Smart Analog

SA-Designer を使ったシステム開発手順（R8C ファミリ編）

R20AN0247JJ0100
Rev.1.00
2013.03.25

要旨

本アプリケーションノートは、Smart Analog を使用し、R8C を想定した環境で簡単なシステムの開発手順を以下のとおり解説します。

・アナログフロントエンド回路の設計

・プログラムの作成

・回路データの登録とビルド

・動作確認

目次

1.　概要

本アプリケーションノートは、Smart Analog を使用し、R8C を想定した環境で簡単なシステムの開発手順を解説するものです。

Smart Analog IC に内蔵されている温度センサを使用し、温度に応じて LED の点滅間隔を変えるシステムです。

R8C CPU ボードと Smart Analog IC を使用し、SA-Designer と High-performance Embedded Workshop を使ったロードモジュールの作成、プログラムの動作確認を説明します。

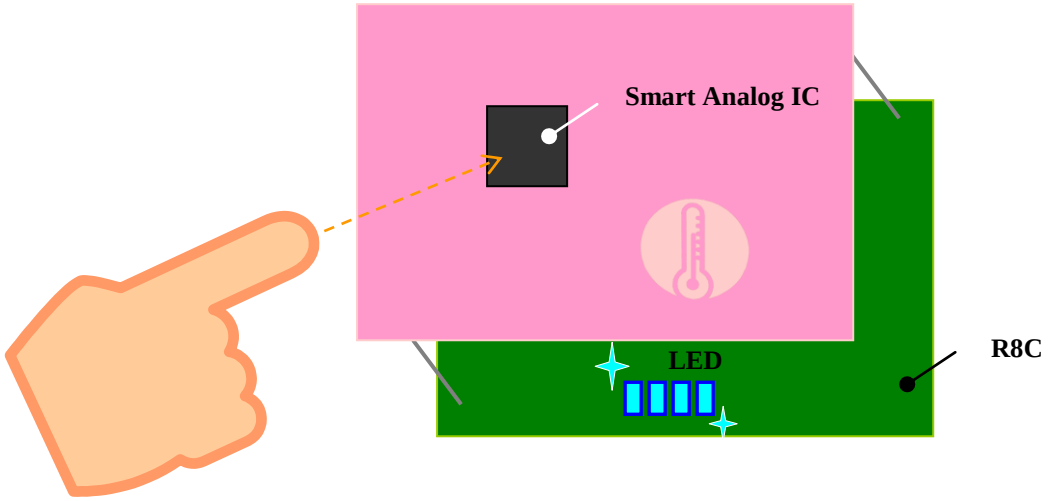


図 1.　ユーザシステム

1.1　開発環境

本アプリケーションノートでは、以下の開発環境を使用しています。

1.1.1　ハードウェア

・ホスト PC
・CPU ボード：　R8C、Smart Analog IC500
・E8a エミュレータ

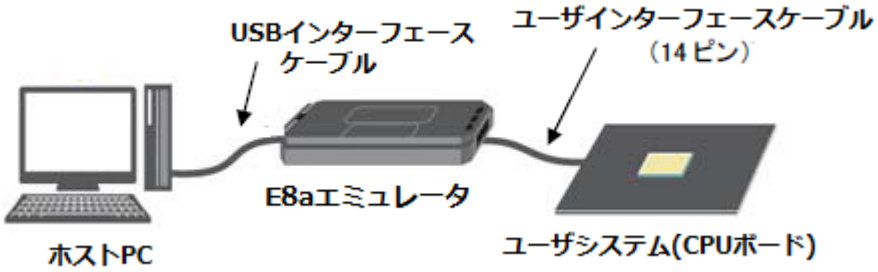


図 2.　ハードウェア構成

1.1.2　ソフトウェア

・SA-Designer　（V1.00.00）
・統合開発環境　High-performance Embedded Workshop　（Version 4.09.01.007）

2.　開発手順

2.1　　概要

作成するシステムの構築手順を以下に説明します。

構築では、SA-Designer と High-performance Embedded Workshop を使用します。
システムの開発手順を以下に示します。

① アナログフロントエンド回路の設計

：SA-Designer を使用して、アナログフロントエンド回路を設計します。

② プログラム作成

：High-performance Embedded Workshop を使用して、マイコンのクロックやポート、A/D 変換機能の初期設定､およびシステムを動作させるプログラムを作成します。

③ 回路データの設定プログラム登録

：SA-Designer で生成した C ソースを High-performance Embedded Workshop に登録して、ビルドを行います。

④ 動作確認

：E8a エミュレータを接続して、プログラムをマイコンに書き込み動作を確認します。

※　SA-Designer、および High-performance Embedded Workshop（Version 4.09.01.007 以上）のインストールを完了してから作業を行って下さい。

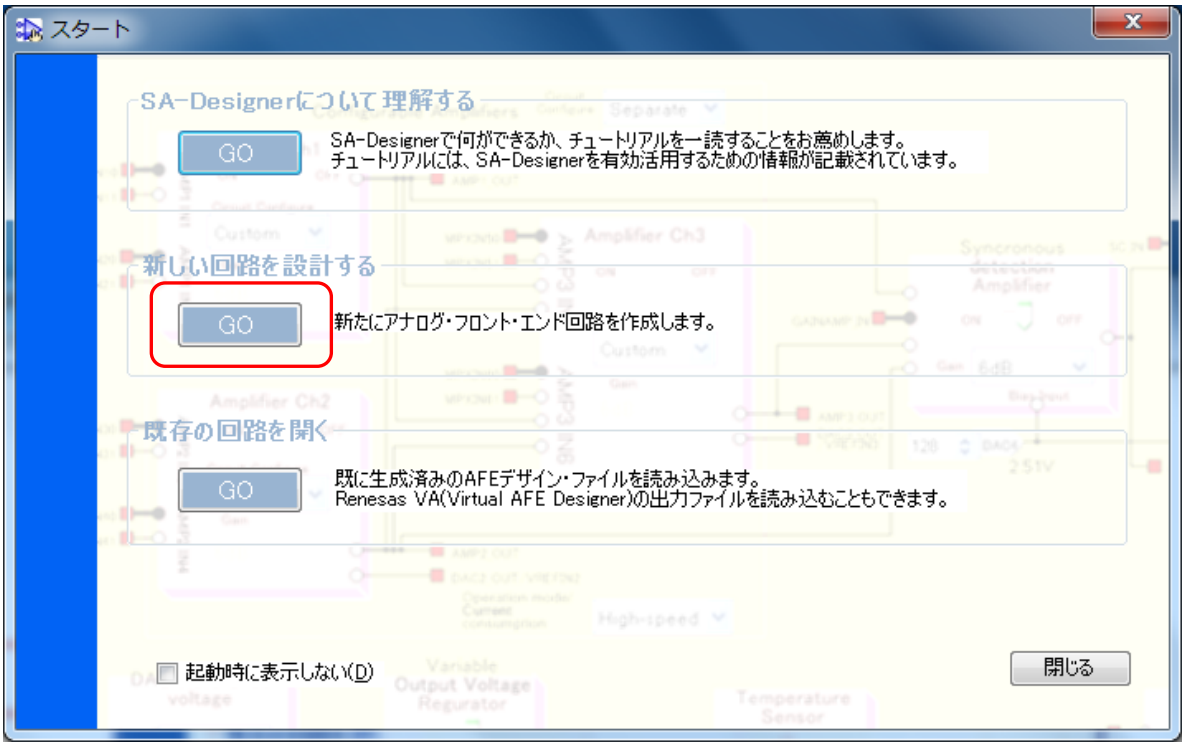
2.1.1　　アナログフロントエンド回路の設計

(1)　SA-Designer の起動

[スタート]→[すべてのプログラム]→[Renesas Electronics Utilities]→[スマート・アナログ・ツール]→[SA-Designer]を選択して SA-Designer を起動します。

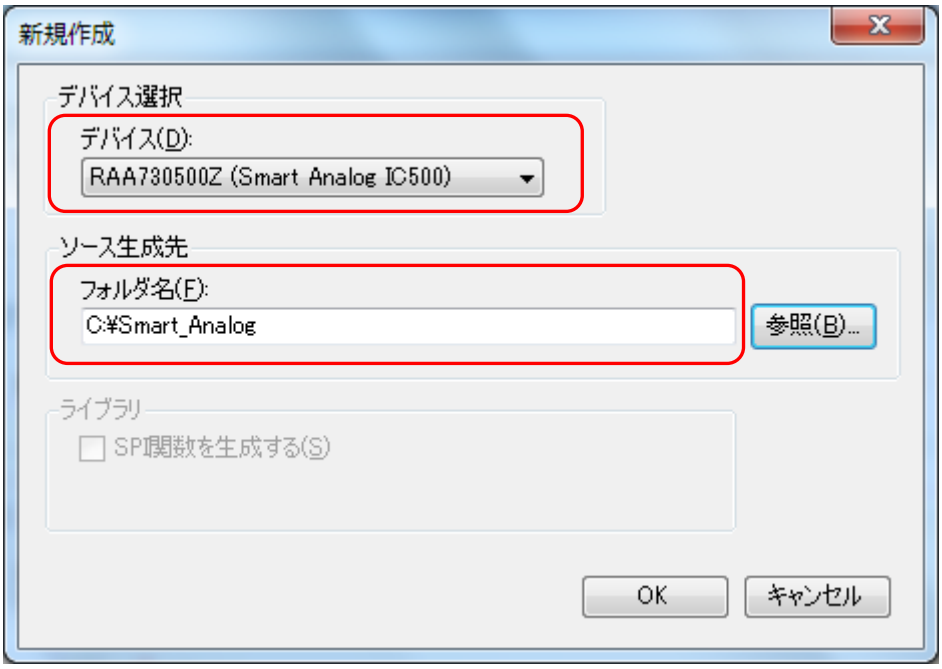
(2)　新しい回路の設計

アナログフロントエンド回路を設計するため、デバイスとソースファイルの生成先を指定します。



[新しい回路を設計する]の[GO]をクリックします。

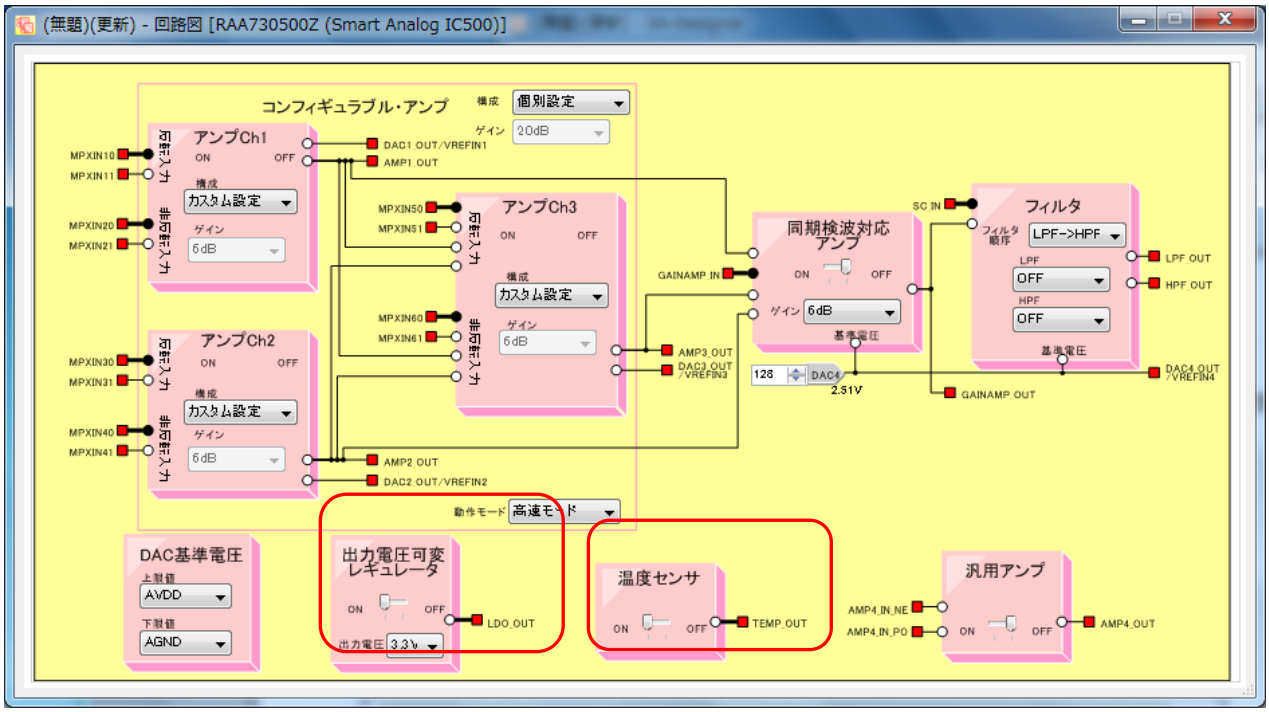
[新規作成]ダイアログでデバイスとソース生成先を指定します。



[デバイス]	“RAA730500Z (Smart Analog IC500)”を選択します。
[フォルダ名]	任意で指定します。ここでは“SmartAnalog”を指定します。 存在するディレクトリを指定します。

(3)　回路図の作成

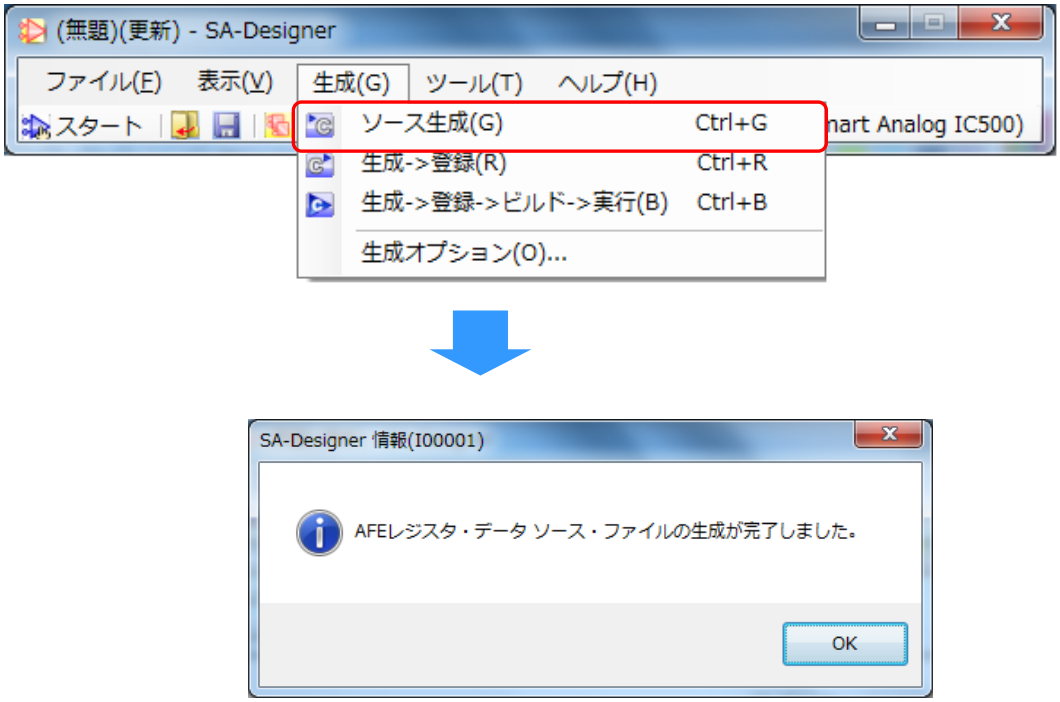
温度センサを使用するようにアナログフロントエンド回路を設計します。
以下に示すように、回路図の初期状態から設定を変更してください。



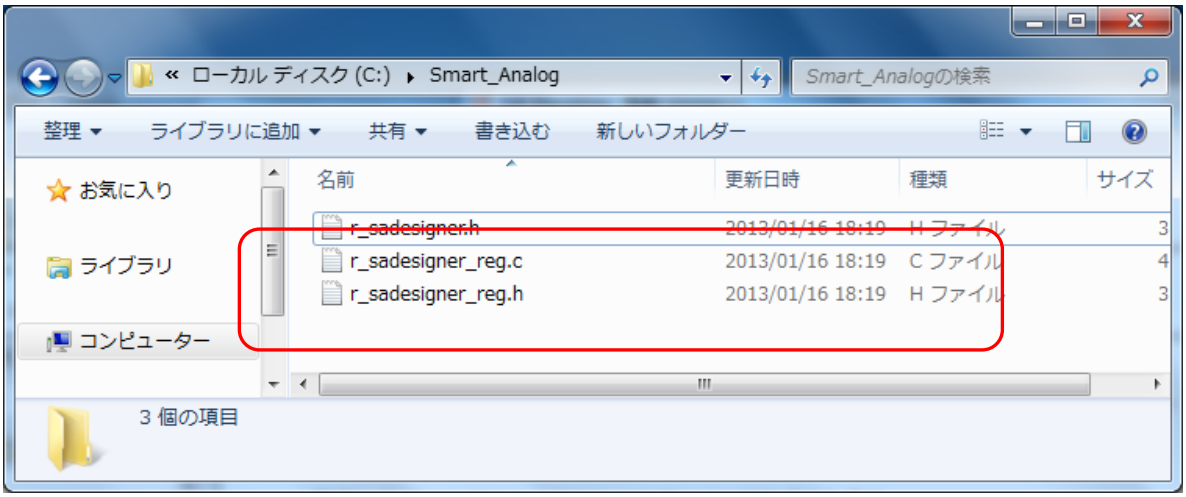
[出力電圧可変レギュレータ]	スイッチを“ON”に設定します。
[温度センサ]	スイッチを“ON”に設定します。

(4)　ソースファイルの生成

設計した回路のデータを設定するためのソースファイルを生成します。
[生成]ー[ソース生成]をクリックすると生成完了のダイアログが表示されます。



指定したフォルダに 3 つの C ソースファイルが生成されます。

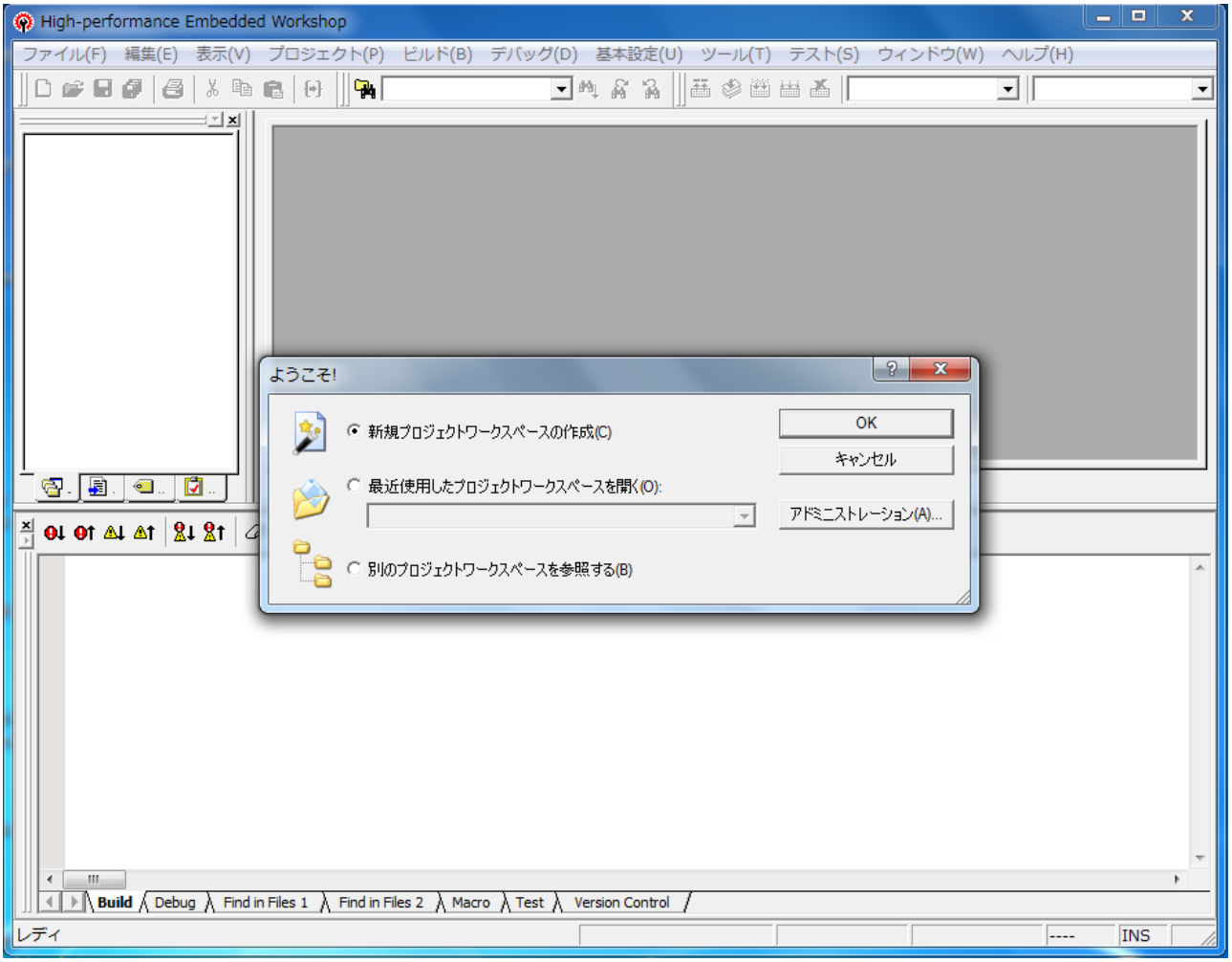


2.1.2　　プログラムの作成

(1)　High-performance Embedded Workshop の起動

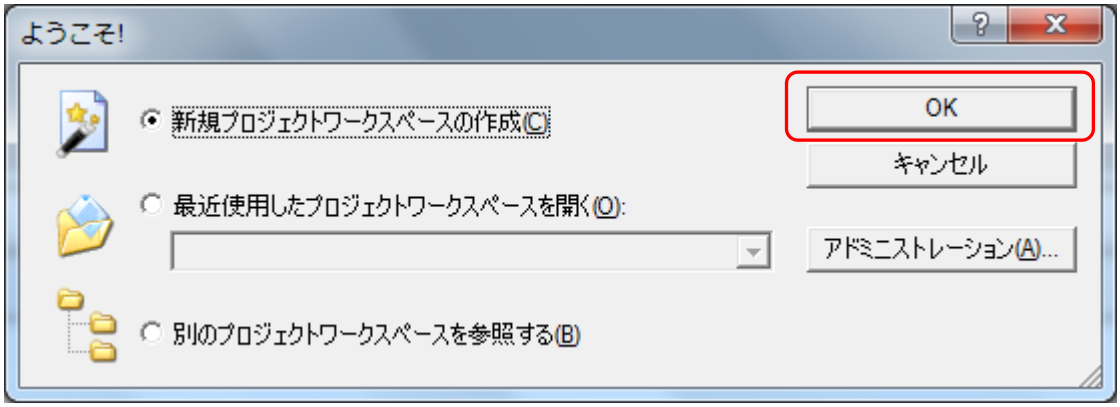
[スタート]→[すべてのプログラム]→[Renesas]→[High-performance Embedded Workshop]→[High-performance Embedded Workshop]を選択して High-performance Embedded Workshop を起動します。

なお、High-performance Embedded Workshop はあらかじめインストールしておく必要があります。



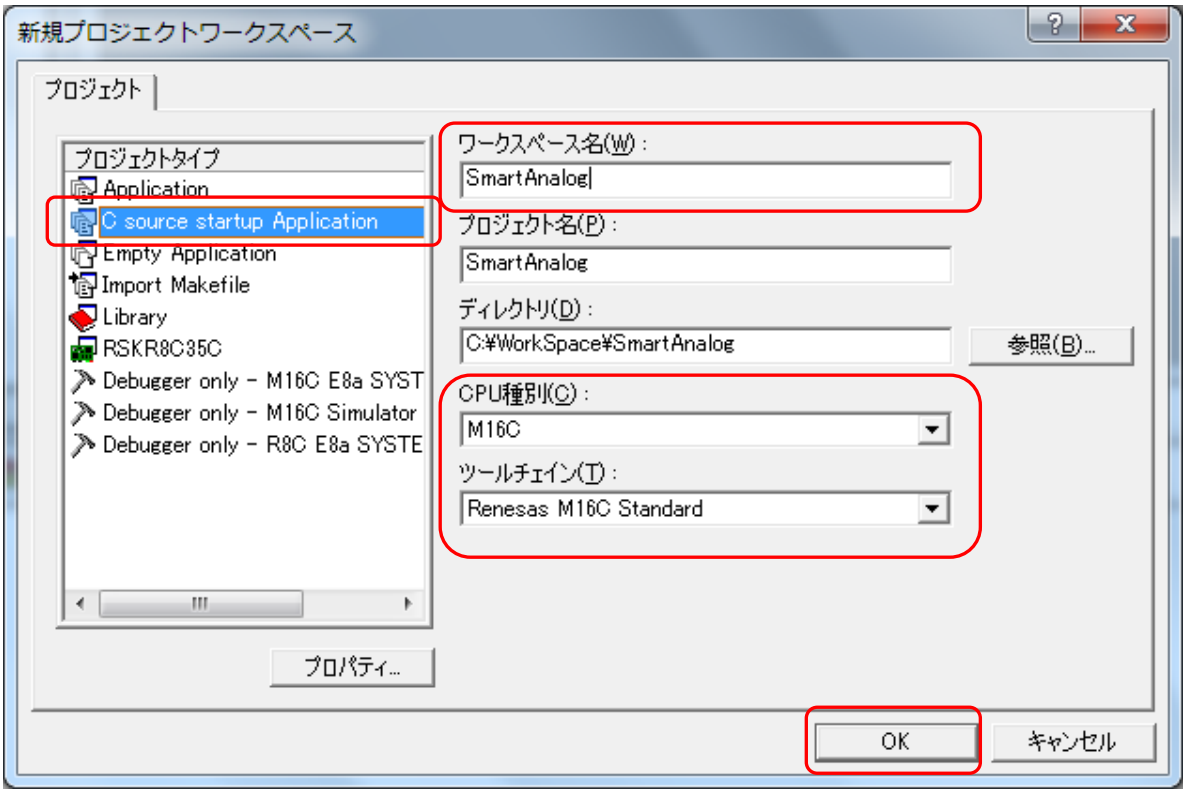
(2)　プロジェクト新規作成

High-performance Embedded Workshop でプロジェクトワークスペースを作成します。



[新規プロジェクトワークスペースの作成]の[OK]をクリックします。

[新規プロジェクトワークスペース]ダイアログでプロジェクトの情報を設定します。

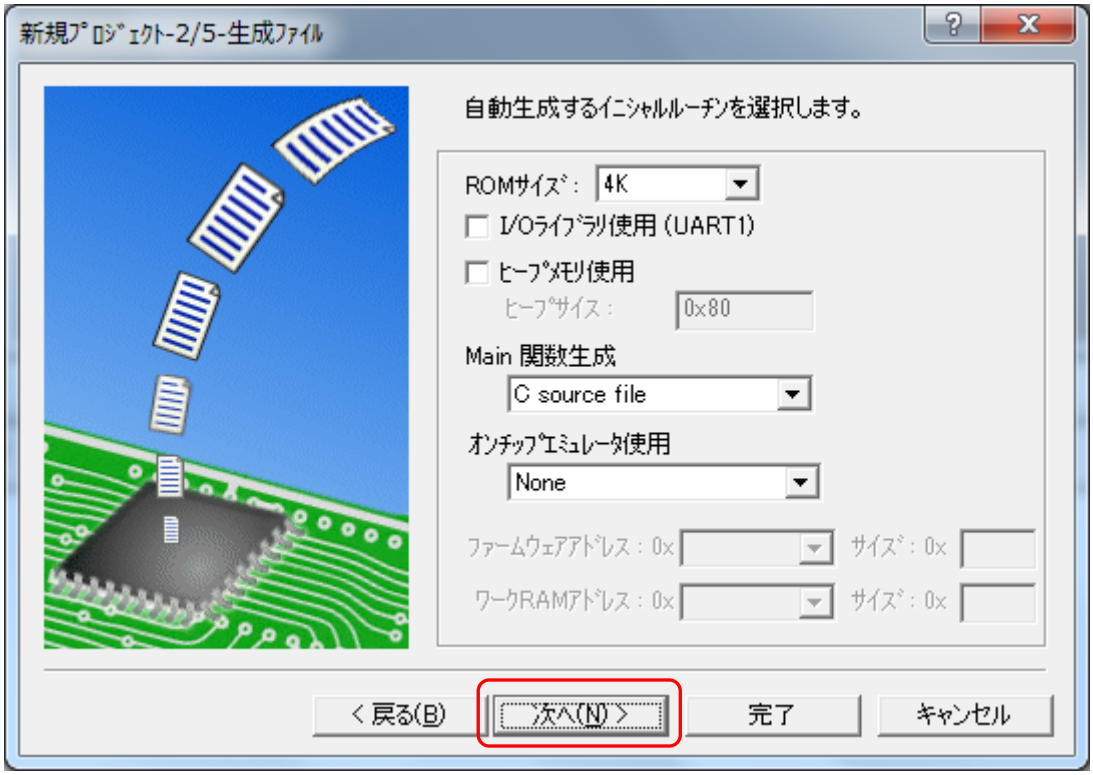


[プロジェクトタイプ]	“C source start up Application”を指定します。
[ワークスペース名]	任意の指定です。ここでは“SmartAnalog”を指定します。
[作成場所]	任意の指定です。デフォルトのディレクトリ”C:¥WorkSpace¥”の後にワークスペース名が自動的に設定されます。
[CPU 種別]	“M16C”を指定します。
[ツールチェイン]	“Renesas M16C Standard”を指定します。

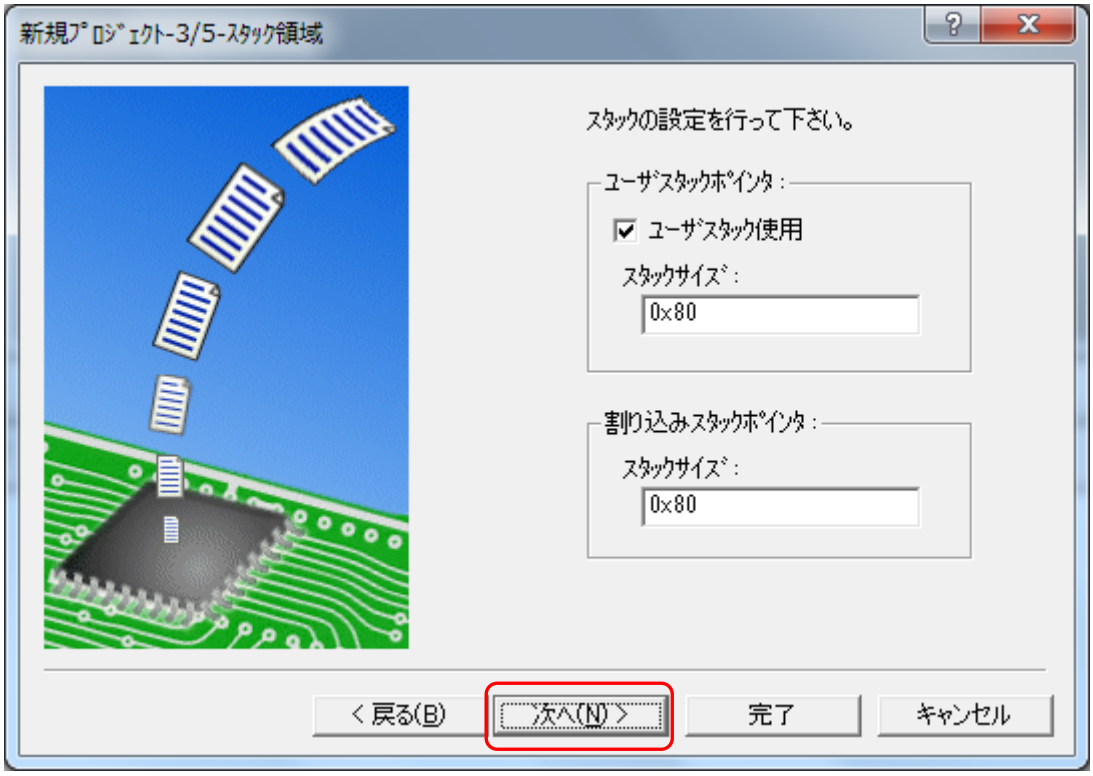
[OK]ボタンをクリックします。

CPU シリーズ、CPU グループをユーザシステムに合わせ選択します。

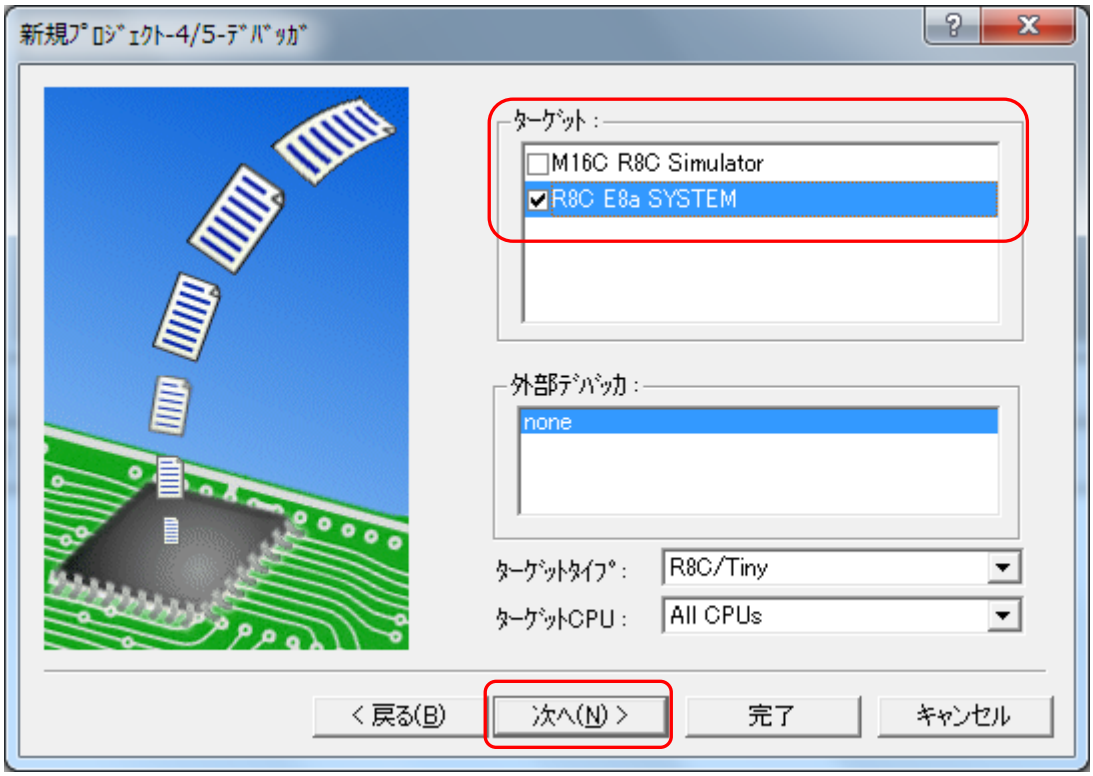
ユーザのシステムに合わせ、High-performance Embedded Workshop が自動生成するイニシャルルーチンを選択します。



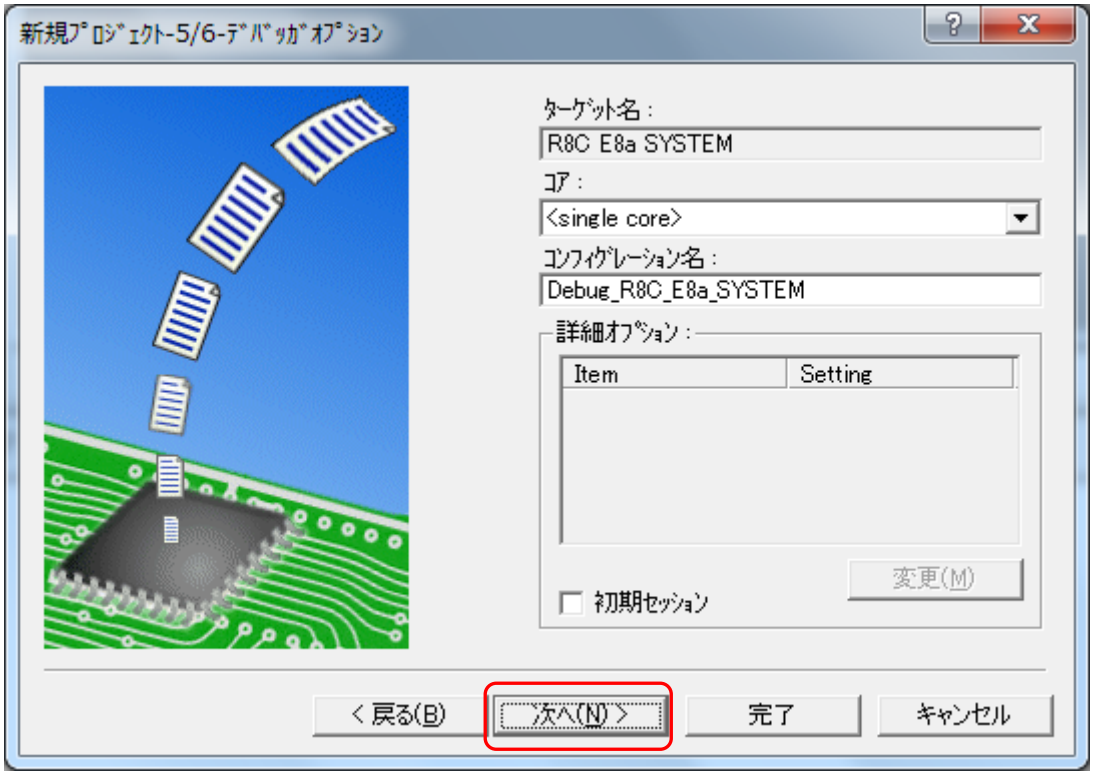
ユーザのシステムに合わせ、スタックの設定を行います。



ユーザシステムに合わせ、ターゲットを選択します。

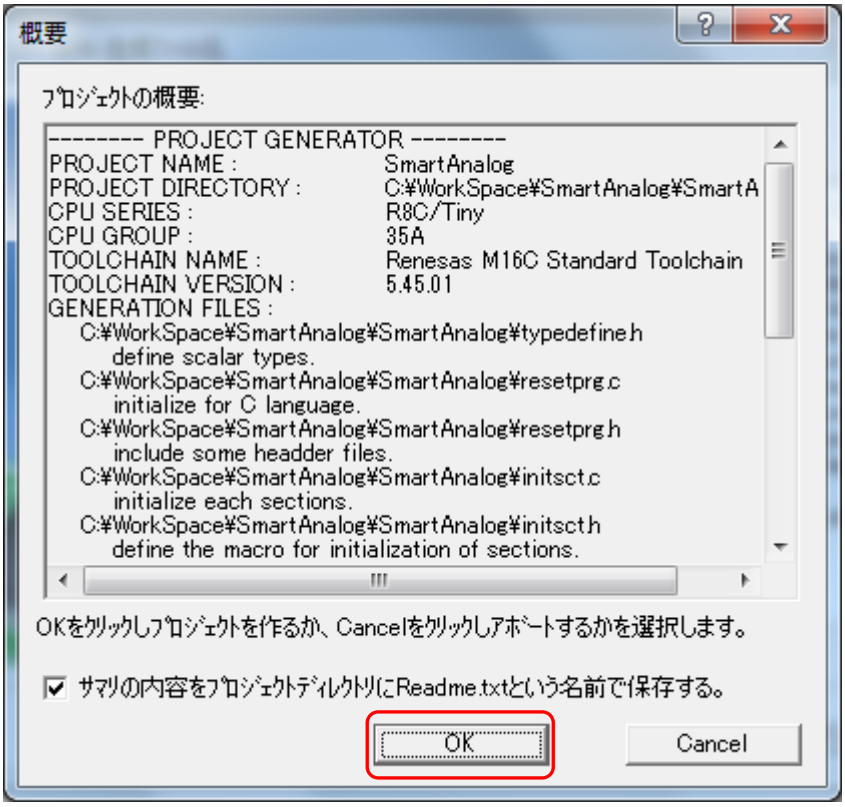


デバッガのオプションを設定します。

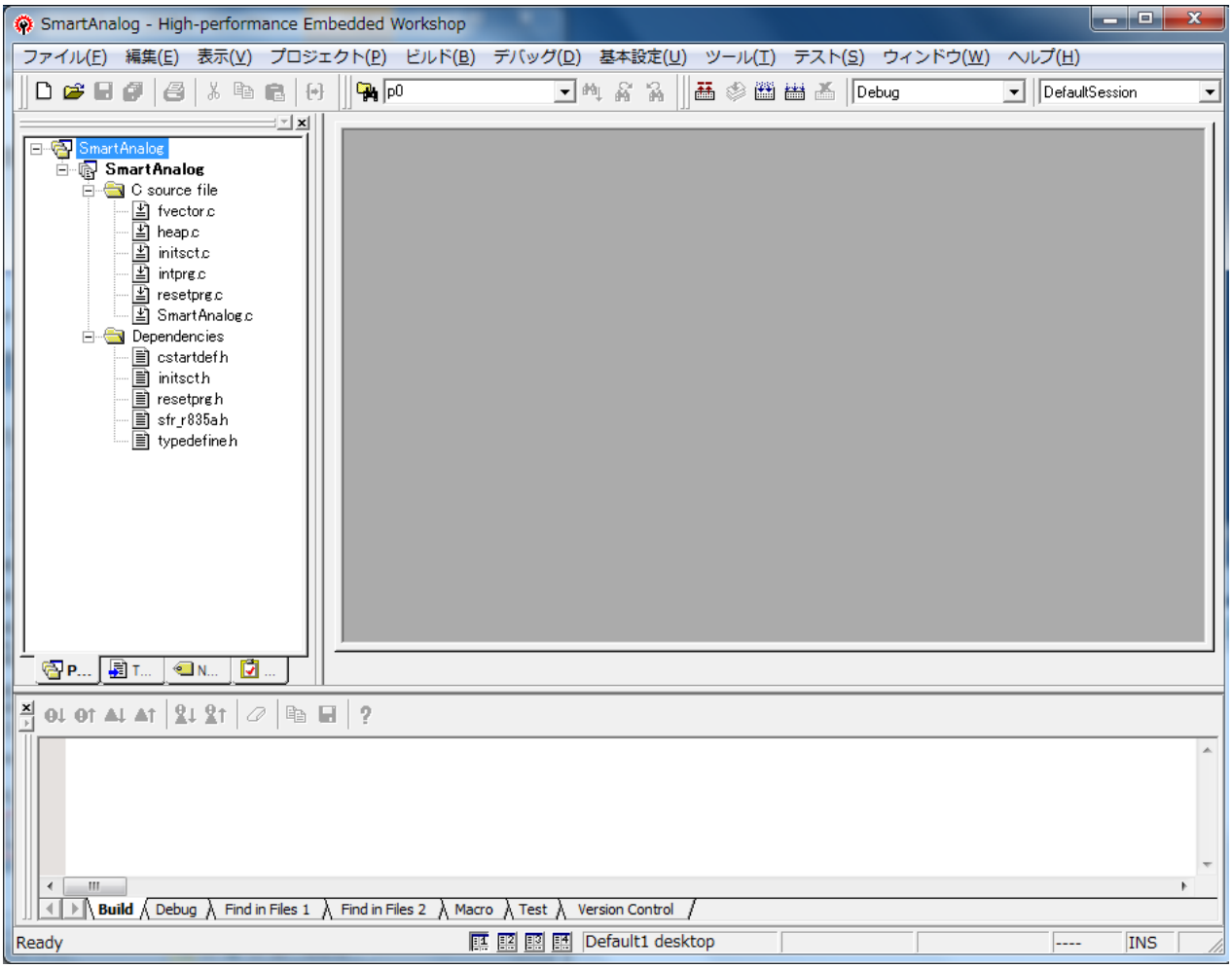


自動生成されるソースファイル名の一覧が表示されます。

プロジェクトの概要が表示され、[OK]ボタンを押すとプロジェクトが作成されます。



プロジェクトが生成され、プロジェクト・ツリーパネルに、作成したプロジェクトの構成がツリー表示されます。

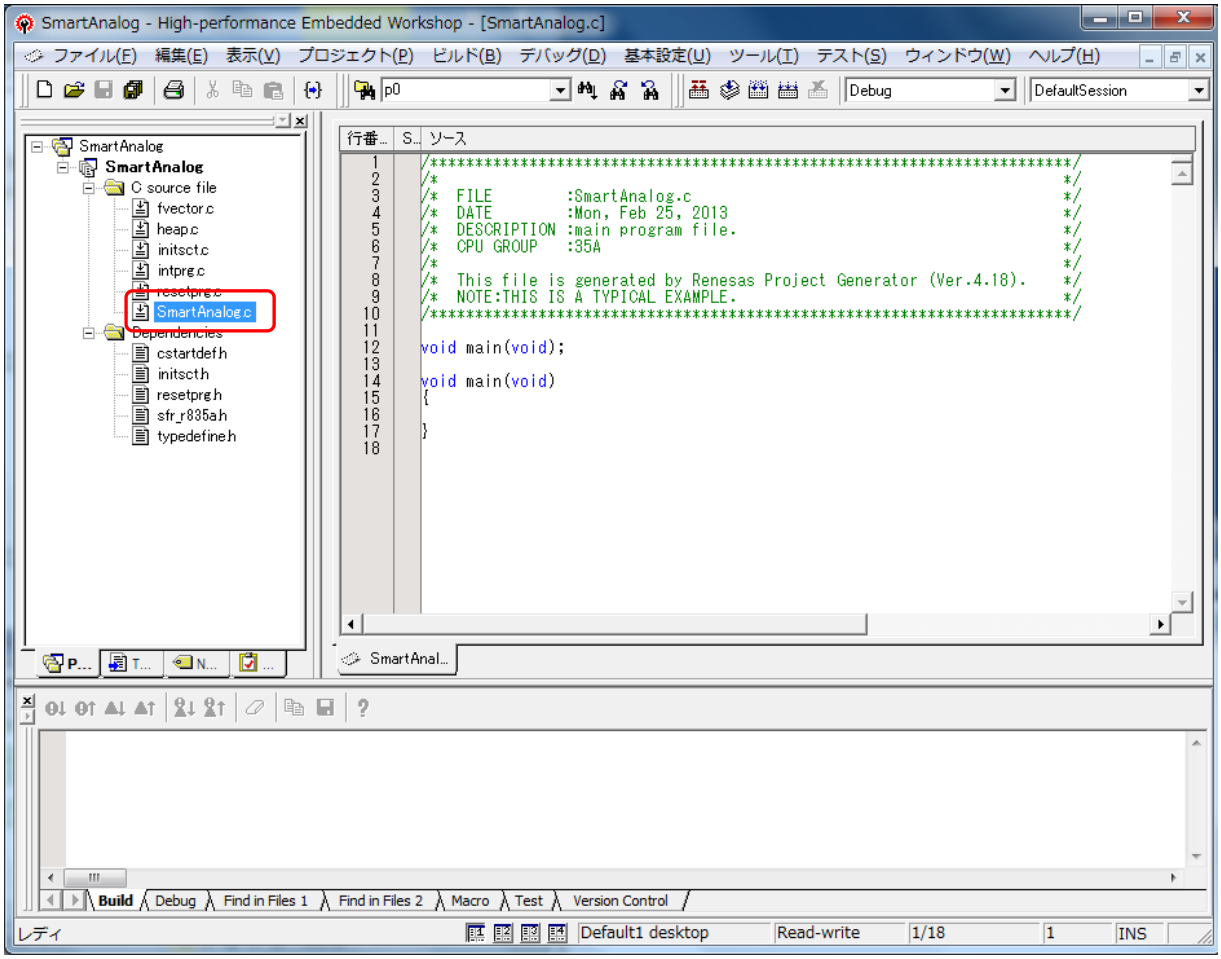


(3)　プログラムの作成

マイコンのクロック設定や、A/D 機能を使用するためのプログラムを作成します。

ここでは、High-performance Embedded Workshop が自動生成したソースファイル SmartAnalog.c および intprg.c にソースコードを記述します。

追加するサンプルプログラムについては『3. サンプルプログラム』をご参照ください。



ソースファイル“SmartAnalog.c”を開きます。

作成するプログラムは、メイン関数、ハードウェアの初期化関数、Smart Analog IC 初期化関数、Smart Analog IC リード/ライト関数、A/D 割り込み処理です。

Smart Analog IC との SPI 通信はクロック同期形シリアルインターフェース(SSU)により通信を行います。

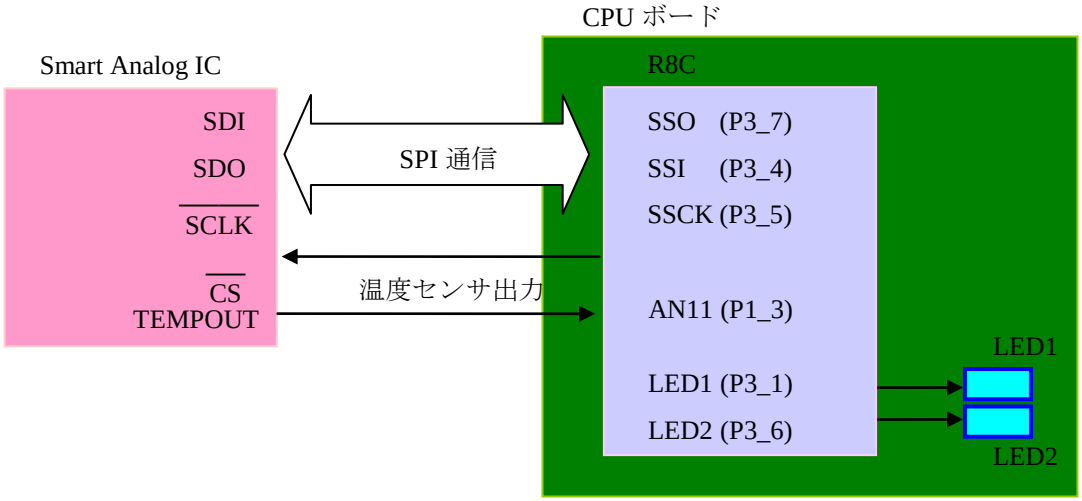


図 3.　ハードウェア接続図

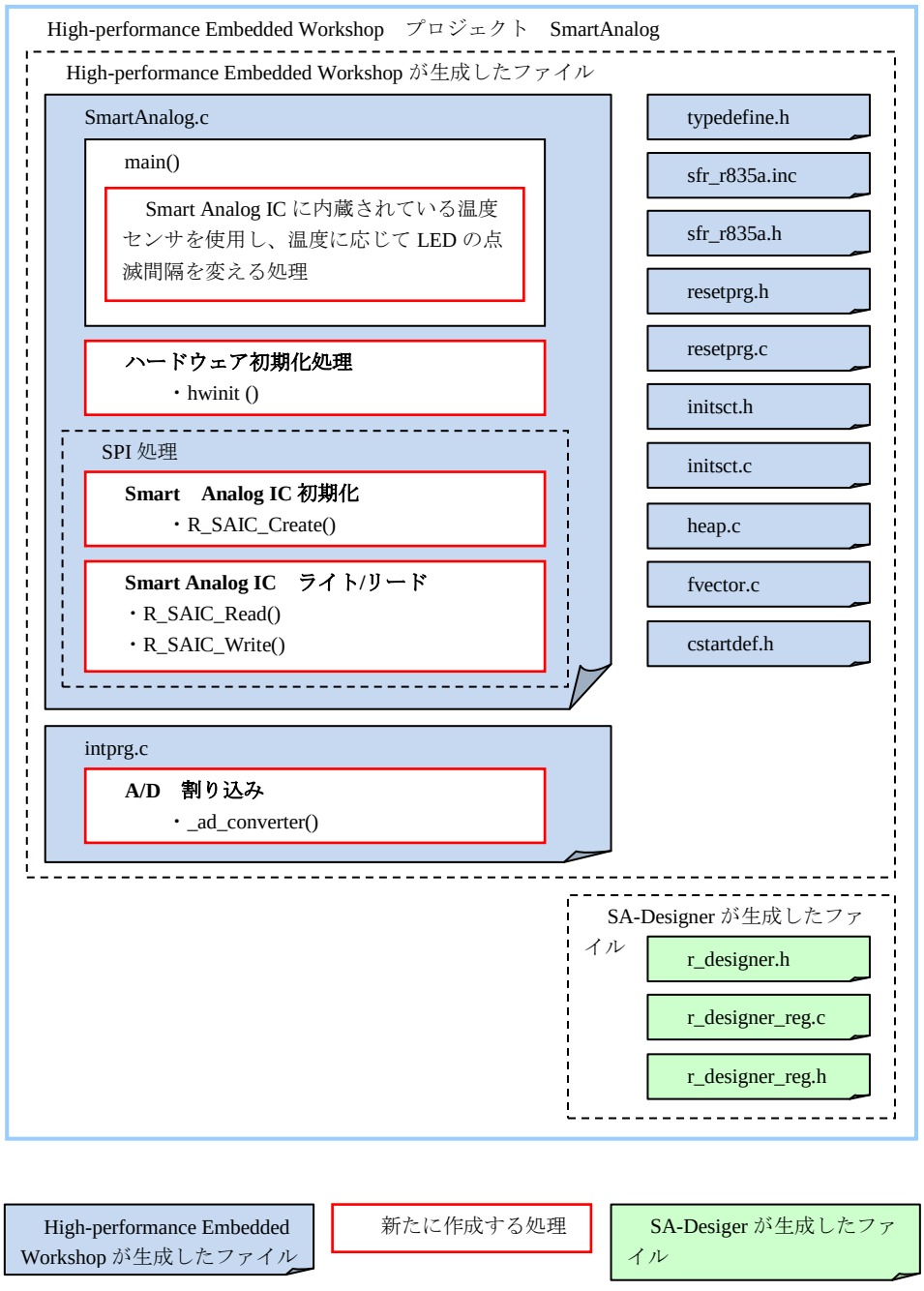
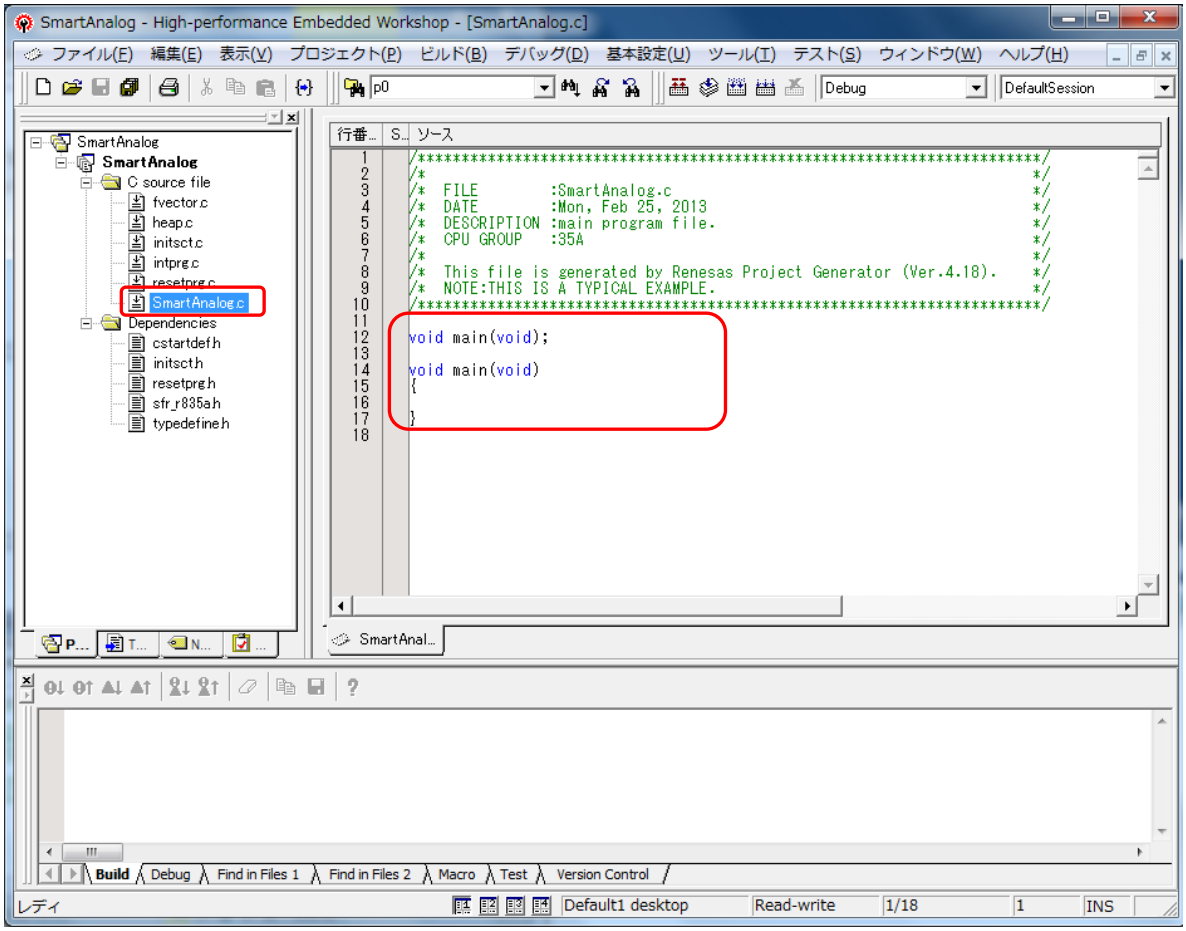

High-performance Embedded Workshop　プロジェクト　SmartAnalog
High-performance Embedded Workshop が生成したファイル
SmartAnalog.c
main()
Smart Analog IC に内蔵されている温度センサを使用し、温度に応じて LED の点滅間隔を変える処理
ハードウェア初期化処理
・hwinit ()
SPI 処理
Smart　Analog IC 初期化
・R_SAIC_Create()
Smart Analog IC　ライト/リード
・R_SAIC_Read()
・R_SAIC_Write()
intprg.c
A/D　割り込み
・_ad_converter()
typedefine.h
sfr_r835a.inc
sfr_r835a.h
resetprg.h
resetprg.c
initsct.h
initsct.c
heap.c
fvector.c
cstartdef.h
SA-Designer が生成したファイル
r_designer.h
r_designer_reg.c
r_designer_reg.h
High-performance Embedded Workshop が生成したファイル
新たに作成する処理
SA-Desiger が生成したファイル

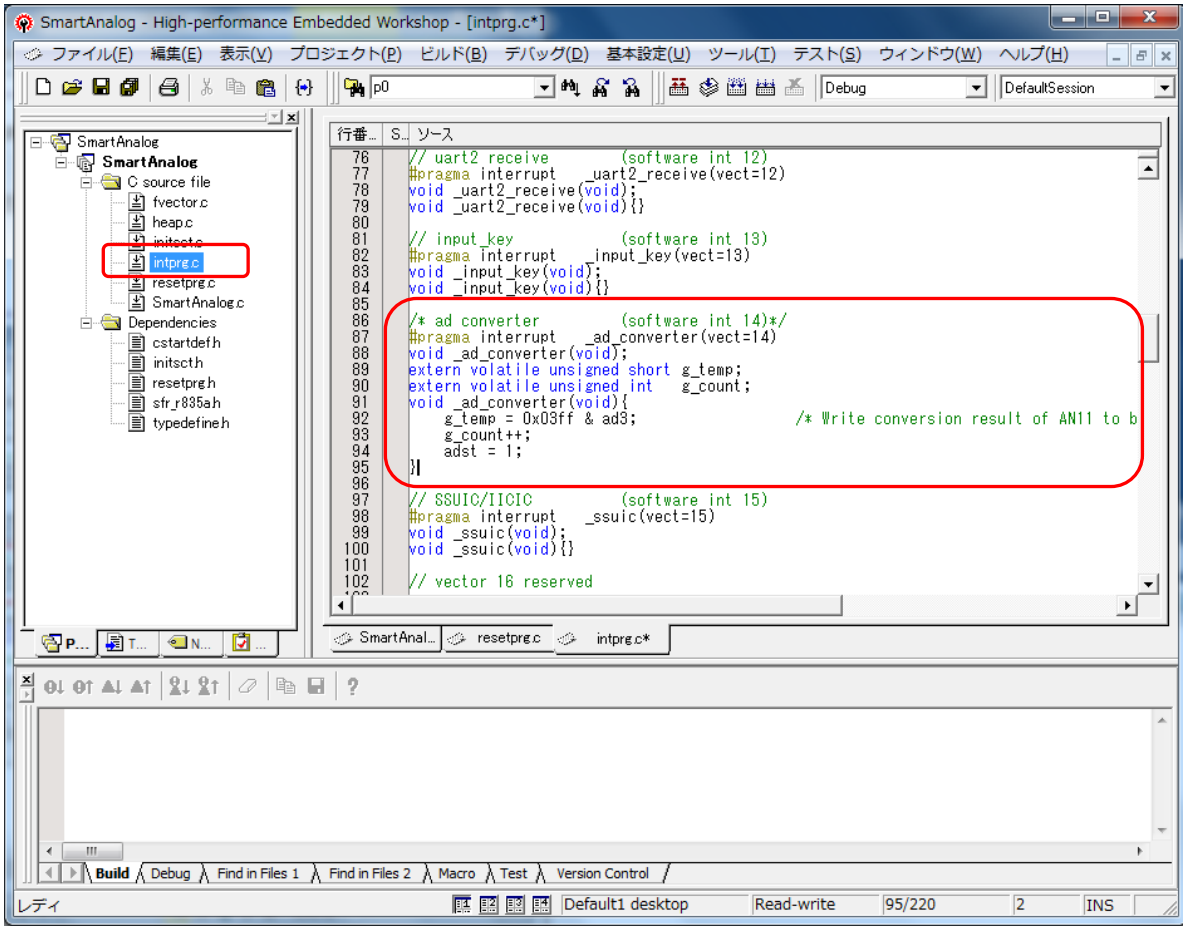
図 4.　システム構成図

ソースファイル SmartAnalog.c のメイン関数 main()に処理を追加します。また、初期化関数 hwinit()と SPI 関数を SmartAnalog.c に追加します。追加する処理については『3. サンプルプログラム』をご参照ください。



main()	[メイン関数] ・ハードウェア初期化関数(hwinit)を呼び出します。 ・Smart Analog IC の初期化(R_SAIC_Create)を行います。 ・A/D の値(温度変化)で LED の点滅間隔を変更します。
hwinit()	[ハードウェア初期化] ・ユーザのシステムに合わせ、ハードウェアの初期化(ポート、SSU、A/D の初期化、割り込みの許可)を行います。
R_SAIC_Create()	[Smart Analog IC 初期化] ・SA- Designer で生成されたレジスタデータを Smart Analog IC に転送します。
R_SAIC_Write()	[Smart Analog IC ライト] ・Smart Analog IC へ SPI 通信によりライトします。
R_SAIC_Read()	[Smart Analog IC リード] ・Smart Analog IC から SPI 通信によりリードします。 (このドキュメントでは使用しません。)

ソースファイル intprg.c の_ad_converter 割込み関数に処理を追加します。追加する処理については『3. サンプルプログラム』をご参照ください。



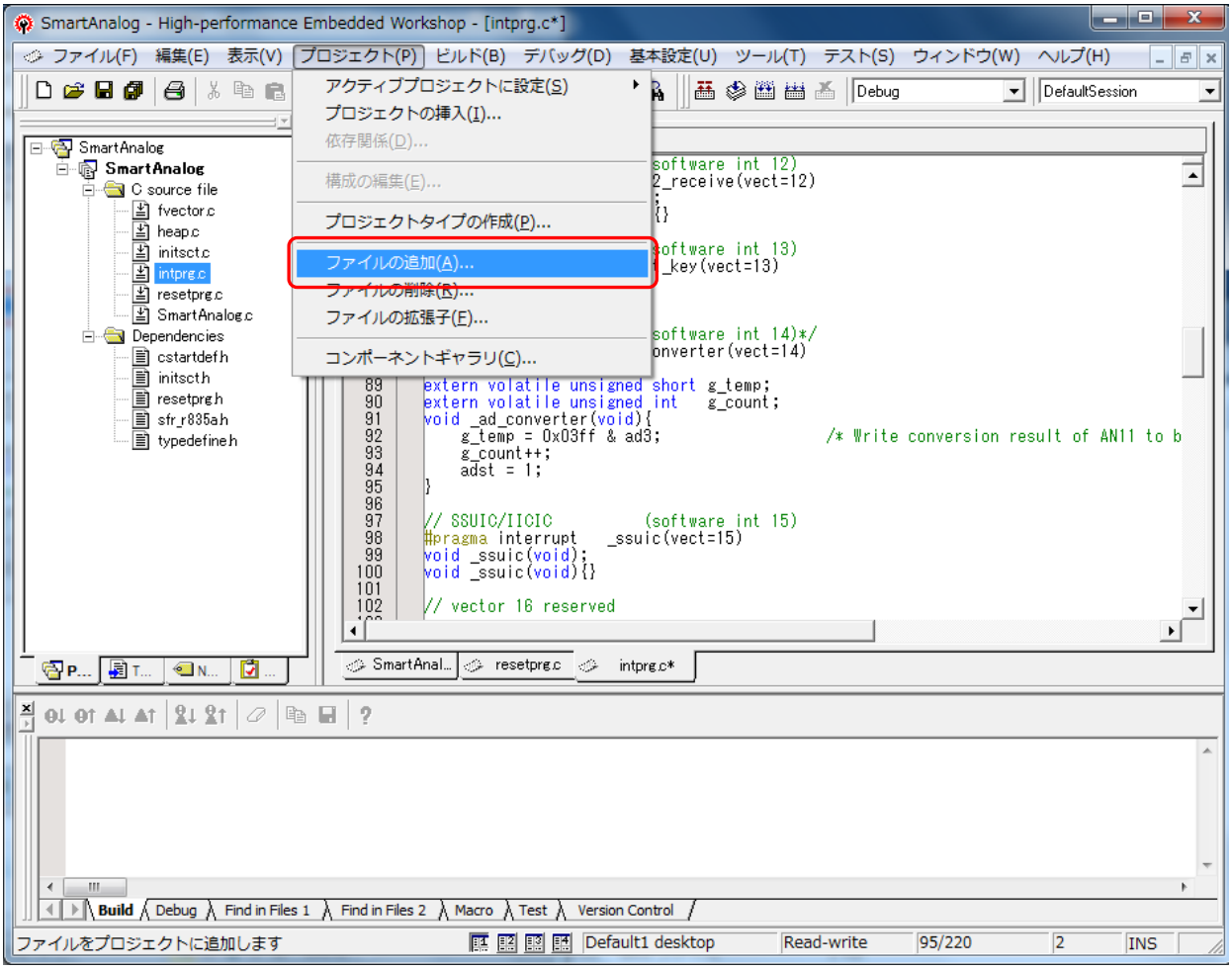
_ad_converter()	[A/D 割り込み関数] ・Smart Analog IC の温度センサからのデータを取得します。 ・割り込み回数をカウントします。 ・A/D 変換開始フラグ(A/D 変換開始)を設定します。

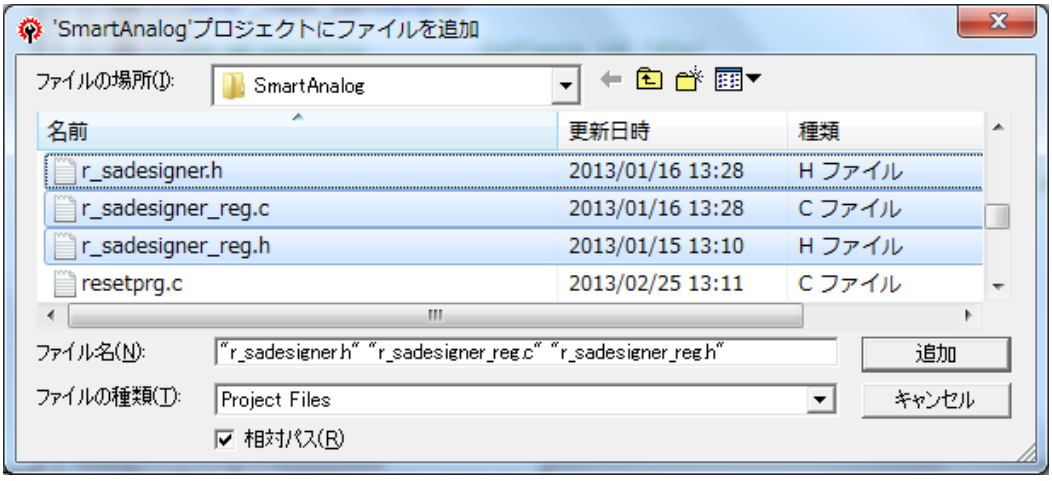
2.1.3　回路データの登録とビルド

(1)　ソースを High-performance Embedded Workshop に登録

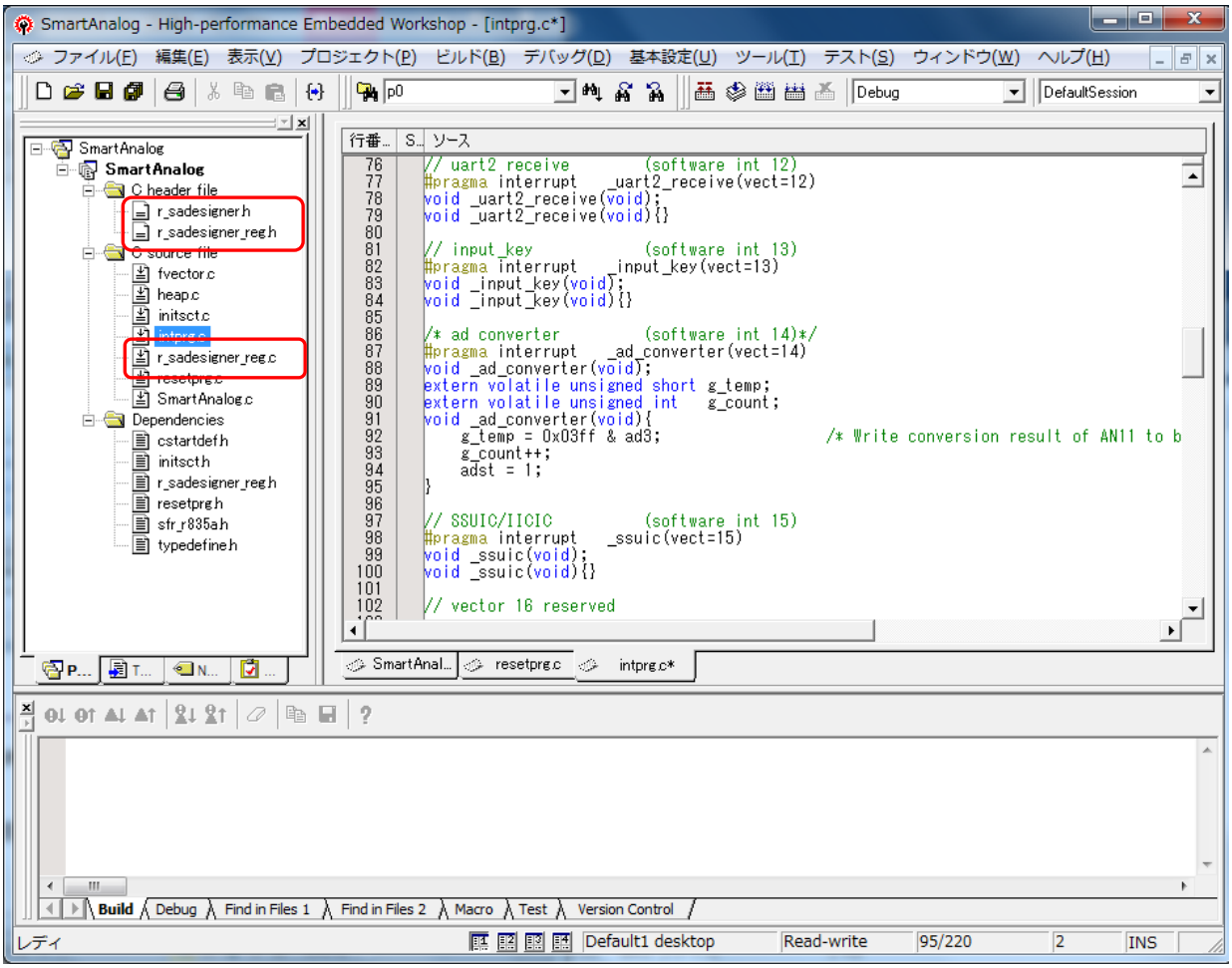
SA-Designer で生成したソースファイルを、High-performance Embedded Workshop で作成したプロジェクトに登録します。登録は [プロジェクト]-[ファイルの追加]で選択したソースファイルを追加します。

SA-Designer で生成したソースファイル(C:¥Smart_Analog)は事前に High-performance Embedded Workshop のワークスペースフォルダ(C:¥WorkSpace¥SmartAnalog¥SmartAnalog)へコピーします。





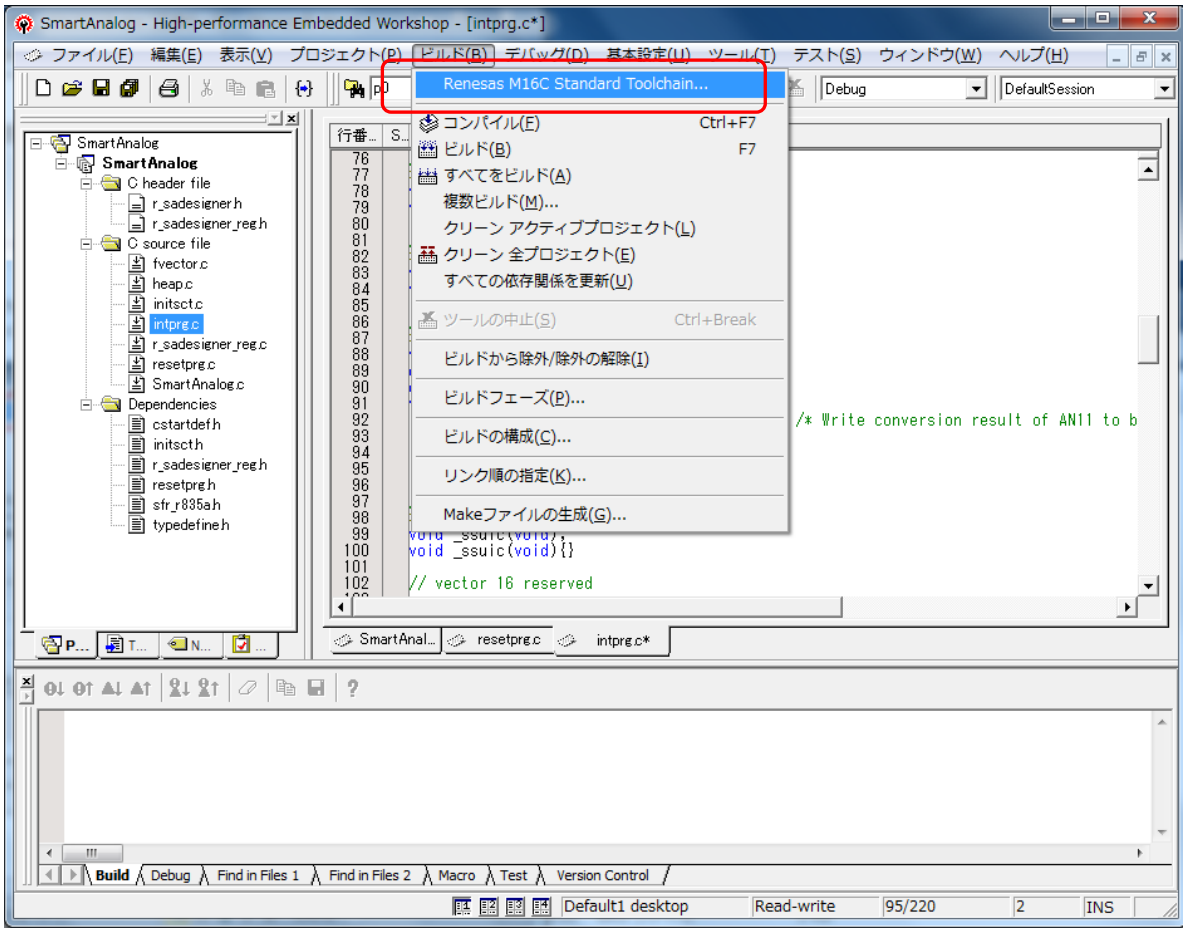
プロジェクト・ツリーに回路データのソースファイルが登録されます。

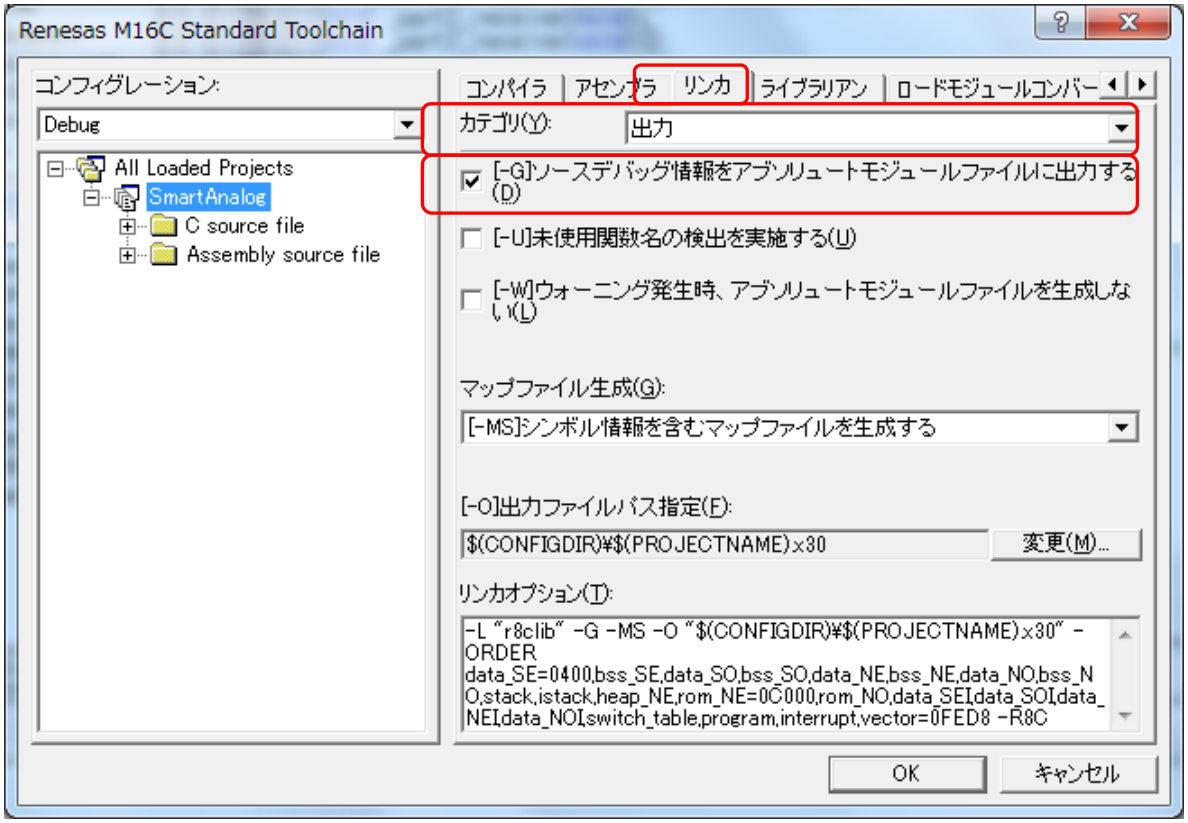


(2)　リンカオプションの設定

リンカオプションを変更し、デバッグ情報をモジュールファイルに出力するように設定します。

[ビルド]-[Renesas M16C Standard Toolchain...]をクリックします。



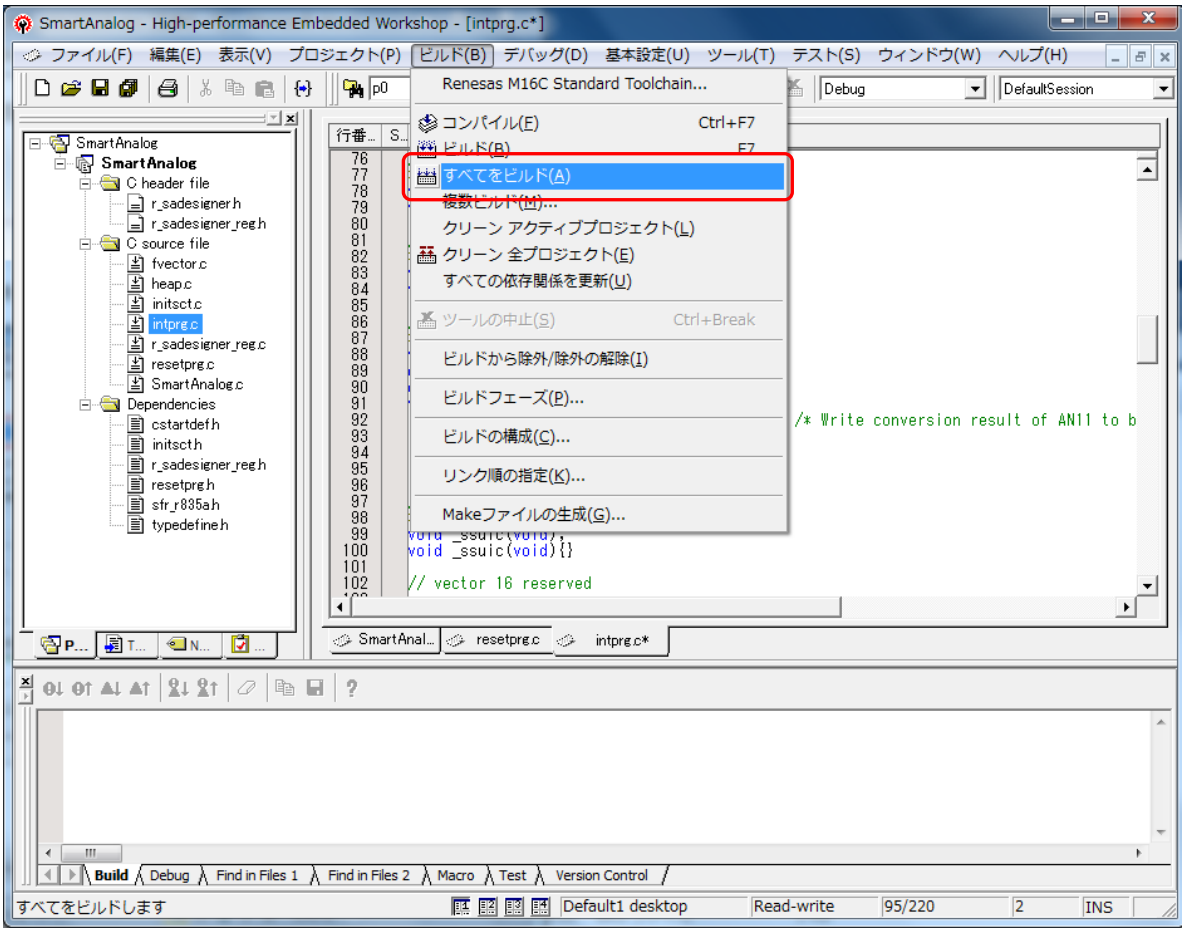


[リンカ]	“リンカ”を指定します。
[カテゴリ]	“出力”を選択します。
[-G]	”ソースデッバグ情報をアブソリュートモジュールに出力する”のチェックボックスを選択します。

[OK]ボタンを押し、設定を反映させます。

(3)　ビルド

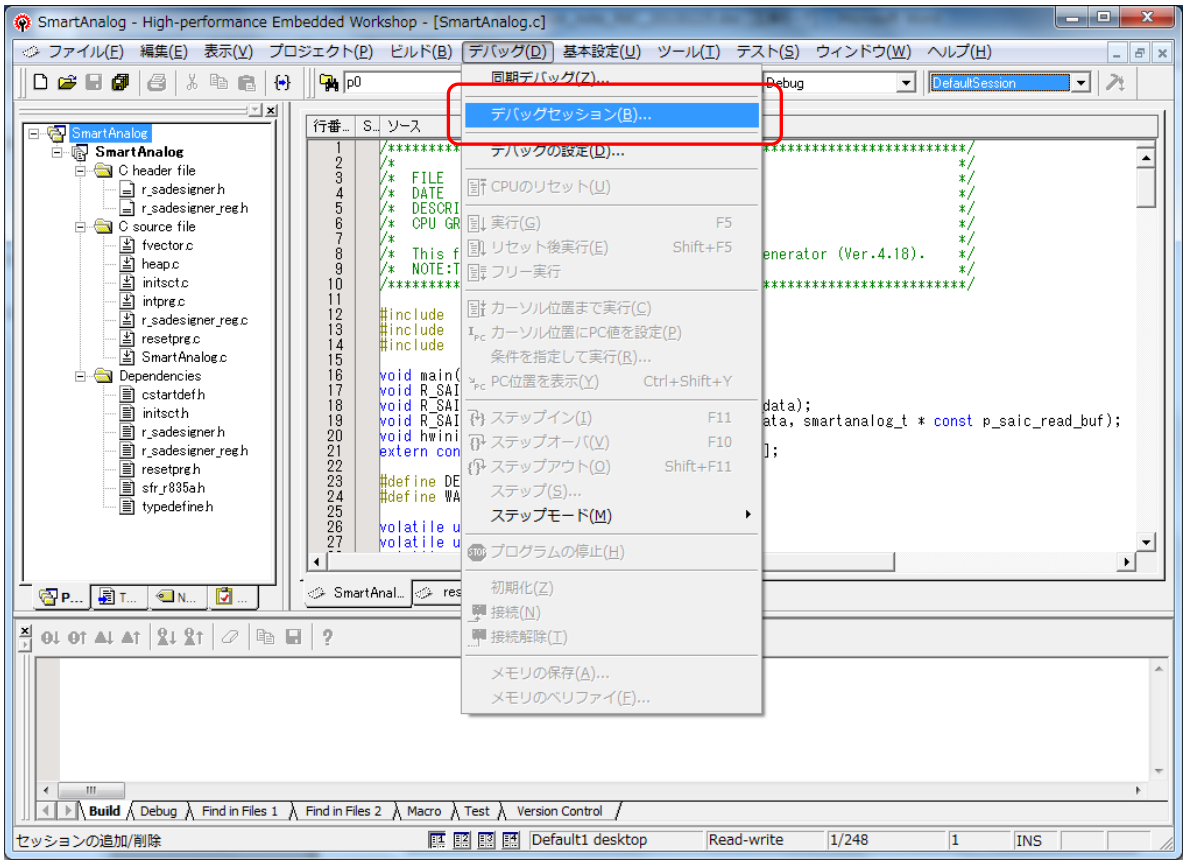
[すべてをビルド]でロードモジュールファイルを生成します。

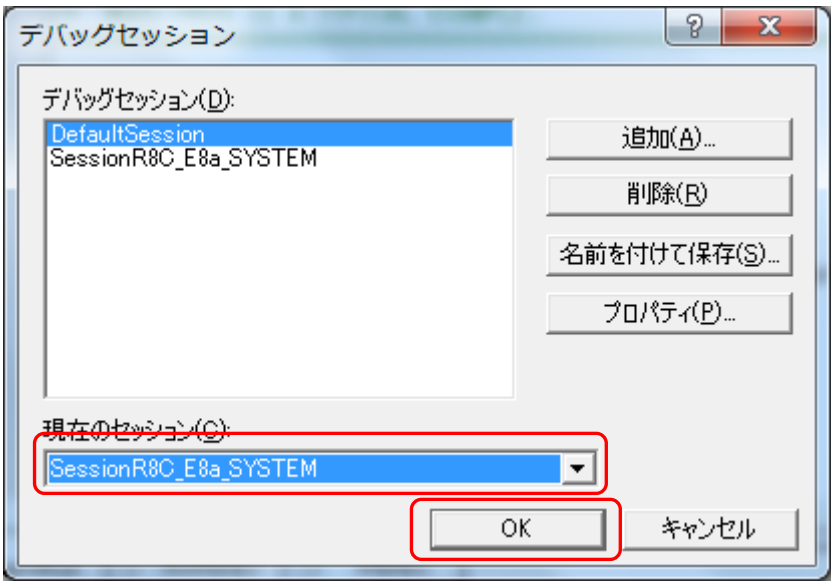


2.1.4　動作確認

(1)　ロードモジュールのダウンロード

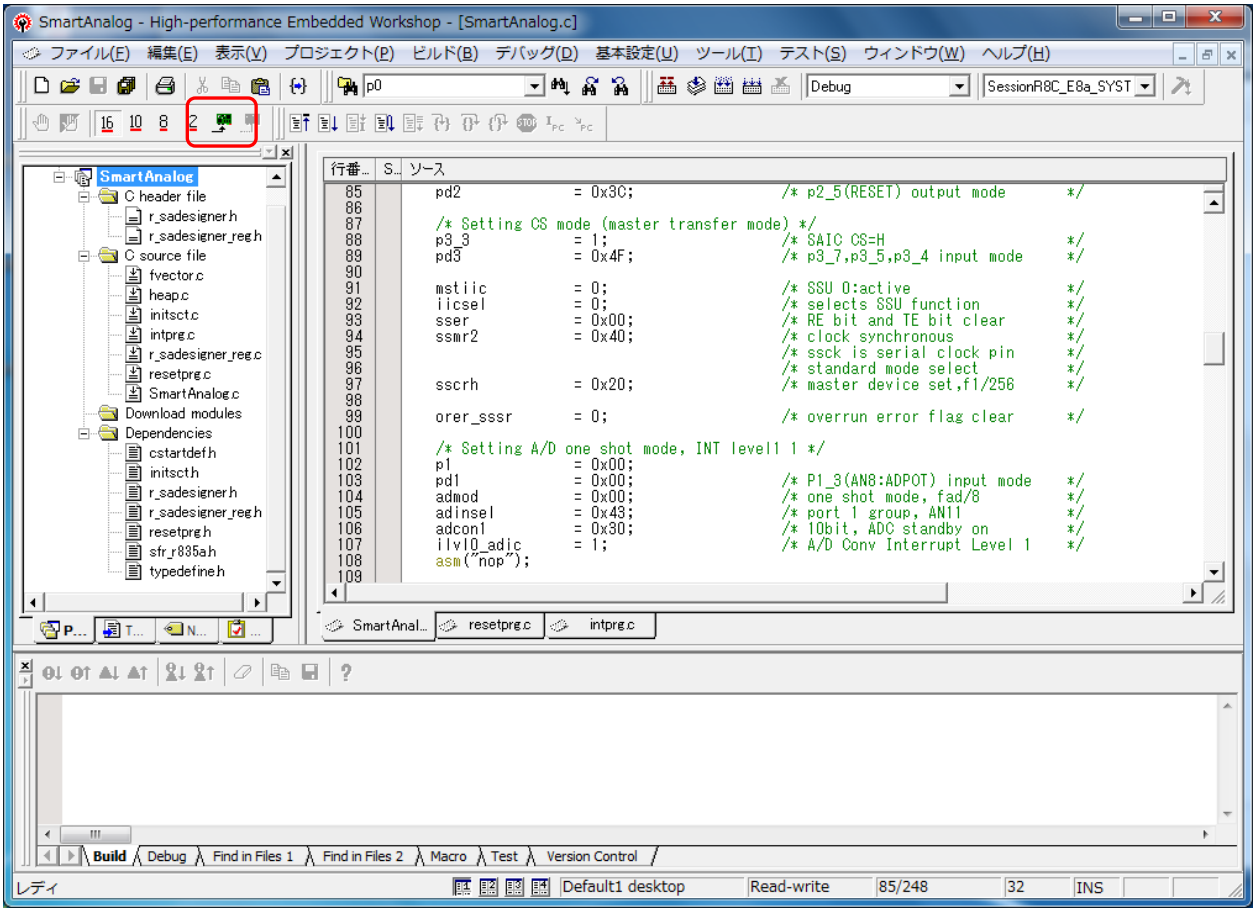
デバックセッションを選択します。



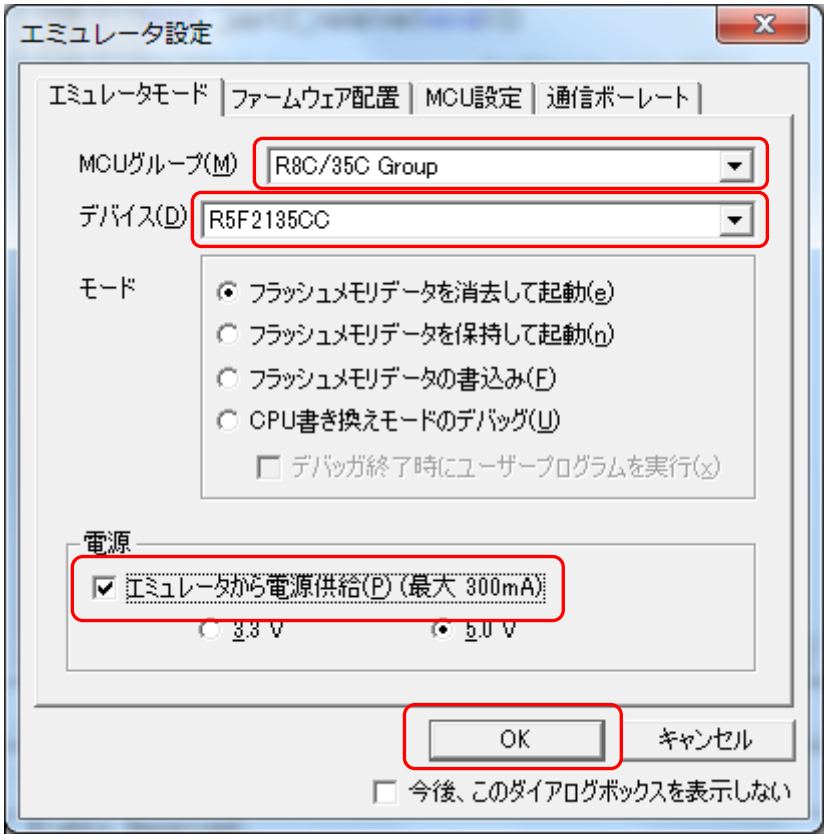


OK ボタンでエミュレータ設定ダイアログが表示されます。

既にデバッグセッションが選択されている場合、ツール・バーにある[接続]をクリックします。

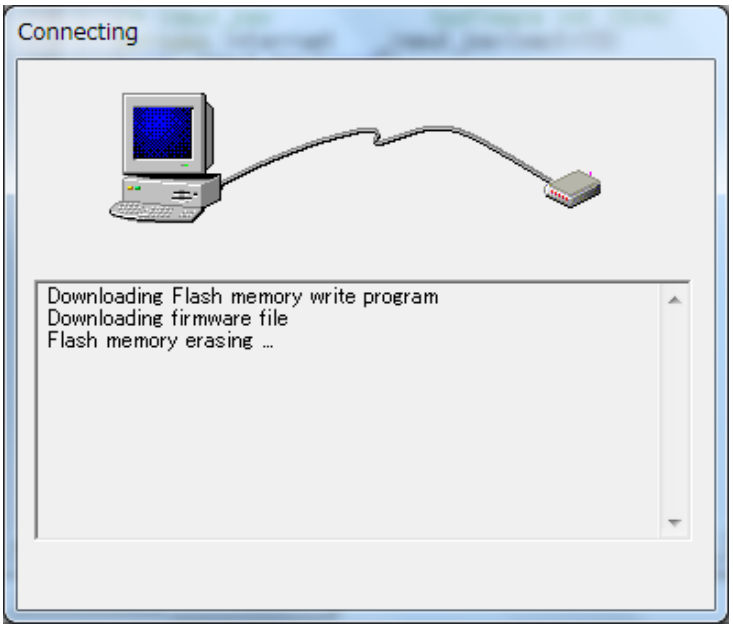


エミュレータ設定で、エミュレータから電源供給を設定します。

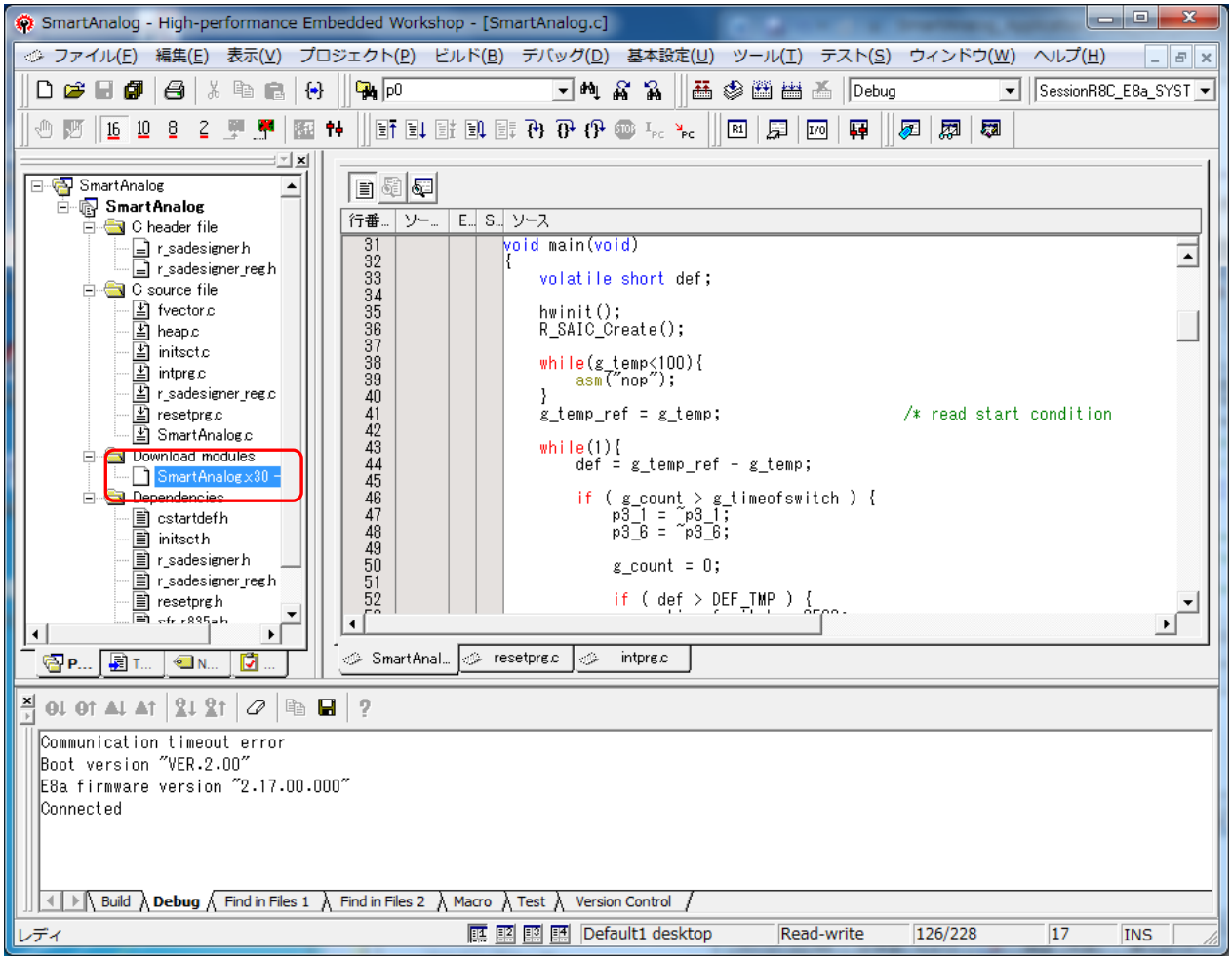


MCU グループ	“R8C/35C Group”を指定します。
デバイス	“R5F2135CC”を選択します。
[電源]ー[エミュレータから電源供給をする]	チェックボックスを選択します。

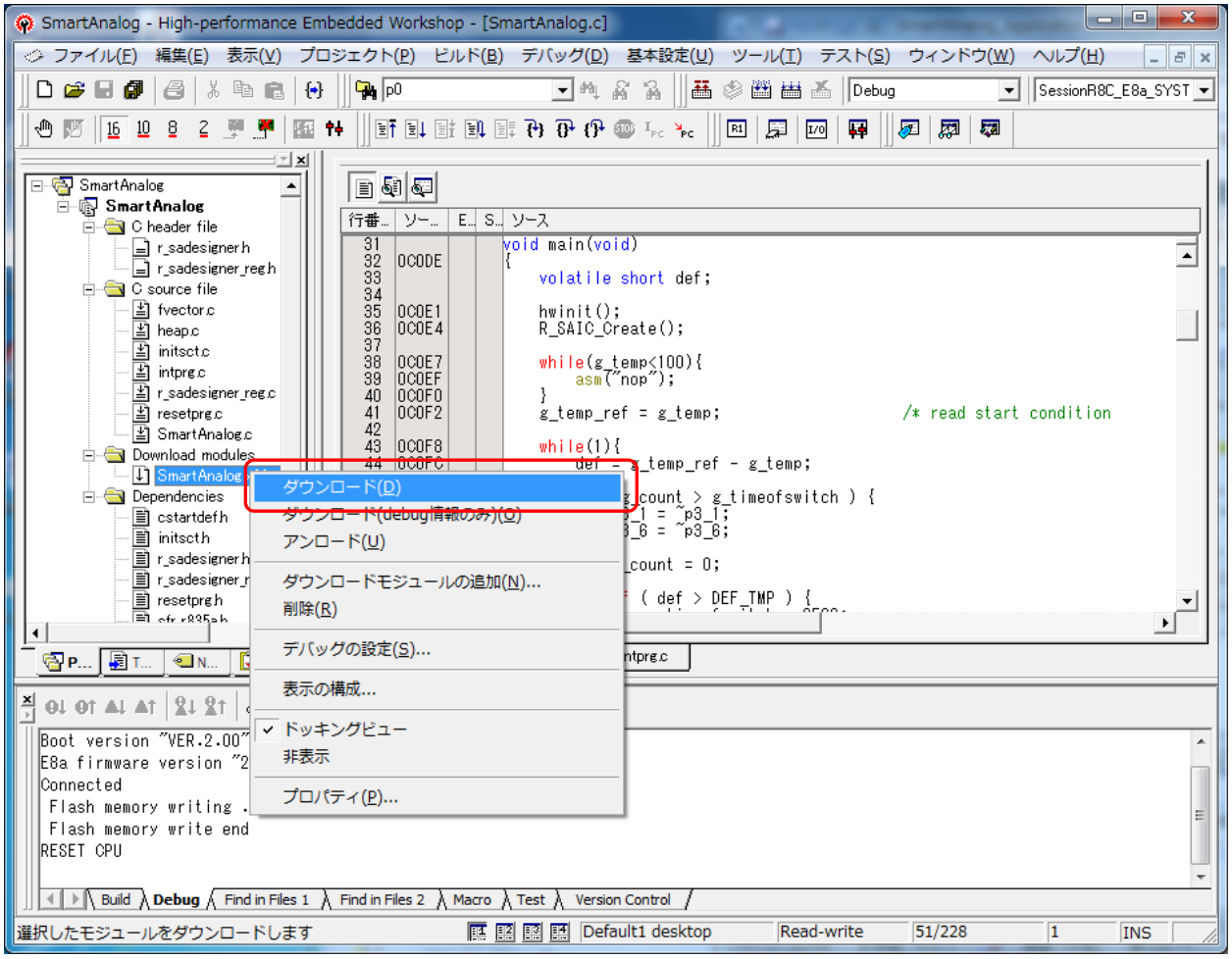
[OK]ボタンをクリックするとエミュレータに接続します。



接続が完了するとダウンロード可能なモジュール名が表示されます。



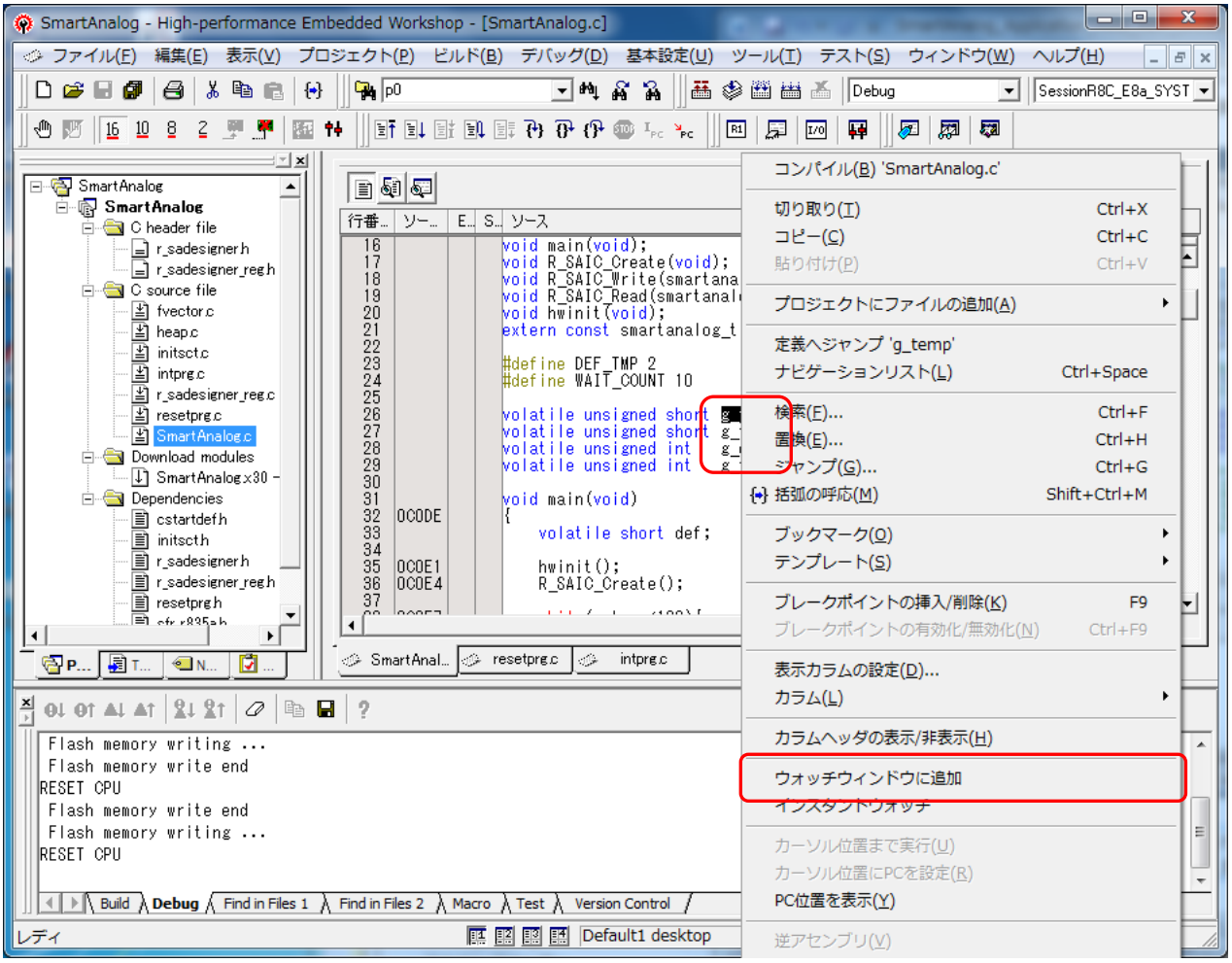
モジュール名を右クリックし、ダウンロードを選択します。



[ダウンロード]	“SmartAnalog.x30 - 00000000”を選択します。

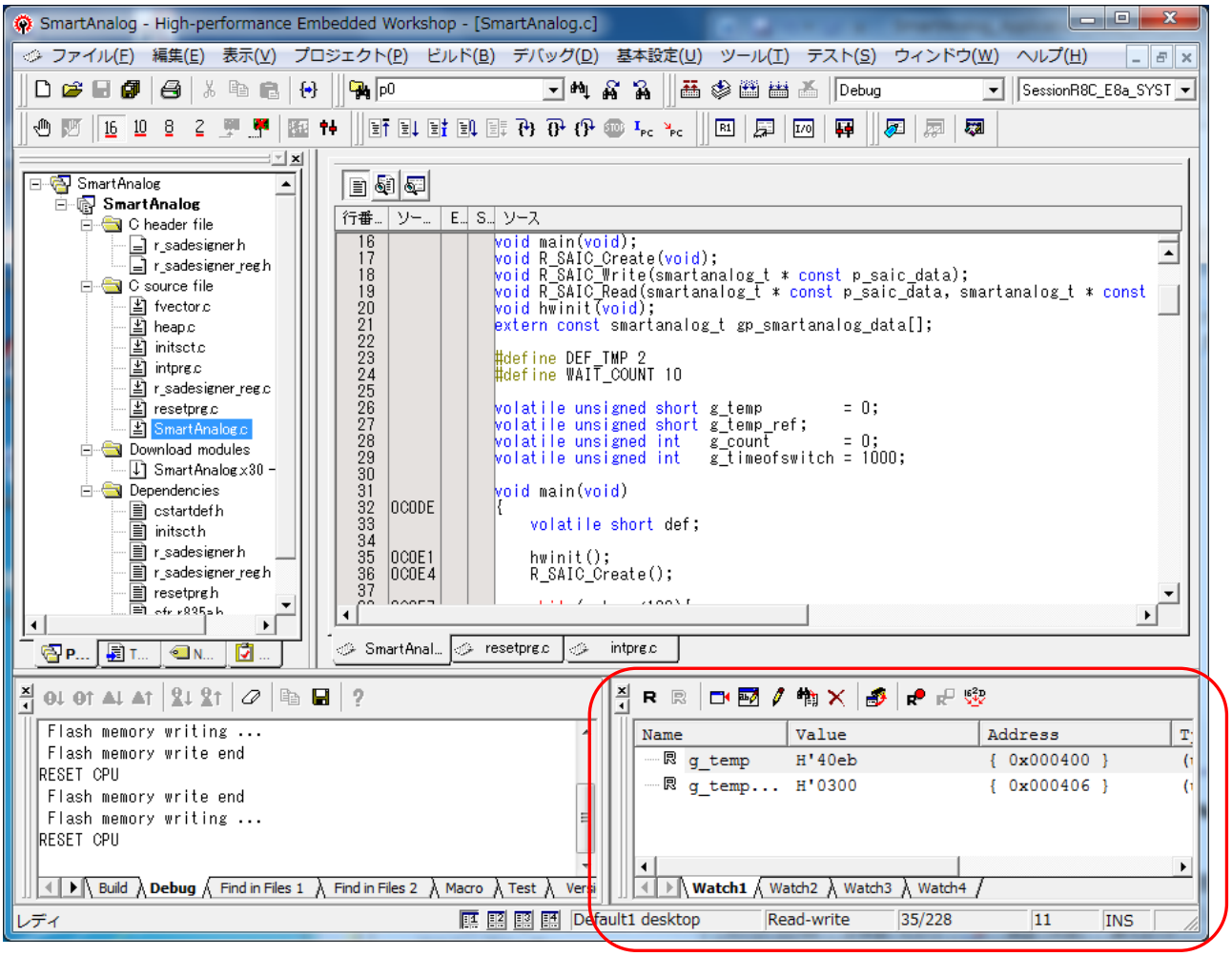
(2)　変数のウォッチ登録

プログラムの動作を確認するために、変数のウォッチ登録を行います。



SmartAnalog.c に記述されている変数 g_temp にカーソルをあてて右クリックし、[ウォッチウインドウに追加]をクリックします。同様に g_temp_ref も登録します。

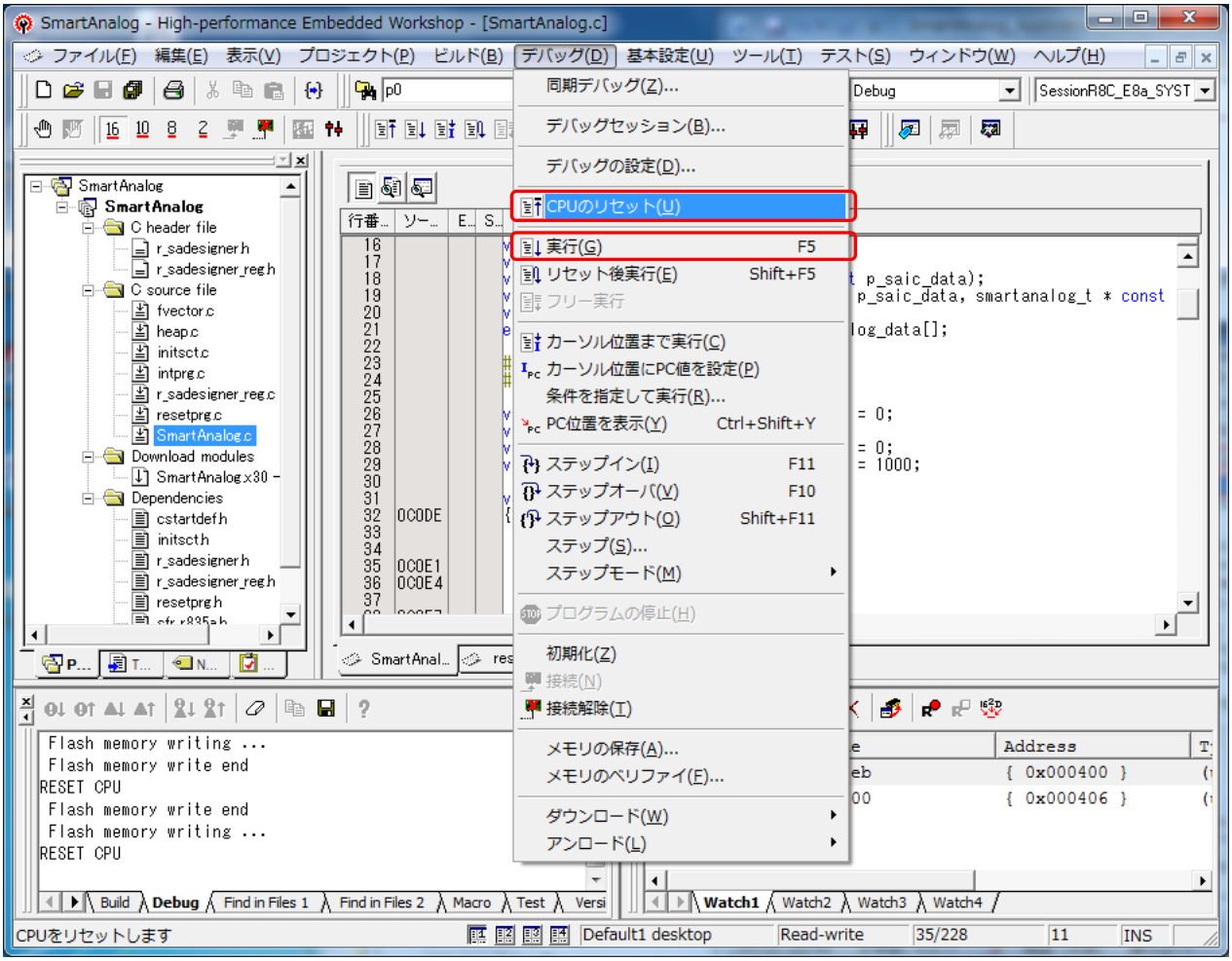
[Watch1]に登録した変数が表示されます。



(3)　プログラムの実行

システムの動作確認を行います。

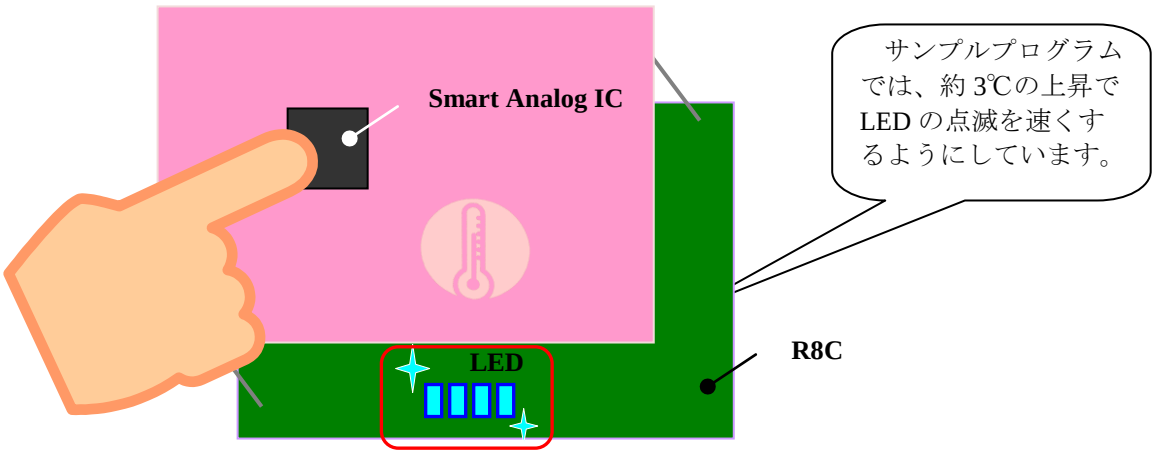
プログラムの実行前に必ず[CPU リセット]を行ってください。



[デバッグ]から[CPU リセット]を行い、[実行]でプログラムを実行します。

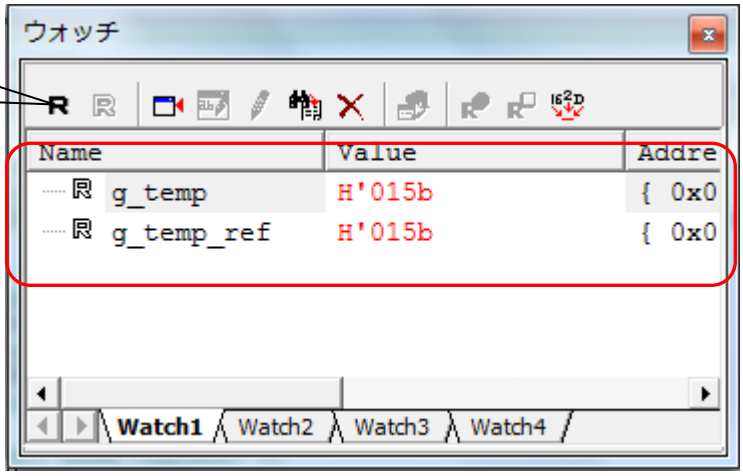
Smart Analog IC500 のマイコンに指をあててみてください。

指をあてるとマイコンの温度が上昇し、A/D 変換値の変数 g_temp の値が小さくなります。また、温度が上昇すると LED の点滅間隔が小さくなります。



温度センサの特性

-5mV/℃に従って、A/D 変換値である変数 g_temp の値が小さくなります。



3.　サンプルプログラム

本アプリケーションノートで使用しているサンプルプログラムを以下に示します。

(1)　メイン関数（SmartAnalog.c のメイン関数 main()に追加）

```
#include    "sfr_r835a.h"
#include    "r_sadesigner_reg.h"
#include    "r_sadesigner.h"

void main(void);
void R_SAIC_Create(void);
void R_SAIC_Write(smartanalog_t * const p_saic_data);
void R_SAIC_Read(smartanalog_t * const p_saic_data, smartanalog_t * const p_saic_read_buf);
void hwinit(void);
extern const smartanalog_t gp_smartanalog_data[];

#define DEF_TMP 2
#define WAIT_COUNT 10

volatile unsigned short g_temp         = 0;
volatile unsigned short g_temp_ref;
volatile unsigned int   g_count        = 0;
volatile unsigned int   g_timeofswitch = 1000;

void main(void)
{
    volatile short def;

    hwinit();
    R_SAIC_Create();

    while(g_temp<100){
        asm("nop");
    }
    g_temp_ref = g_temp;                    /* read start condition         */
```

(2)　初期化関数（SmartAnalog.c に追加）

```
void hwinit(void)
{
    unsigned char osc_stab = 128;

    _asm(" FCLR I");                    /* Disable interrupts */
    prc0 = 1;                           /* Protection off */
    cm13 = 1;                           /* Pin P4_6 and P4_7 are
                                           configured as XIN and XOUT */
    cm05 = 0;                           /* XIN clock oscillates */
    cm06 = 0;                           /* CM16 and CM17 enable */
    cm16 = 0;                           /* Main clock = No division mode */
    cm17 = 0;

    while (osc_stab)                    /* Wait till the oscillator stabilizes */
    {
        --osc_stab;
    }

    ocd2 = 0;                           /* Select XIN clock */
    prc0 = 0;                           /* Protection on */


    p2              = 0x00;
    p2_5            = 1;                /* Smart Analog IC Reset=H      */
    pd2             = 0x3C;             /* p2_5(RESET) output mode      */

    /* Setting CS mode (master transfer mode) */
    p3_3            = 1;                /* SAIC CS=H                    */
    pd3             = 0x4F;             /* p3_7,p3_5,p3_4 input mode    */

    mstiic          = 0;                /* SSU 0:active                 */
    iicsel          = 0;                /* selects SSU function         */
    sser            = 0x00;             /* RE bit and TE bit clear      */
    ssmr2           = 0x40;             /* clock synchronous            */
                                        /* ssck is serial clock pin     */
                                        /* standard mode select         */
```

(3)　割込み関数（intprg.c に追加）

```
extern volatile unsigned short g_temp;
extern volatile unsigned int   g_count;
void _ad_converter(void){
  g_temp = 0x03ff & ad3;                    /* Write conversion result of AN11 to buffer */
    g_count++;
```

(4)　SPI 関数（SmartAnalog.c に追加）

(a)　R_SAIC_Create()

```
/***********************************************************************************/
/*  R_SAIC_Create();                                                               */
/***********************************************************************************/
void R_SAIC_Create(void)
{
    volatile uint16_t w_count;

    p2_5 = 0;                               /* Analog IC Reset                */

    /* Change the waiting time according to the system */
    for(w_count = 0U; w_count < 10; w_count++)
    {
        asm("nop");
```

(b)　R_SAIC_Write()

```
/*****************************************************************************/
/*  R_SAIC_Write(gp_smartanalog_data);                                       */
/*****************************************************************************/
void R_SAIC_Write(smartanalog_t * const p_saic_data)
{
    volatile uint8_t adrs;
    volatile uint8_t dat;
    volatile uint8_t wait = 0;

    smartanalog_t *p_saic_write;
    p_saic_write = p_saic_data;
    te_sser = 1;                            /* transmit enable              */

    while(p_saic_write->address != 0xff) {

        p3_3 = 0;                           /* SAIC CS                      */
        for(wait = 0; wait < 10; wait ++)   /* SA Stable waiting time (tSKA) */
        {
            asm("nop");
        }

        while(tdre_sssr != 1);              /* wait transmit data empty     */
        adrs = (p_saic_write->address & 0x7F) | 0x80; /* 0x80 data write mode*/
        sstdr = adrs;                       /* send SAIC Address data       */

        while(tdre_sssr != 1);              /* wait transmit data empty     */

        dat = p_saic_write->data;
        sstdr = dat;
```

(c)　R_SAIC_Read()

```
/*****************************************************************************/
/*  R_SAIC_Read(gp_smartanalog_data, saic_read_buf);                         */
/*****************************************************************************/
void R_SAIC_Read(smartanalog_t * const p_saic_data, smartanalog_t * const p_saic_read_buf)
{
    volatile uint8_t adrs;
    volatile uint8_t wait;
    smartanalog_t *p_saic_write;
    smartanalog_t *p_saic_read;
    int dummy_read;

    p_saic_write = p_saic_data;
    p_saic_read  = p_saic_read_buf;

    sser = 0;
    sser = 0x18;                          /* transmit, receive enable      */

    while(p_saic_write->address != 0xff) {

        p3_3 = 0;                         /* SAIC CS                       */
        for(wait = 0; wait < 10; wait ++)   /* SA Stable waiting time (tSKA) */
        {
            asm("nop");
        }

        while(tdre_sssr != 1);            /* wait transmit data empty      */
        adrs = p_saic_write->address & 0x7F;
        p_saic_read->address = adrs;
        sstdr = adrs;                     /* send SAIC Address data        */

        while(tdre_sssr != 1);            /* wait transmit data empty      */
        while(tend_sssr != 1);            /* wait receive data             */
        dummy_read = ssrdr;

        sstdr = 0xff;                     /* send dummy data               */

        while(rdrf_sssr != 1);            /* wait receive data             */
        dummy_read = ssrdr;
```

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
1.00	2013.03.25	—	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、本文を参照してください。なお、本マニュアルの本文と異なる記載がある場合は、本文の記載が優先するものとします。

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. リザーブアドレスのアクセス禁止

   【注意】リザーブアドレス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、事前に問題ないことをご確認下さい。

   同じグループのマイコンでも型名が違うと、内部メモリ、レイアウトパターンの相違などにより、特性が異なる場合があります。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。

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・電話番号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 エレクトロニクス販売株式会社　〒100-0004 千代田区大手町 2-6-2（日本ビル） (03)5201-5307

■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/