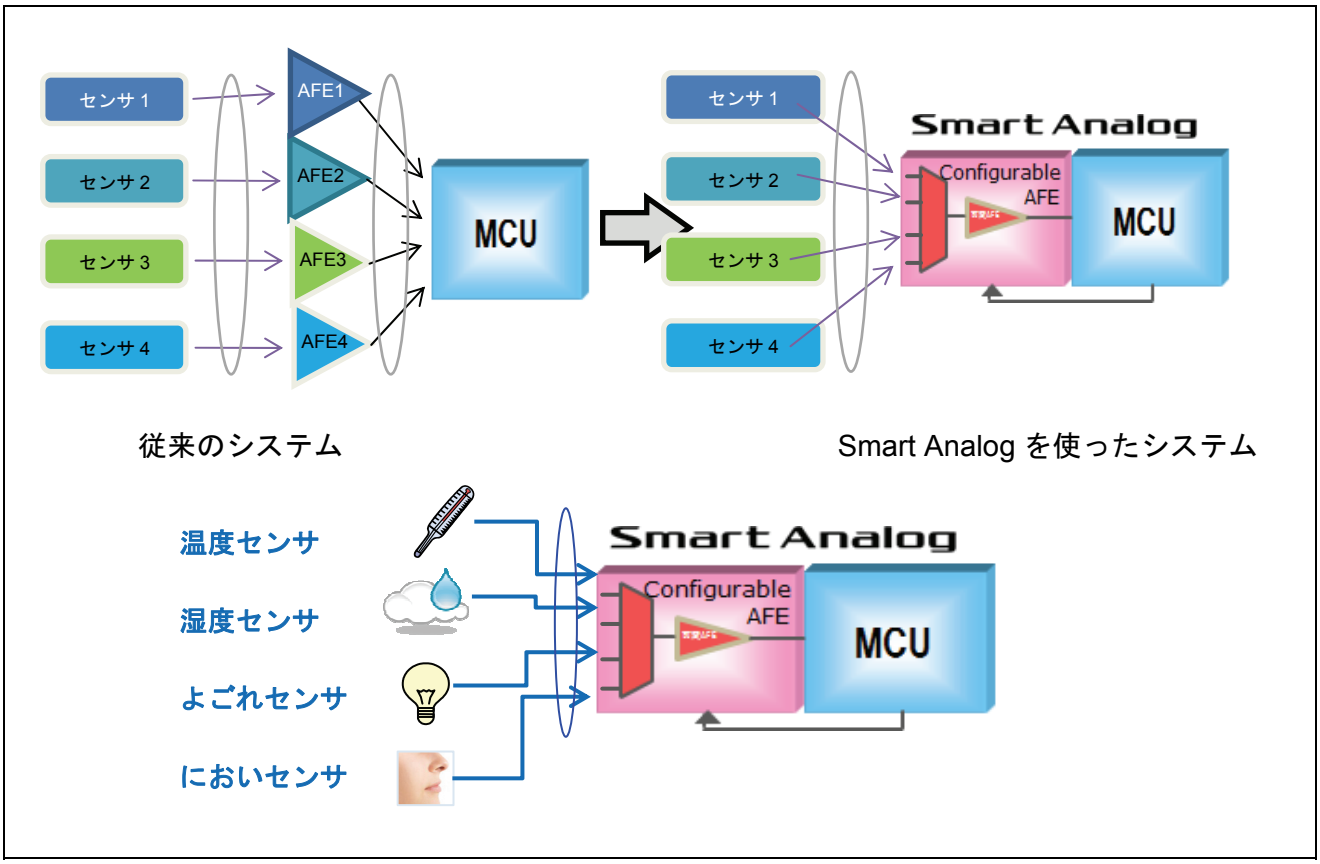
この資料で取得して頂いた知識を元に、今後は別のセンサの制御や Smart Analog 内蔵の機能(フィルタや同期検波など)の利用、などを是非チャレンジしてください。

なお、Smart Analog には他にも便利な機能を沢山搭載していますので、代表例を 3 つご紹介します。

9.1　マルチセンサ制御

Smart Analog には入力段に差動対で 6ch のマルチプレクサを搭載しています。

これと内蔵 AMP のリコンフィギュラブル機能(反転・非反転・差動・I/V 等を動的に変更できる)を利用して制御方法の異なる複数のセンサを同時に制御する「マルチセンサ制御」が実現できます。

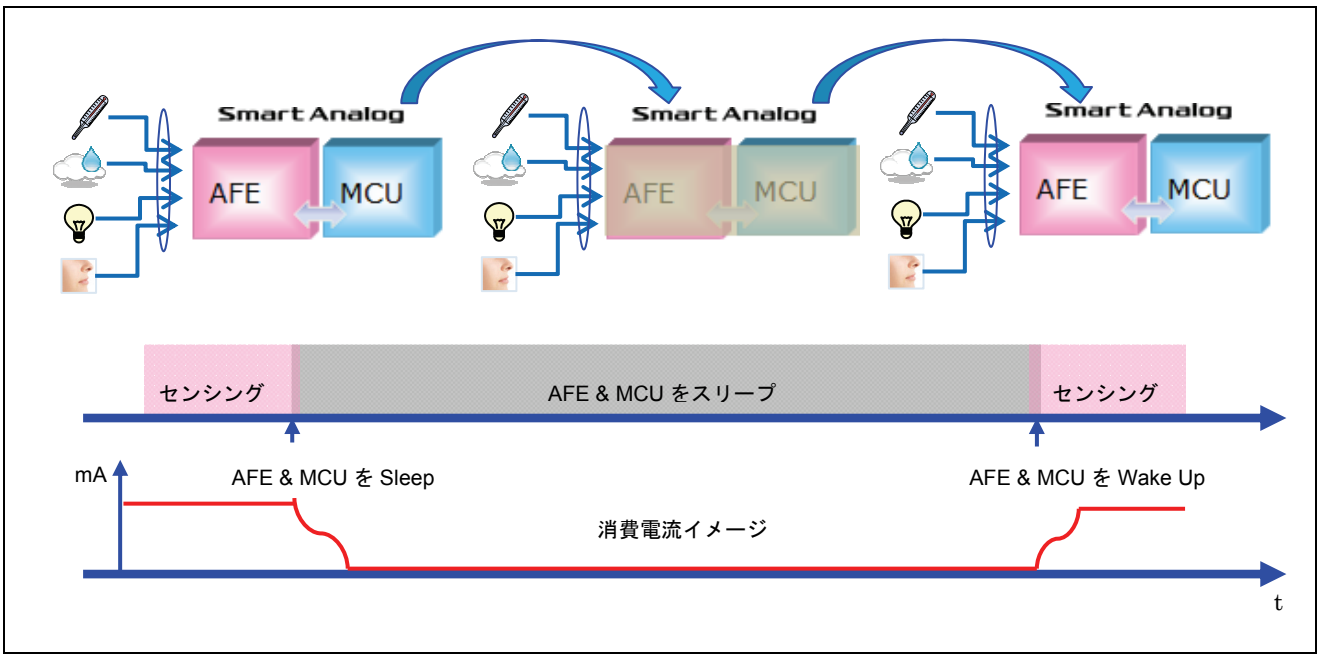


Smart Analog を使ったマルチセンサ制御の例

9.2　間欠動作による低消費電流動作

Smart Analog には内蔵する個々のアナログ回路ブロックを個別に ON/OFF させることができます。

この機能を利用して、センシングしていない時はアナログ回路ブロックを OFF、センシングするときに ON というように制御すれば消費電流を削減することができます。

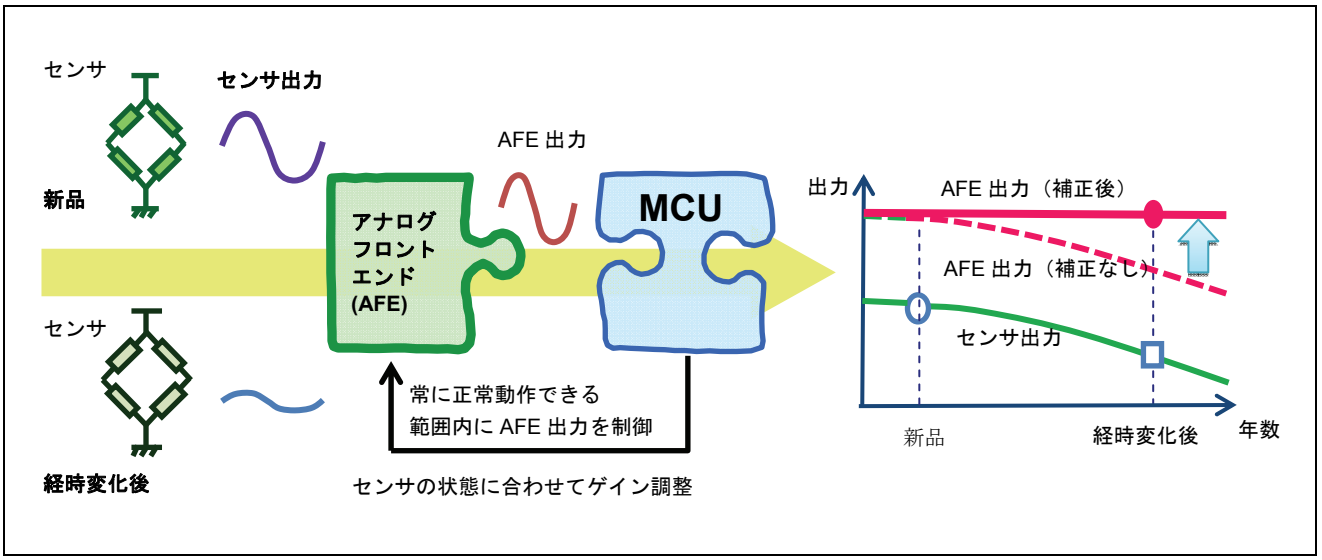


Smart Analog を使った低消費電流動作の例

9.3　自動補正制御

Smart Analog は内蔵アナログ回路のゲインやオフセットなどのパラメータを動的に変更することができます。

この機能を利用して、汚れや経年によるセンサの劣化を自動で補正することができます。



Smart Analog を使った自動補正制御の例

これら以外にも様々な付加機能の実現が可能です。

是非みなさんのアイデアを Smart Analog で試してみてください。

ホームページとサポート窓口

ルネサス エレクトロニクスホームページ
http://japan.renesas.com/

お問合せ先
http://japan.renesas.com/contact/

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
Rev.1.00	2014.02.28	---	初版発行

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 未使用端子の処理
   【注意】未使用端子は、本文の「未使用端子の処理」に従って処理してください。
   CMOS製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI周辺のノイズが印加され、LSI内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。
2. 電源投入時の処置
   【注意】電源投入時は，製品の状態は不定です。
   電源投入時には、LSIの内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。
   外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。
   同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. リザーブアドレス（予約領域）のアクセス禁止
   【注意】リザーブアドレス（予約領域）のアクセスを禁止します。
   アドレス領域には、将来の機能拡張用に割り付けられているリザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
4. クロックについて
   【注意】リセット時は、クロックが安定した後、リセットを解除してください。
   プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。
   リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
5. 製品間の相違について
   【注意】型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。
   同じグループのマイコンでも型名が違うと、内部ROM、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ輻射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器・システムの設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因して、お客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
2. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
3. 本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害に関し、当社は、何らの責任を負うものではありません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を改造、改変、複製等しないでください。かかる改造、改変、複製等により生じた損害に関し、当社は、一切その責任を負いません。
5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、
   各品質水準は、以下に示す用途に製品が使用されることを意図しております。
   標準水準： コンピュータ、OA機器、通信機器、計測機器、AV機器、
   家電、工作機械、パーソナル機器、産業用ロボット等
   高品質水準：輸送機器（自動車、電車、船舶等）、交通用信号機器、
   防災・防犯装置、各種安全装置等
   当社製品は、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（原子力制御システム、軍事機器等）に使用されることを意図しておらず、使用することはできません。たとえ、意図しない用途に当社製品を使用したことによりお客様または第三者に損害が生じても、当社は一切その責任を負いません。なお、ご不明点がある場合は、当社営業にお問い合わせください。
6. 当社製品をご使用の際は、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他の保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
8. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制するRoHS指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
9. 本資料に記載されている当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。また、当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途に使用しないでください。当社製品または技術を輸出する場合は、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。
10. お客様の転売等により、本ご注意書き記載の諸条件に抵触して当社製品が使用され、その使用から損害が生じた場合、当社は何らの責任も負わず、お客様にてご負担して頂きますのでご了承ください。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。

注1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。
注2. 本資料において使用されている「当社製品」とは、注１において定義された当社の開発、製造製品をいいます。

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口 http://www.renesas.com

※営業お問合せ窓口の住所は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス株式会社　〒100-0004　千代田区大手町2-6-2（日本ビル）

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：http://japan.renesas.com/contact/