RENESAS

アプリケーションノート

R01AN3860JJ0110 Rev.1.10 2018.04.11

要旨

本アプリケーションノートでは、マイコン内蔵周辺 I/O ドライバ自動生成ツール(以下、コード生成ツー ルと呼びます)で作成したマイコン周辺機能の制御プログラム(デバイス・ドライバ・プログラム)を RZ/T1 グループ 初期設定サンプルプログラムへ組み込む手順について説明します。

本サンプルプログラムでは、コンペアマッチタイマ(CMT)を周期カウント動作させ、コンペアマッチ割り込みにより LED を点灯 / 消灯させます。

対象デバイス

RZ/T1 グループ

本アプリケーションノートを他のマイコンへ適用する場合、そのマイコンの仕様にあわせて変更し、十分 評価してください。



H V7
H V7

1.	仕様		
2.	動作環境		
3.	関連アプ	リケーションノート	5
4.	周辺機能	説明	6
5.	ハードウ	ェア説明	7
	5.1 /\-	ードウェア構成例	7
	5.2 使/	用端子一覧	
6.	ソフトウ	ェア説明	9
	6.1 動作	乍概要	
	6.1.1	プロジェクト設定	
	6.1.2	使用準備	
	6.1.3	例外処理ベクタテーブル	
	6.2 使/	用割り込み一覧	
	6.3 ⊐-	ード生成ツール出力コードの組み込み手順	
	6.3.1	コード生成ツールを使用したコードの作成について	
	6.3.2	RZ/T1 グループ 初期設定サンプルプログラムへの取り込み	
	6.4 固定	定幅整数一覧	
	6.5 関導	数一覧	
	6.6 フ	コーチャート	
	6.6.1	ローダプログラム処理	
	6.6.2	コード生成ツール作成アプリケーションプログラム処理	
	6.6.3	共通 main 処理	
	6.6.4	ユーザ使用タイマ(CMT0)割り込み処理	
7.	サンプル	プログラム	
8.	参考ドキ	ュメント	
9.	注意事項		

1. 仕様

表 1.1 に使用する周辺機能と用途を、図 1.1 に動作環境を示します。

表1.1 使用する周辺機能と用途

周辺機能	用途
クロック発生回路(CPG)	CPUクロックおよび低速オンチップオシレータで使用
割り込みコントローラ(ICUA)	コンペアマッチ割り込み(CMIO)で使用
コンペアマッチタイマ(CMT)	コンペアマッチタイマの周期カウント動作で使用
バスステートコントローラ (BSC)	CS0、CS1空間にNORフラッシュメモリを接続、および、CS2、 CS3空間でSDRAMを接続するために使用
SPIマルチI/Oバスコントローラ(SPIBSC)	SPIマルチI/O空間にシリアルフラッシュメモリを接続するため に使用
エラーコントロールモジュール(ECM)	ERROROUT#端子の初期化
汎用入出力ポート	LEDの点灯および消灯のための端子制御に使用



図 1.1 動作環境

2. 動作環境

本アプリケーションノートのサンプルプログラムは、下記の環境を想定しています。

表2.1 動作環境

項目	内容
使用マイコン	RZ/T1グループ
動作周波数	CPUCLK = 450MHz
動作電圧	3.3V
統合開発環境	IAR システムズ製 Embedded Workbench for ARM Version 7.80.2 ARM製 DS-5 TM 5.25 RENESAS製 e ² studio 5.2.0
コード生成ツール	RENESAS製 AP4 1.07 注. e ² studioは、同機能のAP4 1.04相当のコード生成プラグインが含まれています。
動作モード	SPIブートモード 16ビットバスブートモード
使用ボード	RZ/T1 評価ボード (RTK7910018C00000BE)
使用デバイス (ボード上で使用する機能)	 NOR フラッシュメモリ (CS0、CS1空間に接続) メーカ名: Macronix International Co、型名: MX29GL512FLT2I-10Q SDRAM (CS2、CS3空間に接続) メーカ名: Integrated Silicon Solution Inc、型名: IS42S16320D-7TL シリアルフラッシュメモリ メーカ名: Macronix International Co、型名: MX25L51245G



3. 関連アプリケーションノート

本アプリケーションノートに関連するアプリケーションノートを以下に示します。併せて参照してくださ い。

- RZ/T1 グループ初期設定(R01AN2554JJ)
- RZ/T1 グループコンペアマッチタイマ (CMT) (R01AN2555JJ)



4. 周辺機能説明

動作モード、クロック発生回路(CPG)、コンペアマッチタイマ(CMT)、割り込みコントローラ (ICUA)、バスステートコントローラ(BSC)、SPIマルチ I/O コントローラ(SPIBSC)、エラーコントロー ルモジュール(ECM)、リセット、汎用入出力ポートについての基本的な内容は、RZ/T1グループ・ユー ザーズマニュアル ハードウェア編を参照してください。



5. ハードウェア説明

5.1 ハードウェア構成例

図 5.1 にハードウェア構成例を示します。



図 5.1 ハードウェア構成例



使用端子一覧 5.2

表 5.1 に使用端子と機能を示します。

表5.1 使用端子と機能

端子名	入出力	内容
A1~A25 ^{注1}	出力	NORフラッシュメモリ、SDRAMへのアドレス信号出力
D0~D15 ^{注1}	入出力	NORフラッシュメモリ、SDRAMのデータ信号入出力
CS0# ^{注1}	出力	CS0空間に接続されたNORフラッシュメモリへのデバイス選択信号出力
CS1# ^{注1}	出力	CS1空間に接続されたNORフラッシュメモリへのデバイス選択信号出力
CS2# ^{注1}	出力	CS2空間に接続されたSDRAMへのデバイス選択信号出力
CS3# ^{注1}	出力	CS3空間に接続されたSDRAMへのデバイス選択信号出力
RAS# ^{注1}	出力	SDRAMへのRAS#制御信号出力
CAS# ^{注1}	出力	SDRAMへのCAS#制御信号出力
RD/WR# ^{注1}	出力	SDRAMへのリード制御信号またはライト制御信号出力
CKE ^{注1}	出力	SDRAMへのCKイネーブル制御信号出力
RD# ^{注1}	出力	リード中を示すストローブ信号出力
BS#	出力	本サンプルプログラムでは、未使用
WE0#/DQMLL ^{注1}	出力	D15~D8に対するライトストローブ信号出力
WE1#/DQMLU ^{注1}	出力	D7~D0に対するライトストローブ信号出力
SPBSSL ^{注1}	出力	スレーブセレクト
SPBCLK ^{注1}	出力	クロック出力
SPBMO/SPBIO0 ^{注1}	入出力	マスタ送出データ/データ0
SPBMI/SPBIO1 ^{注1}	入出力	マスタ入力データ/データ1
SPBIO2 ^{注1}	入出力	データ2
SPBIO3 ^{注1}	入出力	データ 3
MD0	入力	動作モードの選択
MD1	入力	MD0 = "L"、MD1 = "L"、MD2 = "L"(SPIフートモード) MD0 = "L"、MD1 = "H"、MD2 = "L"(16ビットバスブートモード)
MD2	入力	
PM7 ^{注1}	出力	LED10の点灯および消灯

注. 1.#は負論理(またはアクティブロー)を示す記号です。 注1. 各端子は、コード生成ツールで端子機能を設定。



6. ソフトウェア説明

6.1 動作概要

本サンプルプログラムは、RZ/T1 グループ初期設定サンプルプログラムをベースとし、ローダプログラム (ローダ)部は、初期設定サンプルプログラムのローダプログラムをそのまま利用し、ユーザアプリケー ションプログラム (ユーザアプリケーション)部にはコード生成ツールで作成したコードと共通 main の コードを取り込んで使用します。実際の組み込み手順についても後述します。

RZ/T1 グループ初期設定サンプルプログラムの動作詳細については、RZ/T1 グループ初期設定アプリケーションノートを参照してください。

図 6.1 に本サンプルプログラムの構成概要を示します。







図 6.2 にローダ部とユーザアプリケーション部の動作概要を示します。

ローダ部では、ブート処理後にスタック設定やユーザアプリケーションプログラムのコピー、ユーザアプ リケーションプログラムの先頭番地への分岐をします。

以下の図に示すようにクロック設定とバスコントローラ設定は、従来の初期設定サンプルプログラムの ローダ部とコード生成されたユーザアプリケーション部のそれぞれで設定されています。このため本サンプ ルプログラムでは、ローダ部のクロック設定とバスコントローラ設定を無効化することで設定の競合を避け ています。

ユーザアプリケーション部は、コード生成ツールで作成された割り込み、クロック、バス設定および各周 辺機能の設定を行います。使用する動作モード(SPIブート、16ビットバスブート)毎に設定します。

また各動作モードによらない共通 main 関数として、コンペアマッチタイマ(CMT)を周期カウント動作 させ、コンペアマッチ割り込みにより LED の点灯 / 消灯をする設定を行います。

本組み込み手順によって、初期設定サンプルプログラムのローダ部を利用してコード生成ツールでユーザが自由に設定したコードを取り込むことが可能となります。



図 6.2 ブート処理後の動作概要

RENESAS

6.1.1 プロジェクト設定

本サンプルプログラムは以下の3種類のプロジェクトを含みます。

- ① RZ_T1_init_boot —— RZ_T1_init_nor_boot.eww : 16ビットバスブート版 RZ_T1_init_serial_boot.eww : SPIブート版
- ② RZ_T1_init_ram —— RZ_T1_init.eww : RAM実行版

初期設定サンプルプログラムにコード生成ツールで作成したプログラムを組み込んだ際のフォルダ構成を **表 6.1**に示します。本サンプルプログラムで組み込んだフォルダ、ファイルについては太字部で示します。 太字部以外は初期設定サンプルプログラムと同じ構成です。

開発環境およびプロジェクト設定については、RZ/T1 グループ初期設定アプリケーションノートを参照してください。

		主なフ	オルダ構	标(EWARM)	
プロジェクトフォルダ	サブフォルダ				備考
RZ_T1_init_boot	inc				初期設定編のインクルードファイル格納フォルダ
	lib				EWARM用ライブラリフォルダ
	src	cg_src_nor		cg_src *.cgp *.ipcf	【コード生成ツールで作成したフォルダ】 (16 ビットバスブート版(NOR)) ・ ソースコード出力フォルダ(cg_src) ・ コード生成ツールワークファイル(.cgp) ・ プロジェクト・コネクション・ファイル(.ipcf)
		cg_src_serial		cg_src *.cgp *.ipcf	【コード生成ツールで作成したフォルダ】 (SPIブート版(Serial)) ・ ソースコード出力フォルダ(cg_src) ・ コード生成ツールワークファイル(.cgp) ・ プロジェクト・コネクション・ファイル(.ipcf)
		common			初期設定編のサンプルプログラムフォルダ
		drv			ドライバフォルダ
		sample		user_main.c	初期設定編のmain用フォルダ 共通 main (user_app_main)
RZ_T1_init_ram	inc				RAM版初期設定編のインクルードファイル
	lib				EWARM用ライブラリフォルダ
	src	cg_src_ram	F	cg_src *.cgp *.icpf	【コード生成ツール追加フォルダ】 (RAM版) • ソースコード出力フォルダ(cg_src) • コード生成ツールワークファイル(.cgp) • プロジェクト・コネクション・ファイル(.ipcf)
	·	common			初期設定編のサンプルプログラムフォルダ
		drv			ドライバフォルダ
		sample		user_main.c	RAM版初期設定編のmain用フォルダ 共通 main (user_app_main)

表6.1 コード生成ツールで作成したコードを組み込んだ初期設定サンプルプログラムのフォルダ構成 (1/2)



				主なフォルダ権	冓成(e ² studio)	
プロジェクトフォルダ		サブフォル	ダ			備考
RZ_T_nor_sample –		.setting		CodeGenerator		【コード生成ツールで作成したフォルダ】 (16 ビットパスブート版(NOR)) ・ コード生成ツールワークファイル(.cgp)
	\vdash	inc				初期設定編のインクルードファイル格納フォルダ
		src		cg_src		【コード生成ツールで作成したフォルダ】 (16 ビットパスブート版(NOR)) ・ ソースコード出力フォルダ(cg_src)
				common		初期設定編のサンプルプログラムフォルダ
				drv		ドライバフォルダ
				sample -	user_main.c	初期設定編のmain用フォルダ 共通main (user_app_main)
				iodefine.h ^{注1}		Inc フォルダから移動した e ² studio 用iodefine
RZ_T_sflash_sample -		.setting		CodeGenerator		【コード生成ツールで作成したフォルダ】 (SPIブート版(Serial)) ・ コード生成ツールワークファイル(.cgp)
	┝	inc				初期設定編のインクルードファイル格納フォルダ
	L	SIC		cg_src		【コード生成ツールで作成したフォルダ】 (SPIブート版 (Serial)) ・ ソースコード出力フォルダ(cg_src)
				common		初期設定編のサンプルプログラムフォルダ
				drv		ドライバフォルダ
				sample -	user_main.c	初期設定編のmain用フォルダ 共通main (user_app_main)
				iodefine.h ^{注1}		Inc フォルダから移動した e ² studio 用iodefine
RZ_T_ram_sample —		.setting		CodeGenerator		【コード生成ツールで作成したフォルダ】 (RAM版) • コード生成ツールワークファイル(.cgp)
	L	inc				RAM版初期設定編のインクルードファイル
	L	SIC	Τ	cg_src		【コード生成ツールで作成したフォルダ】 (RAM版) ・ ソースコード出力フォルダ(cg_src)
				common		初期設定編のサンプルプログラムフォルダ
				drv		ドライバフォルダ
				sample -	user_main.c	RAM版初期設定編のmain用フォルダ 共通main (user_app_main)
				iodefine.h ^{注1}		Inc フォルダから移動した e ² studio 用iodefine

表6.2 コード生成ツールで作成したコードを組み込んだ初期設定サンプルプログラムのフォルダ構成 (2/2)

注1. e²studio環境にコード生成ツールで作成したコードを組み込む際は、各プロジェクトのincフォルダにあるiodefine.hをプロジェクトフォルダの直下に移動してください。



6.1.2 使用準備

ご使用になられるプロジェクトによって、RZ/T1 評価ボード (RTK7910018C00000BE) 上にある SW4 の設 定が異なります。表 6.3 に SW4 の設定を示します。SW4 の各設定につきましては、RZ/T1 評価ボード RTK7910018C00000BE ユーザーズマニュアルに記載しています。詳細は、「8. 参考ドキュメント」を参照 してください。

表6.3 SW4の設定

サンプルプログラム	SW4-1	SW4-2	SW4-3	SW4-4	SW4-5	SW4-6
16ビットバスブートモード版	ON	OFF	ON	ON	ON	OFF
SPIブートモード版	ON	ON	ON	ON	ON	OFF
RAM実行版			上記いずれか	の SW4 設定		

6.1.3 例外処理ベクタテーブル

RZ/T1 には 7 種類の例外処理(リセット、未定義命令、ソフトウェア割り込み、プリフェッチアボート、 データアボート、IRQ、FIQ)があり、コード生成ツールを使用する場合 0000 0000h 番地から 34 バイトの領 域(0000 0000h 番地~0000 0024h 番地)を使用します。例外処理ベクタテーブルには、各例外処理への分 岐命令を記述します。

表 6.4 に本サンプルプログラムにおける例外処理ベクタテーブルの内容を示します。必要に応じて変更してください。

例外	ハンドラアドレス	備考
RESET例外	0000 0000h	自身へ分岐(暴走防止)
未定義命令例外	0000 0004h	自身へ分岐(ユーザで自由に定義)
ソフトウェア例外	0000 0008h	自身へ分岐(ユーザで自由に定義)
プリフェッチアボート例外	0000 000Ch	自身へ分岐(ユーザで自由に定義)
データアボート例外	0000 0010h	自身へ分岐(ユーザで自由に定義)
Reserved	0000 0014h	自身へ分岐(ユーザで自由に定義)
IRQ例外	0000 0018h	自身へ分岐(暴走防止)
FIQ例外	0000 001Ch	自身へ分岐(コード生成ツールで上書き)

表6.4 例外処理ベクタテーブル

注. コード生成ツールが作成したコードにより、上記FIQ例外を含む0000 001Ch~0000 0024hまでをアプリケーションプログラ ム内で上書きするためご注意ください。



6.2 使用割り込み一覧

表 6.5 にサンプルプログラムで使用する割り込みを示します。

表6.5 サンプルプログラムで使用する割り込み

割り込み(要因ID)	優先度	処理概要
CMT0割り込み(CMI0)	15	コード生成ツールで指定したインターバル時間(100ms)に よってコンペアマッチするたび、LED10が点灯/消灯を繰り 返します。



6.3 コード生成ツール出力コードの組み込み手順

ここでは EWARM とコード生成ツールを使用した場合の手順例を示します(動作モードは SPI ブートモードです)。

6.3.1 コード生成ツールを使用したコードの作成について

本サンプルでは、CMT0を使用した周期カウント動作によるコンペアマッチ割り込みと LED10の点灯 / 消 灯を行うため PORTM7の設定を行います。サンプル例を以下に示します。

- (1) コード生成ツールを起動
- (2) 新規プロジェクトを作成

新規プロジェクト作成する場合は、以下の設定が必要になります。

マイクロコントローラ(R7S910018CBGを使用する場合)を選択、使用するコンパイラ(IAR EWARM)、プロジェクト名、プロジェクト場所を決定します。



• プロジェクト名とプロジェクト場所についての注意点

はじめにコード生成ツールを使用してプロジェクトを作成する場所は、ユーザの指定する任意の場 所で構いませんが、本サンプルプログラムでは各動作モード毎にコード生成ツールの出力ファイル を格納する場所を分けています。

サンプル例)

16 ビットバスブートモード用フォルダ名:cg_src_nor

SPI ブートモード用フォルダ名:cg_src_serial

作成した各フォルダを初期設定サンプルプログラムへ組み込む方法については、「6.3.2の(1)

コード生成ツールで作成した環境の移動(コピー)」を参照ください。

RENESAS

(3) コンペアマッチタイマ(CMT)、ポートの設定
【コンペアマッチタイマ設定】
使用するコンペアマッチタイマ設定を下記のように設定してください。
使用チャネル: CMT0
クロック設定: PCLKD/512
インターバル時間:100ms
コンペアマッチ割り込み許可(CMI0):使用
優先順位:レベル15
画面① CMT0 設定

/ロジェクト・ツリー	9 × 🧏 周辺機能 🧾 コード・プレビュー 🕋	プロパティ
	🔁 コード生成 🏂 💕 📓 🏭 🗱 📁 💷 🤅	3 0. 0 0 0. 0 0 0 <i>8 8 7 4 7 a</i> 2 4 4 4 4 4 4
	CMT0 CMT1 CMT2 CMT3 CMT4 CM	175
R_CMT0_Create	- コンパアマッチタイマ動作語定	
R_CMT0_Start	◎ 使用しない	● 使用する
	- クロック設定	
€ CMT2	PCLKD/8 PCLKD/32	PCLKD/128 PCLKD/512
	0	0.111
	-インターバル時間設定	
-U r cg cmt user.c	インターバル時間	100 ms 👻 (実際の値: 99.997013)
CMT0		
	-割り込み設定	
R_CMT0_Create_UserInit	✓ コンペアマッチ割り込みを許可(CMI0)	
	優先順位	レベル15
m → CMT4		
🕁 🛶 СМТ5		
🚽 🔰 r_cg_cmth		
冬周辺機能で"使用する"を選択す	ると対応した機能の設定が出来ます	必要に応じて追加の設定をしてください また 設定



٦

【ポート設定】 使用するポート設定を下記のように設定してください。 使用端子: PORTM7 (LED10) 入出力:出力 出力設定:"1を出力"に設定

• 画面② PORTM7 設定

Port0		Port1	Port2	Port3	Port4	Port5	Port6	Port?
PortE	1	PortF	PortG	PortH	PortJ	PortK	PortL	PortM
PM0 使用し 	い ◎ 入力	◎ 出力	入力ブルアップ振	抗/ブルダウン抵抗無効	•	[] 1を出力		
PM1 ④ 使用し	い ◎入力	◎ 出力	入力プルアップ抵	抗/ブルダウン抵抗無効	¥	[] 1を出力		
PM2 ④ 使用し	い ◎入力	◎ 出力	入力プルアップ振	抗/ブルダウン抵抗無効		□ 1を出力		
PM3	ぬ の入力	◎ 出力	入力ブルアップ抵	抗/ブルダウン抵抗無効		□ 1を出力		
PM4	দ্য লামান	◎ 屮力	「入力 プルマップ 挿	お/ゴルガウンがお無か				
PM5		0 807	(7077/177715	167 718777718188880				
 使用し PM6 	201 ◎ 入力	◎ 出力	入力プルアップ抵	抗/プルダウン抵抗無効	*	18出力		
● 使用し	ぬ の入力	◎ 出力	入力ブルアップ抵	抗/ブルダウン抵抗無効	×	🗌 1を出力		
PM7 ⑦ 使用し	৯০ ⊚ ১⊅	• 出力	入力プルアップ抵	抗/ブルダウン抵抗無効	¥	🔽 1を出力		
								•

(4) クロック、バス機能の設定

本設定例は、初期設定サンプルプログラム (SPI ブートモード) と同じ設定となるよう例示しています。 【クロック発振回路設定】 下記のように設定してください。 ブートモード設定: SPI ブート PLL1 回路設定: 動作にチェック 低速オンチップオシレータ (LOCO) 設定: 動作にチェック クロックソース: PLL1 CPU クロック (CPUCLK): 450 (MHz) 外部バスクロック: 75 (MHz)



Г

画面③クロック設定

	0 0 0 0 0 0 0 0 <i>Q Q Z</i> ==	8	12 0 14 0 23
クロック設定 デバッガインタフェース設定 ブロック		.,	
-ブートモード設定	-		
◎ 16ビット/32ビットバスブート	● SPIプート		
-メインクロック発振器設定			
メインクロック発振源	発振子/外部発振		(OSCTH 端子の状態で設定されます)
周波数	25		(MHz)
発振停止検出	無効	•	
- PLLO 回路設定			
周波数	1200		(MHz)
PLL1 回路設定			
☑ 動作 国注意教	900		(MHz)
/4///2020	300		(191727
- IKEAフテザクオラレー気に000/igg/E			
周波数	240		(kHz)
-内部クロック設定(クロックソースはPLLOまたはPLI	L1)		
クロックソース	PLL1	•	
CPUクロック (CPUCLK)	450	•	(MHz)
システムクロック (ICLK)	150		(MHz)
高速周辺モジュールクロック (PCLKA)	150		(MHz)
低速周辺モジュールクロック (PCLKB)	75		(MHz)
外部バスクロック (CKIO)	75	•	(MHz)
トレースI/Fクロック (TCLK)	150	-	(MHz)
-内部クロック設定(クロックソースはPLLO) ―――			
高速周辺モジュールクロック (PCLKC)	150		(MHz)
低速周辺モジュールクロック (PCLKD)	75		(MHz)
低速周辺モジュールクロック (PCLKE)	75	•	(MHz)
低速周辺モジュールクロック (PCLKF)	60	•	(MHz)
低速周辺モジュールクロック (PCLKG)	60	•	(MHz)
低速周辺モジュールクロック (PCLKH)	60		(MHz)
高速シリアルクロック (SERICLK)	150	•	(MHz)
- IWDTクロック設定			
IWDT クロック (IWDTCLK)	120		(kHz)
-ECMクロック設定	240		(LH-)
EOM/19/ (EOMOLK)	240		(KH2)
- Ethernetクロック設定 EthernetクロックD (ETCLKD)	12.5	-	(MHz)
EthernetクロックE (ETCLKE)	25	-	(MH ₂)
			(
- Δ 2 50 59 設定 Δ Σ 1/F 50 50 (供給元選択(ch.0~ch.2))	PLL0(Master動作)	-	
⊿ ∑1/F(ch1~ch2)供給ch濯根	MCLK0~2からのクロック入力を使用	-	
4 Σ1/Ehn wh0 (DSCI K0)	25		(MHz)
4 51/5hn-b0/5/#3819	Tat	•	(1112)
コンドウロックの単注選択		•	
△ 2 J/Fクロック11共裕元/選択(ch.3)	PLLU(Master動作)	•	
Δ Σ I/Fクロック1 (DSCLK 1)	25	•	(MHz)
△ ΣI/Fクロック1極性選択	正転	•	



【バスステートコントローラ設定】 下記のように設定してください。 一般設定:バス動作設定を使用する CS0 設定:SRAM CS1 設定:SRAM CS2 設定:SDRAM CS3 設定:CS2 共通 アドレス端子選択設定:グループ選択 グループ選択設定:A1 ~ A25

画面④バスステートコントローラ設定(一般設定)

and the second	S0 CS1	CS2 CS3	CS4 CS5					
汉動作設定							+	
◎ 使用した	30)					• f	更用する	
部バス領域	設定			*				
V CSU (6	0000000h	63FFFFFFh,	40000000h	43FFF	FFFh ミラー)	SR	AM	•
🔽 CS1 (6	4000000h ~	67FFFFFFh,	44000000h	~ 47FFF	FFFh ミラー)	SR	AM	•
🔽 CS2 (6	8000000h ~	6BFFFFFFh,	48000000	a 4BFF	FFFFh ミラー)	SD	RAM	-
✓ CS3 (6	C000000h ~	6FFFFFFFh,	4C000000h	~ 4FFF	FFFh ミラー)	SD	RAM	
🔲 CS4 (7	0000000h ~	73FFFFFFh,	50000000h	~ 53FFF	FFFh ミラー)	SR	AM	+
CS5 (7	40000006 ~	77FFFFFFh	54000000	~ 57FFF	FFFh ≥ラー)	SB	AM	
ドレス端子遠	択股定 ——							
● グルー:	7進択					© 1	固定功度获	
ドレフ催子ガ	ループ選択設定	È						
1.624111.2	NG A1		•		次の単	端子まで A25	5	-
次の端子が								
次の端子が	定		[7] A3	🔽 A4	[7] A5	[] A6	[√] A7	
次の端子が 次の端子が ドレス端子設	定	[7] A2	101110	hained .	[7] A 12	[√] A14	I√ A15	
次の端子が ドレス端子設 回 A0 (7) A8	定	✓ A2 ✓ A10	V A11	🔽 A12	V mio		CI 000	
次の端子が 次の端子が ドレス端子設 〇 A0 〇 A8	定	 ✓ A2 ✓ A10 ✓ A18 	 ✓ A11 ✓ A19 	 ✓ A12 ✓ A20 	V A13	V A22	V M20	
次の端子が ドレス端子設 日 A0	定 ———	✓ A2		[]] A10		V A14	V A15	
次の端子が ドレス端子設 日 A0 マ A8 マ A16	定	 ✓ A2 ✓ A10 ✓ A18 	 ✓ A11 ✓ A19 	✓ A12✓ A20	✓ A13	✓ A22	V M20	



٦

CS0 :

領域設定:バス幅 16 ビット
バスタイミング設定:各サイクル数
外部ウエイト設定:外部ウエイトマスクの外部ウエイト入力無視
画面④バスステートコントローラ設定(CS0 設定)

- 般設定 CSU CS1 CS2 CS3 CS4 CS5		96 YA 1967 1997 199	
領域設定			_
バス幅	16ピット		•
バスタイミング設定	サイクル数	期間	
アクセスウェイト数	6	- 80	(ns)
RD#、WEn#ネゲート→アドレス、CSn#ネゲート遅延ステート数	0.5	• 6.666667	(ns)
アドレス、CSn#アサート→RD#、WEn#アサート遅延ステート数	2.5	- 33.333333	(ns)
同一CS空間に対するリードーリードサイクル間のアイドルステート挿入	0	• 0	(ns)
同一CS空間に対するリード-ライトサイクル間のアイドルステート挿入	0	• 0	(ns)
別CS空間に対するリードーリードサイクル間のアイドルステート挿入	0	• 0	(ns)
別CS空間に対するリードーライトサイクル間のアイドルステート挿入	0	• 0	(ns)
ライトーリード/ライトーライトサイクル間アイドルステート挿入	1	- 13.333333	(ns)
外部ウェイト設定			
外部ウェイトマスク	外部ウェイト入力無視	ŧ.	
外部ウェイトタイムアウト検出機能	禁止		*
外音Pウェイト数	255	3400	(ns)



CS1 :

領域設定:バス幅16ビット バスタイミング設定:各サイクル数 外部ウエイト設定:外部ウエイトマスクの外部ウエイト入力無視

•	画面④バスステー	トコントロ	ユーラ設定	(CS1 設定)
---	----------	-------	-------	----------

一般設定 CS0 CS1 CS2 CS3 CS4 CS5			
領域設定			_
バス幅	16ビット		•
-バスタイミング設定	サイカル教	甘日月月	
リードアクセスウェイト数	6 •	80	(ns)
ライトアクセスウェイト数	リードアクセスと同じ 👻	80	(ns)
RD#、WEn#ネゲート→アドレス、CSn#ネゲート遅延ステート数	0.5 👻	6.666667	(ns)
アドレス、CSn#アサート→RD#、WEn#アサート遅延ステート数	2.5 👻	33.333333	(ns)
同一CS空間に対するリードーリードサイクル間のアイドルステート挿入	0 👻	0	(ns)
同一CS空間に対するリードーライトサイクル間のアイドルステート挿入	0 🗸	0	(ns)
別CS空間に対するリードーリードサイクル間のアイドルステート挿入	0 👻	0	(ns)
別CS空間に対するリードーライトサイクル間のアイドルステート挿入	0 🗸	0	(ns)
ライトーリード/ライトーライトサイクル間アイドルステート挿入	1 👻	13.333333	(ns)
- 外部ウェイト設定			_
外部ウェイトマスク	外部ウェイト入力無視		-
外部ウェイトタイムアウト検出機能	禁止		
外部ウェイト教	255	3400	(ns)

CS2 :

領域設定:CS2 /バス幅 16 ビット (デフォルト) 領域設定:CS3 /バス幅 16 ビット (デフォルト) 種類:通常の SDRAM (デフォルト) アドレスビット数:各エリアのロウ・カラムアドレスビット数 モード設定:各エリアのバーストリード/バーストライト クロックセレクト:CKIO / 16 リフレッシュタイムコンスタントレジスタ値:36 コンペアマッチ割り込みを許可:チェック解除 バスタイミング設定:各サイクル数



• 画面④バスステートコントローラ設定(CS2 設定)





初期設定サンプルプログラムと同じ設定となるようユーザ関数の出力設定を行います。

• コード生成ツールでユーザ独自の初期化処理コード (r_xxx_user.c) を作成。



コード生成ツールで錠前マークが付いていない API 関数はデフォルトではコード生成されません。ここでは R_BSC_Create_UserInit 関数を使用するための設定を行います。プロジェクト・ツリーのコード・ プレビューで当該の API 関数を選択後に右クリックして「コード生成 (G)」を選択することで、実際の コード生成時に R_BSC_Create_UserInit 関数が r_cg_bsc_user.c ファイルと共に生成されるようになりま す。選択されると開いた錠前のマークが付きます。コード生成ツールの詳細な使用方法については、 「AP4, Appliet3 ユーザーズマニュアル 共通操作編 (R20UT3420JJ)」を参照ください。 作成されるファイル: r cg bsc user.c

コードを生成した API 関数に設定する記述については、コード生成後の(6)で行います。



٦

【SPI マルチ I/O バスコントローラ設定】 一般設定:機能設定を外部アドレス空間リードモード選択 転送速度設定:ベースビットレートを 75Mbps に設定 転送フォーマット設定:クロック遅延などを設定

•	画面⑤ SPI	マルチ I/O	バスコントロ	ーラ設定	(一般設定)
---	---------	---------	--------	------	--------

184480-		
◎ 使用しない	◎ 外部アドレス空間リードモード	◎ SP動作モード
転送速度設定		
ベースビットレート	75000 (Kbps)	〈実際の値:75000、エラー:0%〉
データアライメント設定		
📝 8ビット単位でデータスワップを	有効にする	
転送フォーマット設定 ―――		
SPBSSL信号極性設定		Low アクティブ
SPBSSLインアクティブ期間のSPE	BCLKの出力(CPOL)	出力値は0固定 ▼
送受信データに対するSPBCLKの)エッジ(CPHAT/ CPHAR)	偶数エッジで送信/偶数エッジで受信 →
SPBSSL信号のアクティブからSPE	BCLK信号からクロックが出力されるまでの期間(クロック遅延)	1SPBCLK -
転送最終時のSPBCLK信号エッ	1.5SPBCLK	
またごそんタマイルと、ケーのまたごそ月日からまで	の期間(ケアクセス)	1SPBCLK

【注意点】

初期設定サンプルプログラムでは、シリアルフラシュを高速にするために一度 SPIBSC を SPI 動作モードに設定し、シリアルフラッシュを Quad I/O モードに設定します。その後外部アドレス空間リードモードに再設定をしています。

一方、本手順では、SPIBSC は外部アドレス空間リードモードで、シリアルフラッシュは Single I/O モードの前提で設定をしています。使用するコマンドは FAST READ4B(0x0C)コマンドです。



٦

設定:

その他各種設定は、下図赤枠内を設定

• 画面⑤ SPI マルチ I/O バスコントローラ設定(設定)

ードコントロール設定				
読み出し動作	バーストリード	~		
バースト長	2	× (6	(ドット트)	
SPBSSLインアクティブ条件	各転送後にSPBSSLを化	ンアクティブ	*0/130/	~~~
ビット値設定				
コマンドビット幅	1ビット(SPBMOピンのコマ)	ンド出力〉		~
オプションデータビット幅	1ビット(SPBMOピンのオブ	ションのコマンドと	出力〉	~
アドレスビット幅	1ビット(SPBMOピンのアド	レス出力)		~
オブションデータビット幅	1ビット(SPBMOピンのオブ	ションデータ出力	>	~
ダミーサイクルビット幅	1ビット(SPBMIピンでのダミ	ミーHi−Z 出力)		~
データビット幅	1ビット(SPBMIピンのデータ	\$入力)		~
データ端子状能設定	L			
J Juli 11/12/6X/JE	SPBSSLインアクティブ時の	の状態	1ビット/ 2ビット幅時	の状態
SPBIO0端子	出力値 Hi-Z	~	出力値 Hi-Z	~
SPBIO1端子	出力值 Hi-Z	~	出力值Hi-Z	
SPBIO2端子	出力值 Hi-Z	~	出力値Hi-Z	<u> </u>
SPBIO3编子	出力値 Hi-Z	×	出力値 Hi-Z	
データフォーマット設定 ―――				
אעדב	0×0C			
アドレスビット数	32ビット	~		
🗌 オブショナルコマンドイネーフ	ブル			
オプションのコマンド	0×00			
□ オブションデータイネーブル				
オブションデータ数	1/5/1-5	~		
オブションデータ	0×00			
☑ ダミーサイクルイネーブル				
ダミーサイクル数設定	8サイクル	~		
32 ビット拡張アドレス設定				
外部アドレス有効範囲	ビット[25:0]	~		
上位アドレス値(EAV)	0×00			



(5) 端子機能設定

兼用機能として実際に使用する端子を設定してください。 サンプル例)端子配置表から端子機能の各設定画像と説明(1/3)

眩					
	選択機能	端子割り当て 検索端子割り当て 🔎	端子番号	入出力	備
-5	A0	設定されていません	◆ 設定されて…	.	
	A1	PG0/ A1/ PO2	▼ R7	▼ 出力	
マパルスユニッ	A2	PG1/ A2/ PO3	▼ V6	▼ 出力	
163	A3	PG2/ A3/ PO4/ TOC0/ RSPCK1	▼ R8	▼ 出力	
<u>پ</u> ۲	A4	PG3/ A4/ PO5/ TIC1/ MISO1	▼ T8	▼ 出力	
・レータ	A5	PG4/ A5/ PO6/ TOC1/ MOSI1	▼ V7	▼ 出力	
	A6	PG5/ A6/ TCLKA/ PO7/ SSL10	✓ V8	▼ 出力	
	A7	PG6/ A7/ TCLKB/ PO8/ SSL11	т9	▼ 出力	
-ス	A8	PG7/ A8/ PO9	▼ R9	▼ 出力	
-7	A9	PH0/ A9/ PO10	▼ V9	▼ 出力	
, 6	A10	PH1/A10/MTIOC2B/ PO11	▼ V10	▼ 出力	
	A11	PH2/ A11/ MTIOC2A/ PO12	▼ R10	▼ 出力	
AC	A12	PH3/ A12/ MTIOC1B/ PO13	▼ T10	▼ 出力	
シトローラ	A13	PH4/ IRQ4/ A13/ PO14	▼ R11	▼ 出力	
	A14	PH5/ A14/ PO15	▼ T12	▼ 出力	
-2	A15	PH6/ A15/ MTIOC7D/ RTS0#	▼ R12	▼ 出力	
6	A16	PH7/A16/MTIC5W	▼ V11	▼ 出力	
	A17	P20/ A17/ MTCLKD	▼ V12	▼ 出力	
	A18	P25/ A18/ MTCLKC/ TEND1	• Y14	▼ 出力	
	A19	P26/A19/MTIOC8D/DREQ1	▼ T14	▼ 出力	
6	A20	P27/ A20/ MTIOC8C/ TIOC80/ RTS0#	▼ R14	• 出力	
	A21	PT6/ A21/ DREQ2	y J20	▼ 出力	
	A22	PT7/ A22/ DACK2	✓ J19	▼ 出力	
	A23	PK2/A23	▼ F15	• 出力	
	Δ24	PK3/ A24	× G15	• 出力	
	A25	P97/ AN107/ IBO7/ A25/ ADTRG1	▼ E18	• 出力	
		P00/ D0/ MTIOC6A/ TIOCA1/ ADTRG1/ TRACECTL	▼ U18	• 入力/出力	
		P01/ D1/ MTIC5W/ TIOCA2	▼ V19	 → 入力/出力 	
	02	P02/ D2/ MTIC5V/ TIOCA3	▼ V20	▼ 入力/出力	
	D3	P03/ D3/ MTIC5U/ TIOCA4	▼ U20	 → 入力/出力 	
	D4	P04/ D4/ MTIOC3C/ TIOCA5	▼ U19	▼ 入力/出力	
	D5	P05/ D5/ MTIOC3A	▼ V18	▼ 入力/出力	
E	D6	P06/ D6/ MTIOC2B/ TIOCB0	▼ P15	▼ 入力/出力	
	D7	P07/ D7/ MTIOC2A/ TIOCB1	▼ P16	 λ カ/モカ 	
		PE0/ D8/ MTIOC18/ TIOC82/ TPACEDATA0	• T19	 λ h/#h 	
E E		PE1/ D9/ MTCI KD/ TIOCB3/ SSI 03/ TPACEDATA4	- T20	- 1 1/41	
		PE2/ IBO2/ D10/ MTCI KC/ TIOCB4/ SSI	• N15	- 11/41	
	D11	PE3/ IRO3/ D11/ MTIOCOD/ TIOCB5/ CTS1#/ SSI	• P18	 λ h/ψ h 	
	2 012	PE4/ D12/ MTIOCOB/ TIOCOD/ DTS1#/ SSL	- N16	- 11/41	
	2 013	PE5/ D13/ MTIOC0C/ TIOCC3/ TXD1/ MOS	- N18	- 11/80	
E E		PE6/ IRO6/ D14/ MTIOC00/ TIOC00/ PXD1/ MIS	• M16	- <u>1</u>	
			- 110	- ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	



サンプル例)端子配置表から端子機能の各設定画像と説明(2/3)

クロック発生回路 割り込みコントローラ	פאים	選択機能	端子割り当て 検索端子割り当て の		端子番号	入出力	備考
バスステートコントローラ DMA コントローラ		CS0#	P21/ IRQ1/ CS0#/ MTIC5V/ TIOCB1/ CTS0#	*	V13 •	出力	
レ/O ポート		CS1#	PD1/AN109/CS1#	*	E16 -	出力	
マルチファンクションタイマパルスユニッ		CS2#	P45/ CS2#	-	V15 •	出力	
ボートアウトブットイネーブル3 3月日日DWM タイマ		CS3#	PT4/ CS3#/ PO29		M19 -	出力	
バルコロットタイマパルスユニット		CS4#	設定されていません。	•	設定されて・	-	
プログラマブルパルスジェネレータ		CS5#	設定されていません。	*	設定されて・	-	
コンペアマッチタイマW		RD#	P22/ IRQ2/ RD#/ MTIOC7B/ TIOCD0/ SCK0	*	W14 -	出力	
FIFO 内臓ソリアルコミュニク ニンヨン・ I2C バスインタフェース		RD	P24/ IRQ12/ RD/ WR#/ RXD0	*	W13 -	出力	
シリアルペリフェラルインタフェース		WR#	P24/ IRQ12/ RD/ WR#/ RXD0	*	W13 -	出力	
SPI マルチレク バスコントローラ		BS#	P41/ BS#/ SCK0	•	Y15 👻	出力	
エラーコントロールモジュール		AH#	設定されていません	•	設定されて・・	-	_
12 ビットA/D コンバータ		WAIT#	設定されていません	*	設定されて・・	-	
ギガビットイーサネットMAC		WE0#	P36/ WE0#/ DQMLL/ PO0	*	T7 •	出力	
EtherCAT スレーフ・コンドローフ USB20HS ホストモジュール		WE1#	P37/ WE1#/ DQMLU/ PO1	*	т6 👻	出力	
CAN インタフェース		WE2#	設定されていません	*	設定されて・・	-	
シリアルサウンドインタフェース		WE3#	設定されていません	*	設定されて・・	-	
その他		DQMLL	P36/ WE0#/ DQMLL/ PO0	*	T7 •	出力	
		DQMLU	P37/ WE1#/ DQMLU/ PO1	•	т6 👻	出力	
		DQMUL	設定されていません	-	設定されて・・	-	
		DQMUU	設定されていません	-	設定されて・・	-	
		RAS#	P90/ AN100/ RAS#/ TIOCA5/ TXD4	*	F16 -	出力	
		CAS#	PK0/ CAS#/ PO31	•	H19 -	出力	
		СКЕ	P46/ CKE	-	V16 -	出力	
		скіо	P10/ IRQ0/ CKIO/ TIOCA0/ TRACECLK	-	Y19 -	出力	

サンプル例)端子配置表から端子機能の各設定画像と説明(3/3)

1 🕸 🔳 🖬 🔽							
クロック発生回路 エンコーダ I/F	ロック	選択機能	端子割り当て 検索端子割り当て 🔎]	端子番号	入出力	備考
割り込みコントローラ バスステートコントローラ	~	SPBCLK	P62/ SPBCLK	•	W1	▼ 出力	
DMA בטאם -5	~	SPBSSL	P60/ SPBSSL/ CTXD0/ TEND0	•	U1	▼ 出力	
I/O ボート マルチファンカションタイマパルフロニッ	-	SPBIO0	P63/ SPBMO/ SPBIO0	•	U2	▼ 入力/出力	
ポートアウトプットイネーブル3	-	SPBI01	P64/ SPBMI/ SPBI01	•	V2	▼ 入力/出力	
汎用PWM タイマ		SPBIO2	設定されていません	•	設定されて	▼ -	
16 ビットタイマパルスユニット プログラフブル パルスジェクレーク		SPBI03	設定されていません	•	設定されて	▼ -	
コンペアマッチタイマW	-	SPBMO	P63/ SPBMO/ SPBIO0	•	U2	▼ 出力	
FIFO 内蔵シリアルコミュニケーション・	-	SPBMI	P64/ SPBMI/ SPBI01	•	V2	▼ 入力	
I2C バスインタフェース	-						
SPI マルチI/O バスコントローラ							
ΔΣ インタフェース							
エラーコントロールモジュール							
12 ビットA/D コンバータ							
キカビットイーサネットMAC							
CAN インタフェーフ							



٦

- ツールバーにある表示から端子配置表を選択、端子機能にあるバスステートコントローラを選択し、 BS# 端子(PORT41)を設定してください。また、PORT10の高駆動出力設定については、コード生 成後の(6)で行います。
- 注. 兼用端子機能のデフォルト設定はリセット解除後の設定です。実際に使用する端子が選択されているか確認し てください。ロック選択していない端子は、別の設定で端子を使用した場合、変更される場合があります。不 要な競合を避けるために確認後は、使用する端子をロック選択(推奨)してください。また競合エラー等ない ことを確認してください。
- (6) コード生成とユーザ定義コードの編集
 (1)~(5)までの設定が完了後、コード生成を行います。
 コード生成するとコード生成ツールのプロジェクトを作成した cg_rc_serial フォルダ直下に cg_src フォルダとそのフォルダ内にソースコードとヘッダファイルが作成されます。
 - コード生成したコードの編集について
 コード生成ツールでは、コード生成を行う度にコードが上書きされます。ユーザが書き換えたコードを保護するためにはファイル生成制御設定を行い、マージ用コメント行にコードを記述する必要があります。ファイル生成制御の設定は、以下のように[ファイルをマージ]がデフォルトで選択してあります。



マージ用コメント行は、以下のようにコード生成ツールが出力した各コード内に記述されています。 マージ用コメント行の間にある記述は、再度コード生成をしても上書きされずに既存ファイルとのマー ジを行うため、ユーザ・コードの保護が可能です。



【バスステートコントローラの API 関数設定】
(4) で作成したファイルを編集して以下のコードを記述します。
対象ファイル: r_cg_bsc_user.c
対象 API 関数: R_BSC_Create_UserInit
PORT10の高駆動出力設定
バス用クロック(CKIO)に端子の機能を設定

CS2 空間の SDRAM 用 WCR 設定(BSC_CS2WCR_1)
 e2studio 版(AP4 1.04 相当)を使用される場合は、ここで追加の設定を行う必要があります。

SDRAM 初期化の API 関数設定(R_BSC_InitializeSDRAM)
 SDRAM 初期化の API 関数は、ユーザ独自の初期化処理コード内で関数コール
 対象 API 関数内のマージ用コメント行へ以下のコードを記述します。

【コンペアマッチタイマの API 関数設定】 (3) で作成したファイルを編集して割り込み処理に以下のコードを記述します。 対象ファイル:r_cg_cmt_user.c 対象 API 関数:r_cmt_cmi0_interrupt

```
• 周期カウント動作によるコンペアマッチ割り込み処理に LED10 点灯 / 消灯のコードを記述
対象 API 関数内のマージ用コメント行へ以下のコードを記述します。
```

```
/* Start user code. Do not edit comment generated here */
/* Toggle the PM7 output level(LED10) */
PORTM.PODR.BIT.B7 ^= 1; ←割り込み処理(LED10点灯/消灯)
/* End user code. Do not edit comment generated here */
```



【ユーザアプリケーションプログラムの API 関数設定】 コード生成を行うと cg_src フォルダ下にコード生成ツールの main 処理がある r_cg_main.c ファイルが作 成されます。ここでは、SPI ブートモード版の cg_src フォルダにある r_cg_main.c ファイルを編集しま す。 r_cg_main.c ファイルのユーザアプリケーション main 処理では、R_MAIN_UserInit 関数をコールしてい ます。本サンプルプログラムでは、この関数内に 16 ビットバスブート版、SPI ブート (シリアル) 版で 共通となる main 処理の関数コールを記述します。 対象ファイル: r_cg_main.c 対象 API 関数: R_MAIN_UserInit 追加する共通の main 関数名: user_app_main 対象 API 関数内のマージ用コメント行へ以下のコードを記述します。

	/* Start user code. Do not edit comment generated here */	
	<pre>user_app_main();</pre>	←追加する共通のmain関数
	/* End user code. Do not edit comment generated here */	
ļ		

注. 共通 main となる関数名は、ユーザ任意の名称で構いませんが、「6.3.2 の(4) 共通 main ファイルの作成」 で作成する共通 main の関数名と同じ名称にしてください。



6.3.2 RZ/T1 グループ 初期設定サンプルプログラムへの取り込み

コード生成ツールで作成した生成コードを EWARM 版の初期設定サンプルプログラム環境に組み込みます。

サンプル例として「6.3.1 コード生成ツールを使用したコードの作成について」で作成したコードを使用した組み込みを以下に示します。

備考:以降の説明では、初期設定サンプルプログラム Rev1.30 をもとに行数などの説明をしています。組み込みをさ れる場合は、その際の最新の初期設定サンプルプログラムを利用してください。

(1) コード生成ツールで作成した環境の移動(コピー)

初期設定サンプルプログラムのプロジェクトファイル内にあるソースファイル格納フォルダ (src) に 「6.3.1 の(2) 新規プロジェクトを作成」で作成したコード生成ツールのプロジェクトフォルダごと移 動(コピー)します。 サンプル例)

16 ビットバスブートモード用フォルダ名:cg_src_nor

SPI ブートモード用フォルダ名:cg_src_serial



注. コード生成ツールが出力したコードを格納するフォルダ名が(cg_src)で固定のため、16 ビットバスブート モード版(cg_src_nor)と SPI ブートモード版(cg_src_serial)をそれぞれ別々にサンプルプログラムへ取り込め るように異なるプロジェクト名で作成してください。 コード生成されたそれぞれのプロジェクトフォルダ内には、(cg_src)フォルダ とワークスペースファイル (.cgp)が格納されています。

【補足事項】e²studio 環境にコード生成ツールで作成したコードを組み込む際は、各プロジェクトの inc フォルダにある iodefine.h をプロジェクトフォルダの直下に移動してください。詳細は、「6.1.1 プロ ジェクト設定」の表 6.2 コード生成ツールで作成したコードを組み込んだ初期設定サンプルプログラ ムのフォルダ構成 (2/2) を参照ください。

RENESAS

(2) 初期設定サンプルプログラムにあるローダプログラムの編集

- 初期設定サンプルプログラムで使用されている、クロック発振回路設定、バスステートコントローラ設定、SPI マルチ I/O バスコントローラ設定は、「6.3.1 の(4) クロック、バス機能の設定」からコード 生成ツールで作成した設定に置き換えて使用します。そのため、本サンプルプログラムでは、ローダプ ログラムで指定しているこれらの設定を無効化(コメントアウト)します。
 - 初期設定サンプルプログラムにあるプロジェクトフォルダ (RZ_T1_init_boot) 内、ソースファイル格納 フォルダ (src) の common フォルダ以下にある、loader_init2.c ファイルを編集します。
 編集箇所は、ローダプログラム (loader init2 関数) に 2 箇所あります。(111 行目、119 行目)

	/* Set CPU clock and LOCO clock */	
11	cpg_init();	←クロック発振回路の初期化設定をコメントアウト ^{注1}
	: 中略	
	/* Initialize the bus settings */	
//	<pre>bus_init();</pre>	←バスコントローラ(BSC, SPIBSC)の初期化設定をコメントアウト注2

- 注 1. ローダプログラムでクロック、バスの高速化を図りたい場合は、コメントアウトせずに各設定関数をコール することも可能です。ただし、本サンプルプログラムでは以降のコード生成ツールによる初期化処理でク ロック、バスの再設定がされるため注意が必要です。詳細については、「9.の(1) bus_init() 関数を使用した 場合の注意事項」を参照してください。
- 注 2. bus_init() 関数では、シリアルフラッシュメモリの設定を Single I/O モードから Quad I/O モードに変更しま す。一方、本手順でコード生成した場合、シリアルフラッシュメモリの設定は Signle I/O モードを前提とし たリードコマンド (FAST READ4B)を用いる設定です。ローダプログラムでバス設定を行う目的で bus_init() 関数を使用する場合は Quad I/O モードに設定をするなど注意が必要となります。詳細は、「9. (1) bus_init() 関数を使用した場合の注意事項」を参照してください。
- (3) エラーコントロールモジュール (ECM)の初期設定関数をローダプログラムへ移動 初期設定サンプルプログラムでは、main処理中に ECM の設定を行っています。本プログラムでは、 ユーザが作成する共通 main と初期設定サンプルプログラムの main を置き換えます。 そのため初期設定で実施している ERROROUT 端子の初期化処理を行う ECM の設定 (ecm_init)は、 ローダプログラムへ移動します。
 - init_main.c から loader_init2.c へ ecm_init 関数を移植します。(DS-5 の場合は、cpu_init.c) init_main.c ファイルは、初期設定サンプルプログラムのプロジェクトファイル内にあるソースファイ ル格納フォルダ (src) の sample フォルダ以下にあります。
 ECM の拡張疑似エラー 35 の設定は不要なため削除します。
 同様に、loader_init2.c 内の ECM 拡張疑似エラー処理も削除します。

init_main.c ファイルの以下行を loader_init2.c ファイルヘコピーします。

コピー元: init_main.c ファイル

```
    ①関数定義 86 行目:
```

void ecm_init(void);

② main 関数内にある ecm_init 関数コール 117, 118 行目:

```
/* Initialize the ECM function */
```

```
ecm_init();
```



• ③ ecm_init 関数 177 行目~ 202 行目

コピー先: loader init2.c ファイル

• Private variables and functions ヘコピーした①関数定義を 98 行目へ追加

• loader_init2 処理内の set_low_vec 関数コール- _main 関数コール間へ

```
② ecm_init 関数コールをコピーして 129 行目へ追加
```

注 1. ecm_init 関数内の処理は、ATCM に配置されています。このため、直前で実行している R_ATCM_WaitSet() による ATCM のウエイト設定が確実に反映されるよう DMB 命令を追加しています。



ファイルの行末に③ ecm_init 関数をコピーして追加(354 行目~ 359 行目の拡張疑似エラー設定を削除)

```
End of function copy_4byte
* Function Name: ecm_init
~中略~
←ecm_init関数を追加
void ecm_init(void)
{
                    ←使用しないresult定義を削除
 volatile uint8 t result;
 /* Initialize ECM function */
 R ECM Init();
 /* Set extended pseudo error 35 */
 /* Enables internal reset configuration */
 result = R_ECM_Write_Reg32(ECM_COMMON, &(ECM.ECMIRCFG1.LONG), 0x00000004);
}
End of function ecm init
/* End of File */
```



• loader_init2 処理内の reset_check 関数の ECM 拡張疑似エラー処理を削除

```
* Function Name : reset check
* Description : Check the reset source and execute the each sequence.
          When error source number 35 is generated, set P77 pin to High.
* Arguments : none
* Return Value : none
void reset check(void)
{
                              ←使用しないresult定義を削除
  volatile uint8_t result;
  volatile uint32_t dummy;
                               ←使用しないdummy定義を削除
  /* Check the reset status flag and execute the each sequence */
  if (RST SOURCE ECM == SYSTEM.RSTSR0.LONG) // ECM reset is generated
  {
     /* Clear reset status flag */
     R RST WriteEnable(); // Enable writing to the RSTSR0 register
     SYSTEM.RSTSR0.LONG = 0x00000000; // Clear reset factor flag
     R_RST_WriteDisable(); // Disable writing to the RSTSR0 register
     /* Check the ECM error source */ ←以下のif文とif文内を削除
     if (1 == ECMM.ECMMESSTR1.BIT.ECMMSSE102) // Error source number 35 is generated
     {
     ~中略~
     }
  ~以下略~
End of function reset_check
}
```



(4) 共通 main ファイルの作成

コード生成ツールが作成したユーザアプリケーションプログラムは、16ビットバスブートモード版、 SPIブートモード版で異なるファイルを使用しています。 16 ビットバスブートモード版、SPI ブートモード版の共通となる main 処理を共通 main として作成しま す。 本サンプルプログラムでは、共通の main 関数のため、init main.c ファイルと同じ初期設定サンプルプ ログラムのプロジェクトファイル内にあるソースファイル格納フォルダ (src)の sample フォルダに作成 します。 ファイル名称については、ユーザ任意のファイル名称で構いません。本サンプルプログラムでは、作成 する関数名をコード生成ツールが作成したユーザアプリケーションプログラムからコールするため、以 下の名称としています。 サンプル例) 作成する共通の main 関数: user_app_main 作成ファイル: user main.c 以下は、本サンプルプログラムの user main.c ファイルを参考にして作成してください。 Includes <System Includes> , "Project Includes" ←CMT0関数用インクルード定義 #include "r_cg_cmt.h" #include "iodefine.h" #include "r system.h" ~中略~ Private variables and functions void user_app_main(void); ←user_app_mainの関数宣言 * Outline : user main processing * Function Name: user_app_main ~中略~ void user_app_main (void) ←user_app_mainの関数 { R CMT0 Start(); ←サンプル使用CMT0の開始 } End of function main /* End of File */



(5) コード生成ツールで作成したソースコードのコンパイル対象設定 EWARM版の初期設定サンプルプルグラムのコンパイル対象にコード生成ツールで作成したソースコー ドを以下のように設定します。

【プロジェクトコネクションの追加】

- ツールバーの[プロジェクト]メニューから[プロジェクトコネクションの追加]を選択します。
 [プロジェクトコネクションの追加]のダイアログが表示されます。
 使用するコネクションに "IAR Project Connection"を選択し "OK" ボタンを押します。
- 画像⑤-1 プロジェクトコネクションの追加



 [IAR プロジェクト接続ファイルの選択] ダイアログが表示されます。プロジェクト・コネクション・ファイル (.ipcf) を選択して [開く] ボタンをクリックします。 プロジェクト・コネクション・ファイルには、ソースファイルの登録情報が含まれています。 ここで選択するプロジェクト・コネクション・ファイルは、コード生成ツールで作成した以下の フォルダ (推奨) 内にコード生成すると作成されます。 参照先のフォルダサンプル例)
 16 ビットバスブートモード版フォルダ名: cg_src_nor SPI ブートモード版フォルダ名: cg_src_serial 完了すると以下のようにコンパイル対象となるファイルが追加されます。



• 画像 5-2 コンパイル対象となるファイルの追加例



備考: DS-5、e²studio では、コード生成ツールが作成したフォルダにファイルを追加すると自動でプロジェクトが ファイルを認識します。

- (6) 初期設定で使用している main 関数のコンパイル対象の除外設定
 - ワークスペースにある init_main ファイルを選択して右クリック
 [削除]を選択してコンパイル対象の除外設定



(7) 共通 main 関数のコンパイル対象設定

I

• init フォルダにコンパイル対象となる user_main.c を追加

Həli common	メイカ(M)						
→ 🕀 💽 bus_init_s	コンパイル(C)			指定フォルダから	user_main	.cを選択	· 🔟 😢
⊢⊞ [] loader_pai	すべてを再ビルド(B)			名前	更新日時	種類	ť
⊞ 🖸 spibsc_fla	クリーン(L)			init_main.c	2016/12/2	6 21:26 C ファー	イル
HE C spibsc_fla	C-STAT静的解析(C)	•		user_main.c	2016/11/2	5 19:39 C ファー	1)1
→⊞ C spibsc_ios	ビルドを停止(S)						
E exit.c	追加(A)	-	ファイルの追加(F)				
-⊞ Cloader_init2	削除(∨)		グループの追加(G)				
-⊞ Cr_atcm_init.c	名前の変更						
	バージョン管理システム(Y)	•					
⊢⊞ C]rjcujinit.c	ファイルの場所を聞く			< [m		
H 🖸 🖸 r_ram_init.c	ファイルのプロパティ(P)			ファイル名(N): user_main.c	c •	ソースファイル(*.c;*.cpp;* •
□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □	アクティブに設定(F)						التعليد حينات
- Cinit -	· · · · · · · · · · · · · · · · · · ·					₩1<(0)	+1700
U L C user_main.c	•						
- 🕀 🗀 Output							
iodefine.h							



٦

(8) 新規追加したコード生成ツール出力の生成コードで使用するインクルードパスを設定

 ツールバーにあるプロジェクトからオプションを選択し表示されたオプション設定より [C/C++ コン パイラ] → [プリプロセッサ] を選択 追加インクルードディレクトリ設定にコード生成ツールで作成したフォルダを以下のサンプル例の ようにワークファイルパスで追加 サンプル例)

PROJ_DIR\$\src\cg_src_serial\cg_src

Iジェクト(P) I-jet/JTAGjet(I) ツール	(T) ウィンドウ(W)	ノード"RZ_T1_init_seri	al_boot"の	オプション
ファイルの追加(F) グループの追加(G) ファイルリストのインポート(I) プロジェクトコネクションの追加(P) ビルド構成の編集(T)		カテゴリ: 一般オプション 静的解析 ランタイム解析	•	□ 補数ファイルのコンパイル □ 本使用パブリックを破壊
削除(∨)		C/C++コンパイラ		言語 1 言語 2 コード 最適化 出力 リスト フリフロゼッサ 診断 MISF
新規プロジェクトの作成(N) 既存プロジェクトの追加(E)		アセンフラ 出力コンバータ カスタムビルド	-	□ 標準のインクルードディレクトリを無限(1) 追加インクルードディレクトリ(A)(1行に1ディレクトリ)
オプション(0)	Alt+F7	リンカ	-	\$PROJ_DIR\$¥inc \$PROJ_DIR\$¥src¥cg_src_serial¥cg_src
バージョン管理システム(Y)	+	デバッガ シミュレータ		-
メイク(M) コンパイル(C) すべてを再ピルド(B) クリーン(L) バッチピルド(A)	F7 Ctrl+F7 F8	Angel CADI CMSIS DAP GDBサーバ IAR ROMモニタ I-jet/JTAGjet		フリインクルードファイル(R): シンボル定義(D)(1行に1シンボル) □ ファイルへのプリプロセッサ出力(P) □ コメントの(保持(C) □ 朝ineディレクティブ生成(G)
C-STAT静的解析(C)	Þ	J-Link/J-Trace TI Stellaris		
ビルドを停止(S)	Ctrl+Break	Macraigor	*	OK キャンセノ
ダウンロードしてデバッグ(D) ダウンロードせずにデバッグ(H) 実行中のターゲットにアタッチ(U) メイク後デバッガを再起動(K) デバッガを再起動(R) ダウンロード(W)	Ctrl+D Ctrl+R Ctrl+Shift+R			
SFRの設定(U)				
デバイス記述ファイルを開く	۲			



Γ

(9) すべてを再ビルド

ツールバーにあるプロジェクトから以下のように[すべてを再ビルド]を選択します。
 選択すると再ビルドが開始されます。エラーがないことを確認ください。

ファイルの追加(F)	
グループの追加(G)	
ファイルリストのインポート(I)	
プロジェクトコネクションの追加(P)	
ビルド構成の編集(T)	
削除(V)	
新規プロジェクトの作成(N)	
既存プロジェクトの追加(E)	
オプション(0)	Alt+F7
バージョン管理システム(Y)	×
メイク(M)	F7
コンパイル(C)	Ctrl+F7
すべてを再ビルド(B)	
クリーン(L)	
バッチビルド <mark>(</mark> A)	F8
C-STAT静的解析(C)	•
ビルドを停止(S)	Ctrl+Break
ダウンロードしてデバッグ(D)	Ctrl+D
ダウンロードせずにデバッグ(H)	
実行中のターゲットにアタッチ(U)	
メイク後デバッガを再起動(K)	Ctrl+R
デバッガを再起動(R)	Ctrl+Shift+R
ダウンロード(W)	•
SFRの設定(U)	
デバイフ記述ファイルを問く	۰.
ノハイス記述ノアイルを開く	



(10)ダウンロードしてデバッグを実行

 ツールバーにあるプロジェクトから以下のように [ダウンロードしてデバッグ]を選択します。
 エミュレータ接続後、専用フラッシュダウンローダ機能により外付けシリアルフラッシュメモリへ プログラムの書き込みが行われた後に、デバッグが開始されます。





6.4 固定幅整数一覧

表 6.6 にサンプルプログラムで使用する固定幅整数を示します。

表6.6 サンプルプログラムで使用する固定幅整数

シンボル	内容
int8_t	8ビット整数、符号あり(標準ライブラリにて定義)
int16_t	16ビット整数、符号あり(標準ライブラリにて定義)
int32_t	32ビット整数、符号あり(標準ライブラリにて定義)
uint8_t	8ビット整数、符号なし(標準ライブラリにて定義)
uint16_t	16ビット整数、符号なし(標準ライブラリにて定義)
uint32_t	32ビット整数、符号なし(標準ライブラリにて定義)

6.5 関数一覧

初期設定編のサンプルプログラム内で使用している関数については、RZ/T1 グループ初期設定アプリケーションノートを参照してください。本一覧については、初期設定編に追加する関数について表 6.7 に示します。

表6.7 関数一覧

	関数名
user_app_main	



6.6 フローチャート

6.6.1 ローダプログラム処理

図 6.3 にローダプログラム処理のフローチャートを示します。



図 6.3 ローダプログラム処理

ローダプログラム部のフロー詳細については、RZ/T1 グループ初期設定アプリケーションノートを参照してください。

6.6.2 コード生成ツール作成アプリケーションプログラム処理

図 6.4 にコード生成ツール作成アプリケーションプログラムのフローチャートを示します。



図 6.4 コード生成ツール作成アプリケーションプログラム

RENESAS

6.6.3 共通 main 処理

図 6.5 に共通 main のフローチャートを示します。



図 6.5 共通 main 処理

6.6.4 ユーザ使用タイマ(CMT0)割り込み処理

図 6.6 にユーザ使用タイマ(CMT0)割り込み処理のフローチャートを示します。



図 6.6 ユーザ使用タイマ(CMT0)割り込み処理

7. サンプルプログラム

サンプルプログラムは、ルネサスエレクトロニクスホームページから入手してください。



8. 参考ドキュメント

ユーザーズマニュアル:ハードウェア
 RZ/T1 グループ ユーザーズマニュアル ハードウェア編
 (最新版をルネサス エレクトロニクスホームページから入手してください。)

RZ/T1 評価ボード RTK7910018C00000BE ユーザーズマニュアル (最新版をルネサス エレクトロニクスホームページから入手してください。)

- テクニカルアップデート/テクニカルニュース
 (最新の情報をルネサスエレクトロニクスホームページから入手してください。)
- ユーザーズマニュアル:開発環境
 IAR 統合開発環境(IAR Embedded Workbench[®] for ARM)に関しては、最新版を IAR ホームページから入手してください。
 ARM 統合開発環境(Development Studio 5TM)に関しては、最新版を ARM ホームページから入手してください。
 ルネサス エレクトロニクス統合開発環境(e² studio)に関しては、最新版をルネサスエレクトロニクスホームページから入手してください。



9. 注意事項

(1) bus_init() 関数を使用した場合の注意事項

本サンプルプログラムではベースとなる初期設定編サンプルプログラムの bus_init() 関数をコメントアウトして使用していません。ローダプログラムにてバス設定を行うため、bus_init() 関数を使用することも可能ですが、その際は以下の点にご注意ください。

bus_init() 関数では、シリアルフラッシュの設定を Single I/O モードから Quad I/O モードに設定していま す。一方、本サンプルプログラムでは Single I/O モードを前提とした設定を行っているため、シリアルフ ラッシュのリード処理が正常に行えません。bus_init() 関数を使用される場合は、「6.3.1 の(4) クロック、 バス機能の設定」の【SPI マルチ I/O バスコントローラ設定】手順にてコマンドに 0xEC(4READ4B)を設 定するなど Quad I/O モードに対応した設定となるよう変更を行ってください。



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/

お間合せ先

http://japan.renesas.com/contact/



コケミエ	ᆕᄀᄼᆂ
「以言」	記述水

サンプルプログラムへのコード生成ツール適用ガイド アプリケーションノート

David	20 / -		改訂内容
Rev.	発行日	ページ	ポイント
1.00	2017.06.30	—	初版発行
1.10	2018.04.11	2. 動作環境	
		4	表2.1 動作環境 コード生成ツール:RENESAS製AP4のバージョンを変更、注を変更
		3. 関連アプリ	ケーションノート
		5	アプリケーションノートのドキュメント番号を追加
6. ソフトウェア説明 13 6.1.3 例外処理ベクタ・		6. ソフトウェ	:ア説明
		13	6.1.3 例外処理ベクタテーブル 34バイト領域の番地を変更
		13	表6.4 例外処理ベクタテーブル 注:番地を変更
		23	6.3.1 コード生成ツールを使用したコードの作成について コード生成ツールの説明を変更
		24	「画面⑤ SPIマルチI/Oバスコントローラ設定(一般設定)」の画面を変更、【注意点】を変更
		25	「画面⑤ SPIマルチI/Oバスコントローラ設定(設定)」の画面を変更
		27	「サンプル例) 端子配置表から端子機能の各設定画像と説明(3/3)」の画面を変更
		29	6.3.1, 【バスステートコントローラのAPI関数設定】の説明を変更、コードの記述を変更
		31	6.3.2 RZ/T1グループ 初期設定サンプルプログラムへの取り込み 備考:初期設定サンプルプログラムのRev.を変更
		32	6.3.2, (2) 初期設定サンプルプログラムにあるローダプログラムの編集 注1を変更、注2を追加
		33	6.3.2, (3): In the loader_init2 処理: set_low_vec(); と /* Initialize the ECM function */ の間に、 ATCMのウエイト設定待ちのための関数を追加、注1を追加
		9. 注意事項	
		49	9. 注意事項 追加

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。

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生じた損害(お客様 または第二者によって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生じた損害(お客様
2	または第二有い970に生した損害も己がまり。以下回しでり。)に関し、日社は、一切ての負性を良いません。 ※注制ローナ次約に記載された制ロゴーク 図 まーゴロゲニノ フェイルブル ウロ回欧回答の時期のは用いお用して発生した第二者の結婚物 茶が持たの所の
Ζ.	当社要加、本具科に記載された要加ナーダ、図、表、フログラム、アルコリスム、応用凹路例等の情報の使用に起因して先生した第二有の特許権、者作権での他の 知的財産接に対する保定またはこれでに関する公会について、当社は、何にの保証を行うたのではなく、また表にを負うたのではなりません
3	知的別産権に対する反告よたはこれらに関する初ずについて、当社は、何らの床証を行うものではなく、よた員社を負うものではめりよせん。 当社は、太咨料に基づき当社または第三者の姓転提。妻佐提子の他の知め財産族を何ら転送するまのでけありません。
4	当社園、平文社に至って当社会にはが当日の時間に、有性にての他の知道が連進と呼ら時間があるのではのがません。
	バースエンジニアリング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)、交通制御(信号)、大規模通信機器、
	金融端末基幹システム、各種安全制御装置等
	当社製品は、データシート等により高信頼性、Harsh environment向け製品と定義しているものを除き、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(宇宙機器と、海底中継器、原子力制
	御システム、航空機制御システム、プラント基幹システム、軍事機器等)に使用されることを意図しておらず、これらの用途に使用することは想定していませ
	ん。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
6.	当社製品をご使用の際は、最新の製品情報(データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使
	用上の一般的な注意事項」等)をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指
_	定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合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同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、ケーダンート寺において高信頼性、Harsh environment向け製品と定義しているものを除さ、耐放射線設計を行っておりません。仮に当社 制用の地陸士には評価化がたじた根本でもってす。「真真市地」」(※市地スの地社会的場実等またじさせないとう。か安排の声だにないて、同長歌社、孤雄社等歌
	製血のの障害だは読動作が生した場合でのつても、大身争の、火火争のての他社会的損告等を生しさせないよう、お各体の負任において、尤女政計、巡洗対束政 計 認動た性は恐社体の中全恐社なとガェージング加速使、な友性の準要・システントレーズの世界視証を行ってください、特に マイコンパフトウェアけ 単独
	n、感動F的工品はその文主品は35551 シングル理学、3547%の版語・フステムとしての山利休証を打りててたさい。特に、マキュングノージェアは、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RollS
0.	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。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.	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。当社製品および技術を輸
	出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い
	必要な手続きを行ってください。
10	. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
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。
12	. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。
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直接的、間接的に支配する会
	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。

(Rev.4.0-1 2017.11)

RENESAS

ルネサスエレクトロニクス株式会社

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 〒135-0061 東京都江東区豊洲3-2-24 (豊洲フォレシア)

■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:https://www.renesas.com/contact/

■営業お問合せ窓口

© 2018 Renesas Electronics Corporation. All rights reserved. Colophon 6.0