

## Smart Analog

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。

SA-Designerを使ったシステム開発手順(R8Cファミリ編)

- ・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	2
1.1	開発環境	3
1.1.1	ハードウェア	3
1.1.2	ソフトウェア	3
2.	開発手順	4
2.1	概要	4
2.1.1	アナログフロントエンド回路の設計	5
(1)	SA-Designer の起動	5
(2)	新しい回路の設計	5
(3)	回路図の作成	7
(4)	ソースファイルの生成	8
2.1.2	プログラムの作成	9
(1)	High-performance Embedded Workshop の起動	9
(2)	プロジェクト新規作成	0
(3)	プログラムの作成1	7
2.1.3	回路データの登録とビルド2	22
(1)	ソースを High-performance Embedded Workshop に登録2	22
(2)	リンカオプションの設定2	24
(3)	ビルド	26
2.1.4	動作確認2	27
(1)	ロードモジュールのダウンロード2	27
(2)	変数のウォッチ登録	32
(3)	プログラムの実行	34
3.	サンプルプログラム	36
(1)	メイン関数(SmartAnalog.c のメイン関数 main()に追加)	36
(2)	初期化関数(SmartAnalog.c に追加)	37
(3)	割込み関数(intprg.c に追加)	38
(4)	SPI 関数(SmartAnalog.c に追加)	8



#### 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。



※ 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

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

#### (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。

⊵ (無題)(更新)	- SA-Desi	gner			
ファイル( <u>E</u> )	表示( <u>∨</u> )	<u>生成(G)</u>	ツール(I)	ヘルプ( <u>H</u> )	
ネスタート	3 🖃 🚯		🖆 💽 I 🚳	🕜   😐   RAA730500Z (Sma	art Analog IC500)

- SA-Designerについて理	e解する Separa		
GO SA-Desi Ju-Ky	ignerで何ができるか、チュートリアル アルには、SA-Designerを有効活用	を一読することをお薦めします。 月するための情報が記載されてい	ます。
- Currient 新しい回路を設計する	wrone - Amplifier		
利しい回転を設計する			
GO 新たにア:	ナログ・フロント・エンド回路を作成し	ます。 いっつい ヨー・	
	Custom		
Amelifier Ch2	MPCNUT -O		
● 既存の回路を開く		O AMPE OUT	
			28 0 DAGS
GO 既(生成 Benesas	就済みのAFEデザイン・ファイルを読み s VA(Virtual AFE Designer)の出	り込みます。 カファイルを読み込むこともできま	a.
		,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	
2 O-44	AMP2_CUT		
□ おかけにまテレカ(いの)			問じる
			14100

[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。

新規作成
デバイス選択 デバイス( <u>D</u> ): RAA730500Z (Smart Analog IC500) ・
ソース生成先 フォルダ名(E): C¥Smart_Analog 参照(B)
-ライブラリ SPI関数を生成する( <u>S</u> )
OK キャンセル

[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]-[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。

💫 (無題)(更新) - SA-Desig	iner			<b>x</b>
ファイル( <u>E</u> ) 表示( <u>V</u> )	生成	ἶ(G) ツール(T) ヘルプ(H)		
🔊 スタート   🛃 🔚   🛐	C	ソース生成(G)	Ctrl+G	nart Analog IC500)
	C	生成->登録(R)	Ctrl+R	
	0	生成->登録->ビルド->実行(B)	Ctrl+B	
		生成オプション(0)		_
SA-I	Design	er 情報(I00001)		
	i	AFEレジスタ・データ ソース・ファイルの	D生成が完了しまし	った。
				ок

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。

			l	<u> </u>
	ディスク (C:) 🕨 Smart_Analog	✓ 4 Smart_Al	nalogの検索	م
整理 ▼ ライブラリに追	加▼ 共有▼ 書き込む 新	fしいフォルダー	: : :	• 🔳 🔞
☆ お気に入り	名前	更新日時	種類	サイズ
	r_sadesigner.h	2013/01/16 18:19	╫᠊ <del>╤╶</del> ┽╝	3
🍃 ライブラリ	r_sadesigner_reg.c	2013/01/16 18:19 2013/01/16 18:19	C ファイル H ファイル	4
▶ コンピューター				
	•	m		•
3個の項目				



## 2.1.2 プログラムの作成

#### (1) High-performance Embedded Workshop の起動

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

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。

🛞 High-performance Embedd	ed Workshop					
ファイル(F) 編集(E) 表示(V)	プロジェクト(P) ビルド(B)	デバッグ(D) 基本設定(U)	ツール(T) テスト(S)	ウィンドウ(W)	ヘルプ(H)	
▋□☞■₽ ₽ ⊁▫	• 🖻   🕀    🗛 📃	▲ 64 號 器	₩ @ ₩ ₩ ¥  [		•	•
	ようこそ!	-ウスペースの作成(C) ジェクトワークスペースを聞く(O): ウスペースを参照する(B)		? × ○K >センレル レーション(A)		A
VIII VIII VIII VIII VIII VIII VIII VII	in Files 1 $\lambda$ Find in Files 2 $\lambda$ Macro	→ 入 Test 入 Version Control /				TING



## (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。

ようこそ	1	? ×
2	● 新規プロジェクトワークスペースの作成℃	ОК
	○ 最近使用したプロジェクトワークスペースを開く(@):	キャンセル
	○ 別のプロジェクトワークスペースを参照する(B)	
-	[新規プロジェクトワークスペースの作成]の[ <b>OK</b> ]をクリ	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。

新規プロジェクトワークスペース		? ×
プロジェクトタイプ プロジェクトタイプ Application C source startup Application Empty Application Import Makefile Library RSKR8C35C Debugger only - M16C E8a SYST Debugger only - M16C Simulator Debugger only - R8C E8a SYSTE	ワークスペース名(W): SmartAnalog プロジェクト名(P): SmartAnalog ディレクトリ(D): C:¥WorkSpace¥SmartAnalog CPU種別(©): M16C マールチェイン(T): Renesas M16C Standard ▼	参照(B)
	ОК	キャンセル

[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。



新規プロジェクト-1/6-CPU	? <mark>×</mark>
	ツールチェインバージョン:         5.45.01         このプロジェクトで使うCPUのシリーズとグループを選択して下さい。         CPUシリーズ:         M16C/30         M16C/20         M16C/10         M16C/Tiny         R8C/Tiny         CPUグループ:         32A         35A         L3AA         Other         東沢したいCPUグループまたは"Other"を選択してください。
< 戻る( <u>B</u> )	次へ(N)> 完了 キャンセル

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。

新規プロジェクトー2/5-生成ファイル	? <mark>×</mark>
	自動生成するイニシャルルーチンを選択します。
	ROMサイズ:: 4K レ/Oライフ*ラリ使用 (UART1) ビーフ*メモリ使用 ヒーフ*サイス: 0×80 Main 関数生成 C source file オンチッフ*エミュレータ使用
	ファームウェアアト*レス: 0x     ▼     サイス*: 0x       ワークRAMアト*レス: 0x     ▼     サイス*: 0x
< 戻る(日	) ( ) ( ) ( ) ( ) ( ) ( ) ( ) ( ) ( ) (

新規プロジェクトー3/5ースタック領域	? <mark>×</mark>
	スタックの設定を行って下さい。 - ユーザ <sup>*</sup> スタックホ <sup>*</sup> インタ: - マーザ <sup>*</sup> スタック使用 スタックサイス <sup>*</sup> : - ①×80 - 割り込み・スタックホ <sup>*</sup> インタ: - スタックサイス <sup>*</sup> : - ②×80
< 戻る( <u>B</u> )	<u> 流へ(N) テ</u> 完了 キャンセル

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。

新規プロジェクトー4/5ーデバッガ	? ×
	ターケ <sup>5</sup> ット: □M16C R8C Simulator ▼R8C E8a SYSTEM
	外部デンバッ力: none
	ターケ <sup>*</sup> ットタイフ <sup>*</sup> : R8C/Tiny ターケ <sup>*</sup> ットCPU: All CPUs マ
< 戻る( <u>B</u> )	次へ(N) > 完了 キャンセル



デバッガのオプションを設定します。

新規プロジェクトー5/6ーデバッガオプション	? ×
	ターケット名: R8C E8a SYSTEM コア: <pre></pre> コンフィグレーション名: Debug_R8C_E8a_SYSTEM  詳細オフ <sup>*</sup> ション:  Item Setting  © 初期セッション
< 戻る( <u>B</u> )	<u> </u>

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。

新規プロジェクト-6/6-生成ファイル名	? 💌
	以下のソースファイルを生成します: <u>ファイル名 拡 解説</u> typedefine h define scalar types. resetprg c initialize for C language. resetprg h include some headder files. initsct c initialize each sections. initsct h define the macro for initialit fvector c define the fixed vector table intprg c define the top address of th sfr_r835a h define the sfr register. (for sfr_r835a inc define the sfr register. (for heap c define the size of heap. SmartAn c main program file. cstartdef h define the size of stack.
〈 戻る(B)	次へ(N) > 完了 キャンセル



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。

概要	?	x	
プロジェクトの概要:			
PROJECT GENERATOR			
PROJECT NAME : SmartAnalog PROJECT DIRECTORY : C:¥WorkSpace¥SmartAnalog¥S CPU SERIES : R8C/Tiny	martA		
CPU GROUP: 35A TOOLCHAIN NAME: Renesas M16C Standard Toolo TOOLCHAIN VERSION: 5.45.01	:hain	=	
GENERATION FILES : C:WorkSpace¥SmartAnalog¥SmartAnalog¥typedefineh			
C:¥WorkSpace¥SmartAnalog¥SmartAnalog¥resetprg.c			
C:\WorkSpace\SmartAnalog\SmartAnalog\resetprg.h			
Ci¥WorkSpace¥SmartAnalog¥SmartAnalog¥initsct.c initialize each sections.			
C#WorkSpace#SmartAnalog#SmartAnalog#initscth define the macro for initialization of sections.		-	
< III	•		
OKをクリックしフロジェクトを作るか、Cancelをクリックしアホートするかを選択します。			
▼ サマリの内容をプロジェクトディレクトリにReadme.txtという名前で保存す	<b>る</b> 。		
	ancel		



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。

SmartAnalog - High-performance Embedded Workshop	
ファイル(E) 編集(E) 表示(V) プロジェクト(P) ビルド(B) デバッグ(D) 基本設定(U) ツール(I) テスト(S) ウィンドウ(W)	ヘルプ( <u>H</u> )
🔢 🗅 💣 🖬 🕼 🕼 🕼 🕼 🔛 🙀 🕫 🔽 🚽 🖓 👘	▼ DefaultSession ▼
SmartAnalog Smart	
✓ III	× 
Ready	INS



#### (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





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。)

RENESAS

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。

🖗 SmartAnalog - High-performanc	e Embedded Workshop - [intprg.c*]		
◇ ファイル(F) 編集(E) 表示(V)	プロジェクト(P) ビルド(B) デバッグ(D)	基本設定(U) ツール(T) テスト(S) ウ	フィンドウ(W) ヘルプ(H) _ B ×
□ 🕞 🖬 🕼 🖨   ½ 🖻 💼	アクティブプロジェクトに設定( <u>S)</u> プロジェクトの挿入( <u>I</u> ) 依存関係( <u>D</u> )	▶ 🍇 📗 🏭 🍪 🎬 🛗 👗 Debug	DefaultSession
	構成の編集( <u>E</u> )	software int 12) 2_receive(vect=12)	
·····································	プロジェクトタイプの作成( <u>P</u> )	{} oftware int 13) key(yest=12)	
·····································	ファイルの対象(E) ファイルの拡張子(E)		
⊡ — 🔄 Dependencies — 📄 cstartdefh — 📄 initscth	コンポーネントギャラリ( <u>C</u> )	software int 14)*/ onverter(vect=14) ed_short_g_temp:	
∰ resetprøh ∰ sfr_r835ah ∰ typedefineh	90         extern volatile unsign           91         void _ad_converter(voi           92         g_temp = 0x03ff &           93         g_count+;           94         adst = 1;           95         }           96         // SSUIC/IICIC           98         #pragma interrupt _s           99         void _ssuic(void);           100         void _ssuic(void) {}	ed int g_count; d){ ad3; /* Write cc (software int 15) suic(vect=15)	onversion result of AN11 to b
🚰 P 👰 T 🔍 N 💟	SmartAnal. S resetprec S i	ntprg.c*	• •
A 01 01 AL AT 21 21 0	à 🖬 🤰		
<	e 1 ) Eind in Eilee 2 ) Marro ) Teet ) Version	Control (	• •
		ult1 deskton Read-write (	25/220 2 1010
レントリルをノロンエクトに追加します		uitt desktop  Kead-wille	55/220  2  INS   //
🔅 'SmartAnalog'	プロジェクトにファイルを追加		

ファイルの場所(1):	🕌 SmartAnalog	▼ ← 🗈 💣 🖃 ▼		
名前	A	更新日時	種類	*
r_sadesigne	er.h	2013/01/16 13:28	H ファイル	
r_sadesigne	er_reg.c	2013/01/16 13:28	C ファイル	
r_sadesigne	er_reg.h	2013/01/15 13:10	H ファイル	
resetprg.c		2013/02/25 13:11	C ファイル	-
•	III		•	
ファイル名( <u>N</u> ):	"r_sadesignerh" "r_sadesigner_reg.c"	"r_sadesigner_regh"	追加	
ファイルの種類(工):	Project Files		<ul> <li>キャンセル</li> </ul>	
	▼ 相対パス( <u>R</u> )			



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。

🟟 SmartAnalog - High-performance Embedded Workshop - [intprg.c*]						
◇ ファイル(E) 編集(E) 表示(⊻) プロ:	ジェクト(Ⴒ)	<u> ビルド(B)</u> デバッグ( <u>D</u> ) 基本設定(U) ツーノ	L( <u>T</u> )	テスト( <u>S</u> )	ウィンドウ( <u>W</u> ) ヘルプ( <u>H</u>	) _ <i>8</i> ×
	<b>A</b> P	Renesas M16C Standard Toolchain	þ	Debug	✓ DefaultSe	ession 💌
SmartAnalog SmartAnalog C header file r_sadesigner h r_sadesigner h r_sadesigner h r_sadesigner h r_sadesigner reg h r_sadesigner reg c r_sadesigner reg c r_sadesigner reg c r_sadesigner reg c r_sadesigner reg h r_sadesigner reg h	行番… S… 76 77 78 80 81 82 83 84 85 86 83 84 85 86 87 88 89 90 91 92 93 93 92 93 93 95 96 97 98 99 97 98 98 97	<ul> <li>◇ コンパイル(E) Ctrl+F</li> <li>△ ビルド(E) F</li> <li>描 ビルド(E) F</li> <li>描 すべてをビルド(A) 複数ビルド(M) クリーンアクティブプロジェクト(L)</li> <li>益 クリーン全プロジェクト(E) すべての依存関係を更新(U)</li> <li>▲ ツールの中止(S) Ctrl+Breat</li> <li>ビルドから除外/除外の解除(I)</li> <li>ビルドフェーズ(P)</li> <li>ビルドの構成(C)</li> <li>リンク順の指定(K)</li> <li>Makeファイルの生成(G)</li> <li>ord _ssuit(void) {}</li> <li>// vector 16 reserved</li> </ul>	ik	/* Write	conversion result of #	ANII to b
P 🛃 T 🔍 N 🚺	🗇 SmartAn	al] 🧼 resetprec 🥔 intprec*				
💐 OJ OT AJ AT 😫 XT 🖉 🖻 🖬	1 ?					
<ul> <li></li> <li></li> <li></li> <li>Neuild ∧ Debug ∧ Find in Files 1 ∧</li> </ul>	Find in Files 2	$\lambda$ Macro $\lambda$ Test $\lambda$ Version Control /				* *
		🖪 📰 🔝 Default1 desktop	Read-	write	95/220 2	INS



Renesas M16C Standard Toolchain	2 - ×-
コンフィグレーション:	באארק   דעבער דעס   דעראר אין איז איז איז איז דער איז איז דער איז
Debug 💌	カテゴリ(ゾ): 出力 🔽
⊡	▼ [-G]ソースデバッグ情報をアブソリュートモジュールファイルに出力する (D)
⊡… 🛄 C source file ⊡… 🧰 Assembly source file	□ [-U]未使用関数名の検出を実施する(U)
	□ [-₩]ウォーニング発生時、 アブソリュートモジュールファイルを生成しな い(L)
	マップファイル生成(G):
	[-M5]シノホル情報を含むマッフノアイルを生成する <u>▼</u>
	[-0]出力ファイルバス指定(E):
	\$(CONFIGDIR)¥\$(PROJECTNAME)×30 変更(M)
	リンカオプション(丁):
	-L "r8clib" -G -MS -O "\$(CONFIGDIR)¥\$(PROJECTNAME)×30" -
	Ostack,istack,heap_NE,rom_NE=0C000,rom_NO,data_SEIdata_NO,bss_N Ostack,istack,heap_NE,rom_NE=0C000,rom_NO,data_SEIdata_SOIdata_ NEIdata_NOIswitch_table,program,interrupt,vector=0FED8 - R8C
	OK キャンセル

[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。

SmartAnalog - High-performance Embedded Wo	rkshop - [intprg.c*]	
	ビルド(B) デパッグ(D) 基本設定(U) ツール(T)	テスト(S) ウィンドウ(W) ヘルプ(H) 🔤 🖉
📗 🗅 😅 🖬 🕼 🛛 🖧 🖡 🖻 🖀 🖌 🗍 🙀 🔎	Renesas M16C Standard Toolchain	Debug
SmartAnalog		
C header file	■ 9へてをビルド(A)   複数ビルド(M)	
r_sadesigner_regh 80	クリーン アクティブプロジェクト( <u>L</u> )	
I Source file 01 ↓	基 クリーン 全プロジェクト(E)	
→ 말 heap.c 84 → 말 initset c 85	」 すべての依存関係を更新( <u>∪</u> )	_
	メールの中止( <u>S</u> ) Ctrl+Break	
i r_sadesigner_reg.c 88 i resetprg.c 89	ビルドから除外/除外の解除(I)	_
E SmartAnalog.c 90 E ∰ Dependencies 91	ビルドフェーズ( <u>P</u> )	
i cstartdefh 32 93 initscth 93	ビルドの構成( <u>C</u> )	-
IIIr_sadesigner_regh 95 IIIresetprgh 96	リンク順の指定(近)	_
	Makeファイルの生成( <u>G</u> )	
	void _ssuic(void){}	
102	// vector 16 reserved	
	<u> </u>	<u>&gt;</u>
🔄 P , 🔄 T , 🔄 , 🚺 ,	Anal	
🕺 OJ OT AJ AT 🛛 🎗 🎗 T 🖉 🖻 🖬 🤶		
		*
		*
► Build 〈 Debug 〉 Find in Files 1 〉 Find in Files	2 À Macro À Test À Version Control /	4
すべてをビルドします	💷 🔝 🔝 Default1 desktop Rea	ad-write 95/220 2 INS //



#### 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。

SmartAnalog - High-performance Embedded Workshop	[SmartAnalog.c]
◇ ファイル(E) 編集(E) 表示(Y) プロジェクト(P) ビルト	B) デバッグ(D) 基本設定(U) ツール(I) テスト(S) ウィンドウ(W) ヘルプ(H)
🗌 🗅 😅 🖬 🎒 🤮 🐰 🖻 💼 🔂 🔛 🗛 🔍	同期デバッグ(Z) Debug ▼ DefaultSession ▼ 入
□ □ ■ [ 3 SmartAnalog ] / (7番… S. ソース □ ③ SmartAnalog ] / * ***** □ ③ C header file ] / * ***** □ ③ C header file ] / * ***** □ ③ C sadesigner h ] / * 10 □ ① C source file ] / * 10 □ ③ C source file ] / * 10 □ ③ C source file ] / * 10 ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○	デバッグセッション(B)     ****       デバッグの設定(D)     */**
	enerator (ver.4.10). */ */ *** e 目t カーソル位置まで実行( <u>C</u> ) e <i>I<sub>v</sub></i> <sub>c</sub> カーソル位置にPC値を設定( <u>P</u> ) 条件を指定して実行( <u>R</u> ) in( *, PC位置を表示( <u>Y</u> ) Ctrl+Shift+Y SAI P ステップオーバ( <u>W</u> ) F10 Con P ステップオーバ( <u>M</u> ) F10 P ステップオーバ( <u>M</u> ) F10 Con P ステップオーバ( <u>M</u> ) F10 P ステップ
20         Volat           27         volat           27         volat           ✓         SmartAnal	e u の ブログラムの停止( <u>H</u> ) res 初期化( <u>Z</u> ) 開 接続(N)
A OL OT AL AT 24 21 0 B P ?	開 接続解除(I)            メモリの保存( <u>A</u> )            メモリのペリファイ(E)
III  III  III  Build \ Debug \ Find in Files 1 \ Find in Files 2 \ Ma	> $\lambda$ Test $\lambda$ Version Control /
セッションの追加/削除	I II Default1 desktop Read-write 1/248 1 INS

デバッグセッション	8 ×
デバッグセッション( <u>D</u> ):	
DefaultSession SessionB8C_E8a_SYSTEM	追加( <u>A</u> )
	削除( <u>R</u> )
	名前を付けて保存(S)
	プロパティ( <u>P</u> )
現在のセッション(C): SessionR8C_E8a_SYSTEM	
OK	キャンセル

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。

エミュレータ設定	έ <mark>x</mark>
エミュレータモー	ド ファームウェア配置   MCU設定   通信ボーレート
MCUグルーナ	7(M) R8C/35C Group
デバイス( <u>D</u> )	R5F2135CC
モード	○ フラッシュメモリデータを消去して起動(e)
	○ フラッシュメモリデータを保持して起動(n)
	○ フラッシュメモリデータの書込み(E)
	○ CPU書き換えモードのデバッグ(U)
	□ デバッガ終了時にユーザープログラムを実行(x)
<b>電源</b> マロミュレ	ータから電源供給(P)(最大 300mA) C <u>3.3 V (* 5.0 V</u>
	OK キャンセル 「 今後、このダイアログボックスを表示しない

MCU グループ	<b>"R8C/35C Group"</b> を指定します。
デバイス	"R5F2135CC"を選択します。
[電源]-[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。







## 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)

```
″sfr_r835a.h″
#include
#include
           "r_sadesigner_reg.h"
           "r_sadesigner.h"
#include
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");
    }
                                            /* read start condition
    g_temp_ref = g_temp;
                                                                              */
```



(2) 初期化関数 (SmartAnalog.c に追加)

```
void hwinit(void)
{
    unsigned char osc_stab = 128;
                                            /* Disable interrupts */
    _asm(" FCLR I");
    prc0 = 1;
                                            /* Protection off */
                                            /* Pin P4_6 and P4_7 are
    cm13 = 1;
                                               configured as XIN and XOUT \ast/
    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
                    = 0 \times 00;
                    = 1;
                                           /* Smart Analog IC Reset=H
    p2_5
                                                                              */
    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
                                                                              */
                                           /* SSU O:active
                    = 0;
    mstiic
                                                                              */
    iicsel
                    = 0;
                                            /* selects SSU function
                                                                              */
                                            /* RE bit and TE bit clear
    sser
                    = 0x00;
                                                                             */
                    = 0x40;
                                            /* clock synchronous
                                                                              */
    ssmr2
                                            /* ssck is serial clock pin
                                                                             */
                                            /* standard mode select
                                                                              */
```



- (4) SPI 関数(SmartAnalog.c に追加)
- (a) R\_SAIC\_Create()

/**************************************	***************************************	*******/
<pre>/* R_SAIC_Create();</pre>		*/
/************************	****************	*******/
<pre>void R_SAIC_Create(void)</pre>		
{		
volatile uint16_t w_co	punt;	
$p2_5 = 0;$	/* Analog IC Reset	*/
/* Change the waiting	time according to the system $*/$	
for $(w_count = 0U; w_count$	$ount < 10; w_count++)$	
{		
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;
                                  /* transmit enable
   te_sser = 1;
                                                            */
   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");
      }
                                     /* wait transmit data empty
      while(tdre_sssr != 1);
                                                                 */
      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 <u>http://japan.renesas.com/contact/</u>

すべての商標および登録商標は、それぞれの所有者に帰属します。

改訂記録

		改訂内容		
Rev.	発行日	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
	2.	るに生した損害に関し、当社は、一切での負任を良いません。 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. 5	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。 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
I	0.	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。
I		標準水準: コンピュータ、OA機器、通信機器、計測機器、AV機器、
I		家電、工作機械、パーソナル機器、産業用ロボット等
I		高品質水準:輸送機器(自動車、電車、船舶等)、交通用信号機器、
I		防災・防犯装置、各種安全装置等
I		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
I		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
I		途に当社製品を使用したことによりお客様またば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、こ不明点がある場合は、当社宮業にお問い へれたください
I	6	「白わじてんさい。 当社創品をご使用の際け 当社が指定する最大定体 動作電源電圧範囲 放執特性 実装条件その他の保証範囲内でご使用ください 当社保証範囲を招えて当社創
I	0.	コームな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。
I	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
I		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
I		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
I		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。
I	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
I		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
I		関して、当社は、一切その責任を負いません。
I	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
I		た、当在製品およい技術を大重破壊兵器の開発等の日的、車事利用の日約その他車事用速に使用しないでくたさい。当在製品または技術を輸出する場合は、「外 国告辞事が対応回発目は、その供給回題連ば会ま進定し、かかるは今の定めるトロスにとし必要な手続きたってください。
I	10	国為省及び外国員勿広」ての他職山関連広市を短寸し、かかる広市のためるところにより必要な十柄を打つしてたさい。 な家様の転売等に上川 太ご注意書き記載の諸冬班に抵触して当社制具が使用され その使用から損害が生じた提合 当社は何らの表任主負わず な家様にてご負
I	10.	- 03者物の私北寺により、本に江志音と記載の船未住に強屈して当社装品が使用され、その使用がち渡音が生じた場合、当社は何ちの負任も負わり、の者物にそと負 相して頂きますのでごて承ください
	11.	」でで、そのでは、アダマビーが、くれてい。 . 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上パルネサスエレクトロニクス株式会社がその総体主の詳決後の温平数
		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。
1		

# RENESAS

#### ルネサスエレクトロニクス株式会社

http://www.renesas.com

(03)5201-5307

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)

■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/

■営業お問合せ窓口

※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。