

RE01 1500KB, 256KB グループ

R01AN4660JJ0103

Rev.1.03

2020.6.19

CMSIS パッケージを用いた開発スタートアップガイド

要旨

本アプリケーションノートでは、RE01 1500KB, 256KB グループ CMSIS Driver Package を用いたソフトウェア開発手順について説明します。本ドキュメントを参照することで、ドライバを使うための基本的な設定手順および本パッケージ上でドライバがサポートしていない周辺機能のコードを実装する方法を理解することができます。

CMSIS Driver Package は 1500KB グループ、256KB グループ毎に提供されます。本アプリケーションノートでは特に記述のない限り、1500KB グループと 256KB グループは同じ仕様であり、"1500KB" は "256KB" に読み替えてください。また、"本パッケージ" はご使用になるいずれかのグループの CMSIS パッケージに読み替えてください。

対象デバイス

- ・ RE01 1500KB グループ
- ・ RE01 256KB グループ

本アプリケーションノートを他のデバイスへ適用する場合、そのデバイスの仕様にあわせて変更し、十分な評価を行ってください。

まとめ

- ✓ RE01 1500KB, 256KB グループ CMSIS Driver Package は、RE01 1500KB, 256KB グループ のスタートアップコードやドライバをパッケージ化したものです。
- ✓ 2 章で本パッケージのプロジェクトを動かすことができるようになります。
- ✓ 3 章で本パッケージのコンポーネントの特徴がわかります。
- ✓ 4, 5 章で本パッケージのドライバの概要がわかります。
- ✓ 6 章で割り込みや端子などの基本機能が使えるようになります。
- ✓ 7 章でユーザプログラムを作成できるようになります。
- ✓ 8 章でプロジェクトを作成できるようになります。
- ✓ 9 章でデバッグができるようになります。

■用語集

用語	解説
CMSIS	Arm®社が規定しているソフトウェア・インターフェース規格 Cortex Microcontroller Software Interface Standard の略称
CMSIS-Driver	CMSIS で規格されたソフトウェア・インターフェースの周辺機能ドライバ
Device HAL	MCU ベンダ（本ドキュメントではルネサス）の独自仕様のドライバ
CMSIS-CORE	CMSIS で規格された MCU のスタートアップルーチン
R_CORE	CMSIS-CORE にルネサス独自の仕様を追加したもの
R_SYSTEM	ルネサス独自のクロック切り替えや割り込み制御を担当するドライバ
スタートアップ処理	MCU のリセット解除後 main 関数を実行するまでの起動処理のこと
アドレスマッピング	プログラムやデータが MCU 内部空間（アドレス）のどここの領域に割り当てられるかを意味する
コールバック関数	ドライバが特定のイベント発生時に呼び出すユーザが作成した関数
ICU	RE01 の割り込みコントローラユニットのこと
NVIC	Arm® Cortex®-M0+等に搭載されるネスト型ベクタ割り込みコントローラのこと Nested Vectored Interrupt Controller の略語
IRQ 番号	NVIC の外部割込み入力番号のこと
ユーザーズマニュアル	ルネサスが提供するデバイスのマニュアルのこと
EWARM	IAR 社が提供する統合開発環境（IDE） IAR Embedded Workbench® for Arm®
e ² studio	ルネサス社が提供する統合開発環境（IDE）

目次

1. パッケージの概要	6
1.1 CMSIS について	6
1.2 フォルダ構成	7
1.3 サポート機能	8
1.4 パッケージの特徴	9
1.5 動作確認環境	10
2. プロジェクトを動かしてみよう	11
2.1 EWARM 編	12
2.2 e ² studio 編	15
3. コンポーネント	18
3.1 CMSIS-CORE	18
3.1.1 サポートドライバ	18
3.1.2 R_CORE ドライバの主な役割	19
3.2 CMSIS-Driver	19
3.2.1 サポートドライバ	20
3.2.2 拡張機能	20
3.3 CMSIS-DSP	20
3.3.1 DSP ライブライ取り込み手順	20
3.4 HAL-Driver	27
3.4.1 サポートドライバ	27
3.4.2 共通機能ドライバ	28
3.4.3 周辺機能ドライバ	28
4. ドライバ仕様書	29
5. ドライバ基本概念	30
5.1 共通機能ドライバと周辺機能ドライバ	30
5.2 ドライバの構成	30
5.3 共通機能の設定	30
6. 基本機能	31
6.1 スタートアップ処理	31
6.1.1 動作開始時の端子設定	32
6.1.2 動作開始時のクロック・電力制御モードの設定	33
6.2 IO 電源ドメイン不定値伝搬抑止制御	35
6.2.1 対応電源ドメイン	35
6.2.2 ドライバ関数	37
6.3 割り込み制御	38
6.3.1 割り込みベクターテーブルとエントリ関数	38
6.3.2 IRQ 番号の割り当て	39
6.3.3 r_system_cfg.h の編集	40
6.3.4 ドライバ関数	41
6.4 クロック設定	43

6.4.1 クロック定義	43
6.4.2 ドライバ関数	43
6.5 端子設定	44
6.5.1 ドライバ関数	44
6.5.2 ドライバ関数の編集	48
6.6 プログラムの RAM 配置	50
6.6.1 RAM 配置セクションによる RAM 配置方法	51
6.6.2 強制インライン展開による RAM 配置方法	57
 7. ユーザプログラムを作ってみよう	58
7.1 ユーザプログラム作成準備	58
7.1.1 スタートアップ処理の準備	59
7.1.2 共通機能ドライバの準備	61
7.1.3 周辺機能ドライバの準備	64
7.2 ユーザプログラム作成	65
7.2.1 初期設定	67
7.2.2 周辺機能を制御	69
7.2.3 端子を制御	75
7.2.4 割り込みを制御	76
7.3 ユーザプログラム作成例	77
7.3.1 周辺機能ドライバを使用する場合の例（UART 通信）	77
7.3.2 周辺機能ドライバを使用しない場合の例（パルス出力）	79
 8. プロジェクトを作ってみよう	81
8.1 EWARM 編	81
8.1.1 ターゲットプロセッサの設定	82
8.1.2 リンカファイルの設定	82
8.1.3 インクルードディレクトリの設定	83
8.2 e ² studio 編	84
8.2.1 ツールチェーンの設定	84
8.2.2 リンカファイルの設定	85
8.2.3 インクルードパスの設定	86
 9. デバッグを使ってダウンロードしてみよう	87
9.1 デバッグ用 MTB 領域の確保	87
9.2 低消費電力モード使用時の注意点	87
9.2.1 ソフトウェアブレークを設定できない条件	87
9.3 EWARM 編	88
9.3.1 J-Link	88
9.3.2 I-Jet	91
9.4 e ² studio 編	92
9.4.1 J-Link	92
 10. 付録	96
 11. トラブルシューティング	97
11.1 Q：ビルドエラーが発生する	97

RE01 1500KB, 256KB グループ CMSIS パッケージを用いた開発スタートアップガイド

11.2 Q : ドライバ関数を実行すると HardFault Error が発生する	97
11.3 Q : ドライバ関数を実行しているが周辺機能が動作しない	98
11.4 Q : ドライバ関数の戻り値は正常だが、周辺機能の端子から期待した入出力が確認できない	98
11.5 Q : クロックや消費電力低減機能のレジスタに書き込んだが反映されない	98
11.6 Q : 周辺機能のレジスタに書き込んだが反映されない	99
11.7 Q : デバッガを使ってターゲットボードにダウンロードできない	99
11.8 Q : デバッガを接続したが動作しない	99
11.9 Q : 割り込みが発生しない	99
 12. RE01 1500KB グループと RE01 256KB グループ間でのプログラム移行時の注意事項	100
12.1 モード遷移例	100
12.1.1 遷移例 1 : ブーストモードと低リーク電流モード間の遷移	101
12.1.2 遷移例 2 : ノーマルモードとソフトウェアスタンバイモード間の遷移	102
12.1.3 遷移例 3 : High-Speed モードと Low-Speed モード間の遷移	103
 13. サンプルコード	104
 14. 参考ドキュメント	104
 改訂記録	105

1. パッケージの概要

1.1 CMSISについて

本パッケージには、図 1-1 CMSIS 全体像 で示す赤囲いのコンポーネントを同梱しています。

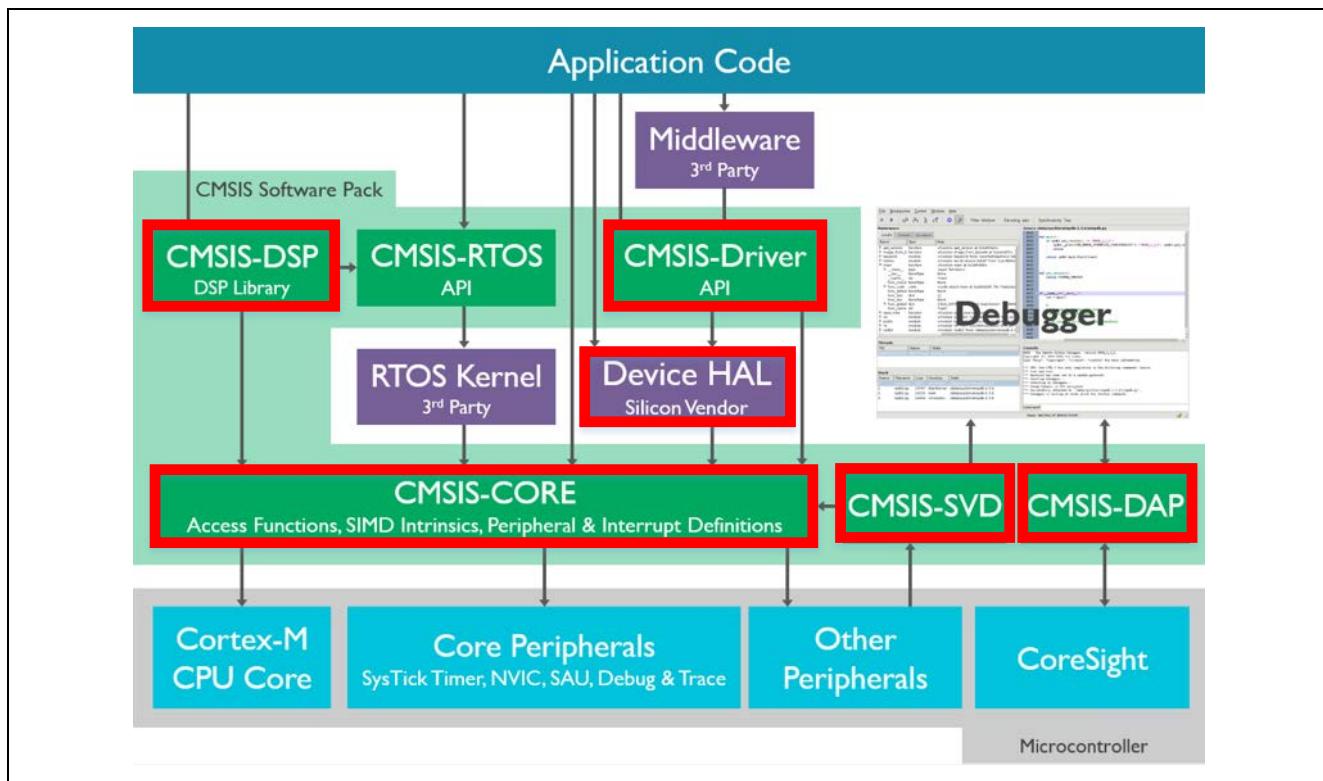


図 1-1 CMSIS 全体像

本パッケージに同梱されるコンポーネントと、ドライバの関係を図 1-2 に示します。

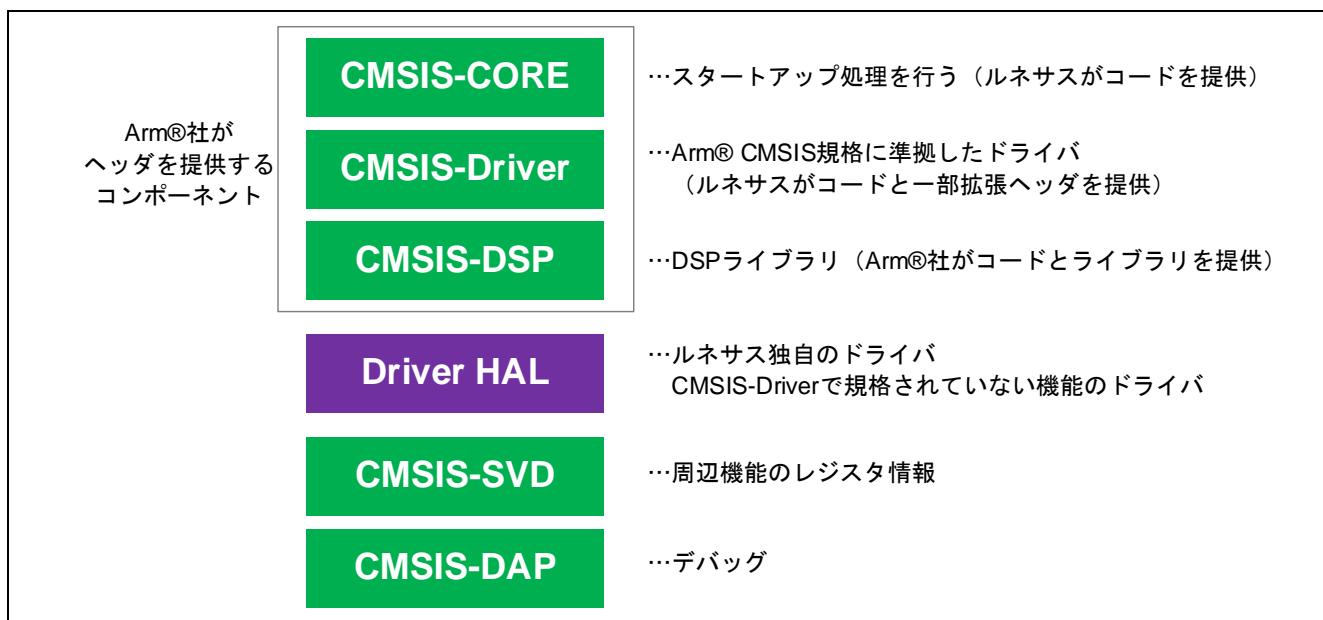


図 1-2 コンポーネントとドライバの関係

1.2 フォルダ構成

本パッケージのフォルダ構成を図 1-3 に示します。RE01 256KB グループは RE01 1500KB グループと同じ構成です。図内の"1500KB"を"256KB"に読み替えてください。

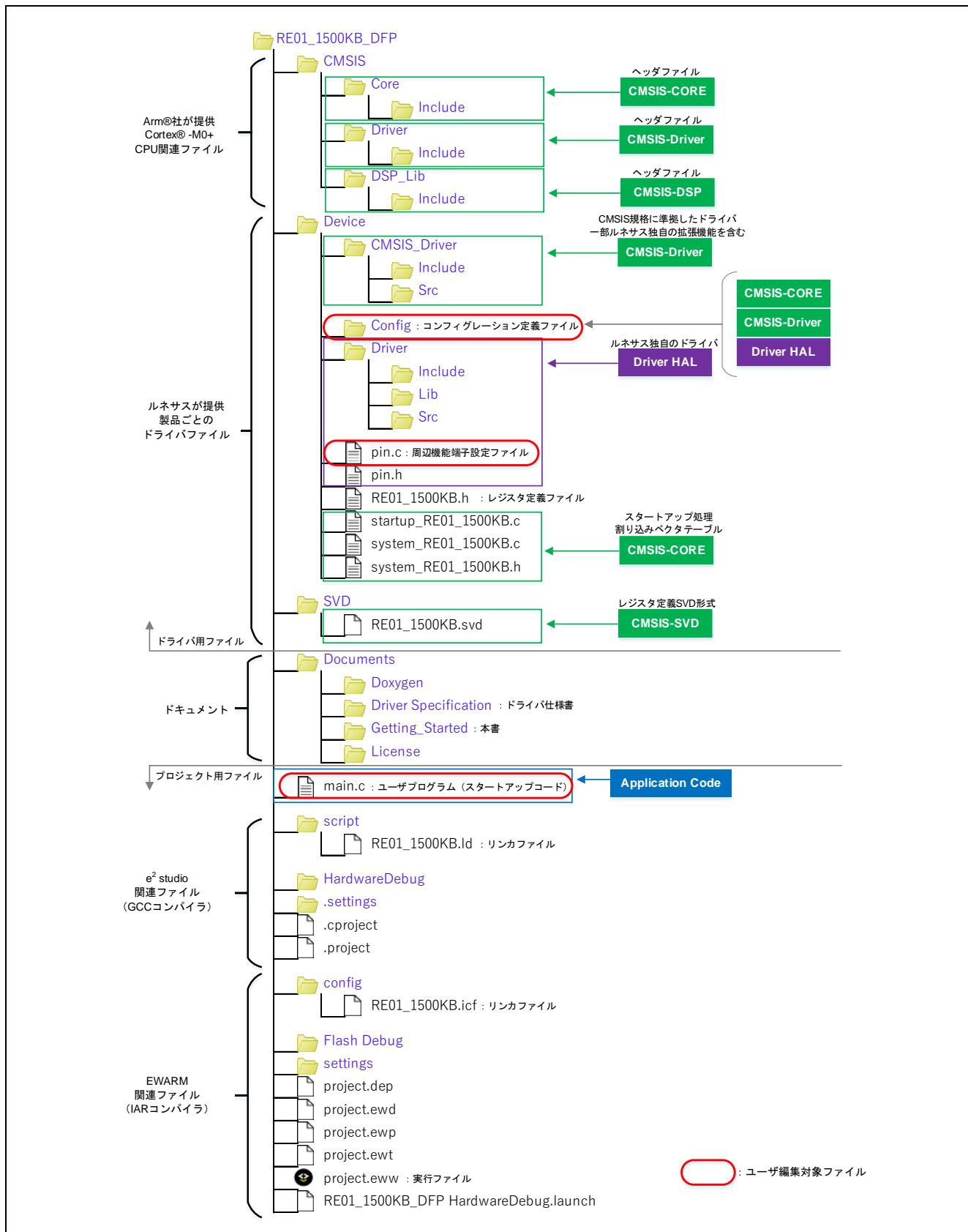


図 1-3 RE01 1500KB グループ CMSIS Driver Package フォルダ構成

1.3 サポート機能

本パッケージでサポートする機能を説明します。各ドライバの仕様は、4章で示す各ドライバ仕様書を参照してください。各機能の詳細は、対象デバイスの「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。

CMSIS-CORE

表 1-1 CMSIS-CORE のサポート機能

ドライバ名	サポート機能	関連ハードウェア機能
R_CORE	割り込みベクタテーブル	割り込みコントローラユニット (ICU)
	スタートアップ処理	クロック発生回路 消費電力低減機能

CMSIS-Driver

表 1-2 CMSIS-Driver のサポート機能

ドライバ名	サポート機能	関連ハードウェア機能
R_SPI	SPIシリアル通信	シリアルペリフェラルインタフェース (SPI)
R_I2C	I2Cシリアル通信	I2Cバスインタフェース (RIIC)
R_USART	UARTシリアル通信	シリアルコミュニケーションインタフェース (SClG, SClI)

Driver HAL

表 1-3 Driver HAL のサポート機能

ドライバ名	サポート機能	関連ハードウェア機能
R_SYSTEM	クロック設定	クロック発生回路
	電力制御モード	消費電力低減機能
	オプション設定	オプション設定メモリ
	割り込み制御	割り込みコントローラユニット (ICU) NVIC ^{注1}
	レジスタライトプロテクト	レジスタライトプロテクション
	プログラムの RAM 展開	(ソフトウェアで対応)
	ハードウェアリソースロック	(ソフトウェアで対応)
R_LPM	IO 電源ドメイン制御 消費電力低減機能 • モジュールストップ • 内蔵フラッシュメモリの電源遮断 など	消費電力低減機能
R_PIN	端子設定	マルチファンクションピンコントローラ
R_ADC	14 ビット AD 変換	14 ビット A/D コンバータ
R_DMAC	DMA 転送	DMA コントローラ (DMAC)
R_DTC	DTC 転送	データトランസファコントローラ (DTC)
R_FLASH	内蔵フラッシュメモリ	フラッシュメモリ
R_GDT	2D グラフィックデータ	2D グラフィックデータ変換回路 (GDT)
R_SMIP	シリアル MIP 液晶	(以下の機能を使用しソフトウェアで対応) • R_SPI ドライバ • 非同期汎用タイマ (AGT)
R_PMI	パラレル MIP 液晶	MIP 液晶コントローラ (MLCD)
R_USB ^{注2}	USB2.0 FS ホスト／ファンクション	USB2.0 FS ホスト／ファンクションモジュール (USB)

【注 1】 Cortex™-M0+ Technical Reference Manual (ARM DDI 0484C) の NVIC の章を参照してください。

【注 2】 RE01 256KB グループ CMSIS パッケージはサポートしていません。

1.4 パッケージの特徴

本パッケージの主な特徴を説明します。

- **スタートアップ処理**

本パッケージは、リセット解除後、main 関数を実行するまでのスタートアップ処理を用意しています。 詳細は、6.1 スタートアップ処理 を参照してください。

- **IO 電源ドメイン不定値伝搬抑制機能**

本デバイスは、複数の IO 電源ドメインを持ちます。リセット解除後は、一部ドメインを除きすべての IO 電源ドメインの不定値伝搬抑制機能が有効です。電源が供給されている IO 電源ドメインの不定値伝搬抑制機能を無効にする必要があります。

詳細は、6.2 IO 電源ドメイン不定値伝搬抑制御 を参照してください。

- **ユーザプログラム作成前の準備**

本パッケージを使用する場合、ユーザはユーザプログラムの作成だけではなく、特定関数の作成や、ドライバのコンフィグレーション定義ヘッダおよびドライバ関数を編集するなどの準備を行う必要があります。

詳細は、6.3 割り込み制御、6.4 クロック設定、6.5 端子設定、7.1 ユーザプログラム作成準備 を参照してください。

- **プログラムの RAM 配置**

本デバイスは、内蔵フラッシュメモリの電源を遮断することで消費電力を低減することができます。内蔵フラッシュメモリの電源を遮断後、RAM に展開したプログラムで動作させる場合は、任意のプログラムを RAM に配置する必要があります。

詳細は 6.6 プログラムの RAM 配置 を参照してください。

1.5 動作確認環境

本パッケージの動作確認環境を以下に示します。

(1) RE01 1500KB グループ

表 1-4 IAR コンパイル環境

項目	内容	ベンダ
使用マイコン	R7F0E015D2CFB 144pin	Renesas
ターゲットボード	Evaluation Kit RE01 1500KB (型名 : RTK70E015DCxxxxBE)	Renesas
統合開発環境 (IDE)	EWARM V8.3 以降 (IAR Embedded Workbench® for Arm®)	IAR システムズ
コンパイラ	IAR V8.32 以降	IAR システムズ
デバッガ	I-Jet J-Link OB (ターゲットボードに搭載)	IAR システムズ SEGGER

表 1-5 GCC コンパイル環境

項目	内容	ベンダ
使用マイコン	R7F0E015D2CFB 144pin	Renesas
ターゲットボード	Evaluation Kit RE01 1500KB (型名 : RTK70E015DCxxxxBE)	Renesas
統合開発環境 (IDE)	e ² studio V.7 以降	Renesas
コンパイラ	GCC V.6 GNU 6-2017-q2-update	—
デバッガ	J-Link OB (ターゲットボードに搭載)	SEGGER

(2) RE01 256KB グループ

表 1-6 IAR コンパイル環境

項目	内容	ベンダ
使用マイコン	R7F0E01182CFP 100pin	Renesas
ターゲットボード	Evaluation Kit RE01 256KB Main Board (型名 : RTK70E0118CxxxxBJ)	Renesas
統合開発環境 (IDE)	EWARM V8.3 以降 (IAR Embedded Workbench® for Arm®)	IAR システムズ
コンパイラ	IAR V8.32 以降	IAR システムズ
デバッガ	I-Jet J-Link OB (ターゲットボードに搭載)	IAR システムズ SEGGER

表 1-7 GCC コンパイル環境

項目	内容	ベンダ
使用マイコン	R7F0E01182CFP 100pin	Renesas
ターゲットボード	Evaluation Kit RE01 256KB Main Board (型名 : RTK70E0118CxxxxBJ)	Renesas
統合開発環境 (IDE)	e ² studio 2020-07	Renesas
コンパイラ	GCC V.6 GNU 6-2017-q2-update	—
デバッガ	J-Link OB (ターゲットボードに搭載)	SEGGER

2. プロジェクトを動かしてみよう

本パッケージに同梱されているプログラムの動作フローを説明します。

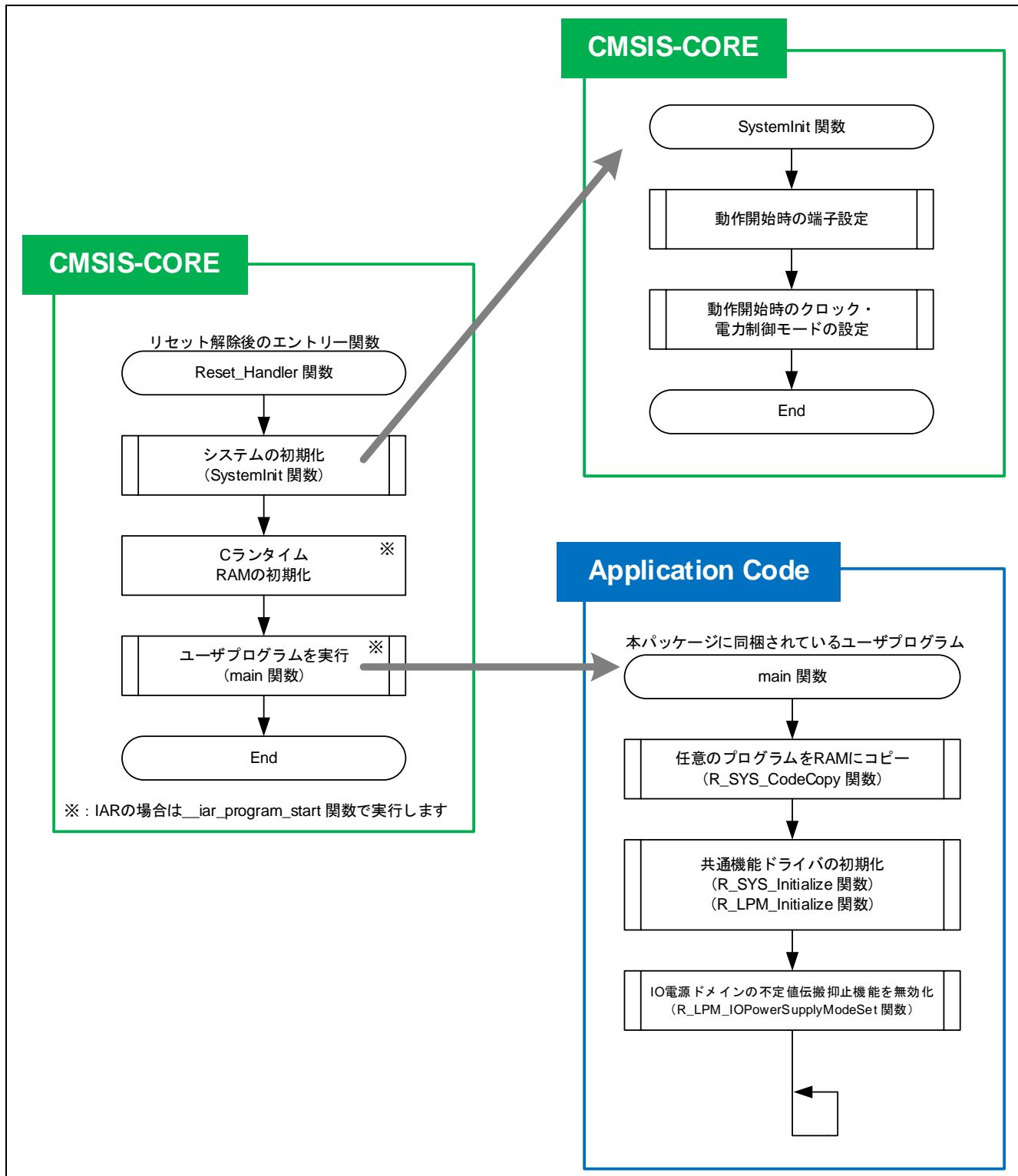


図 2-1 プログラムの動作フロー

2.1 EWARM 編

本パッケージに同梱されているプロジェクトを動かす方法を説明します。

Evaluation Kit ボードを使用する場合の設定例を以下に示します。

① プロジェクトを起動

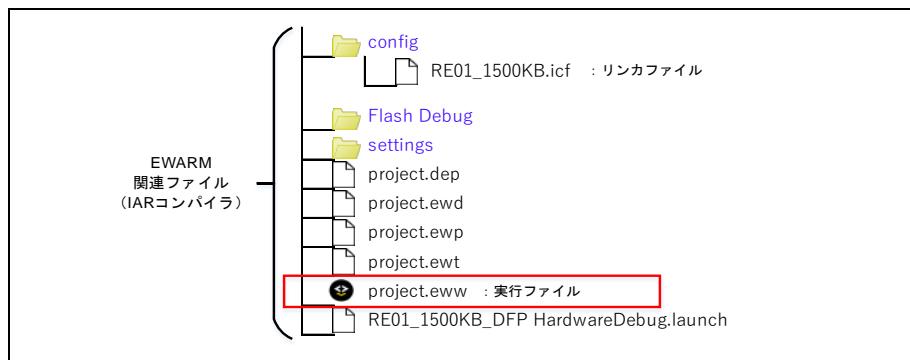


図 2-2 プロジェクト起動ファイル

② コンパイル

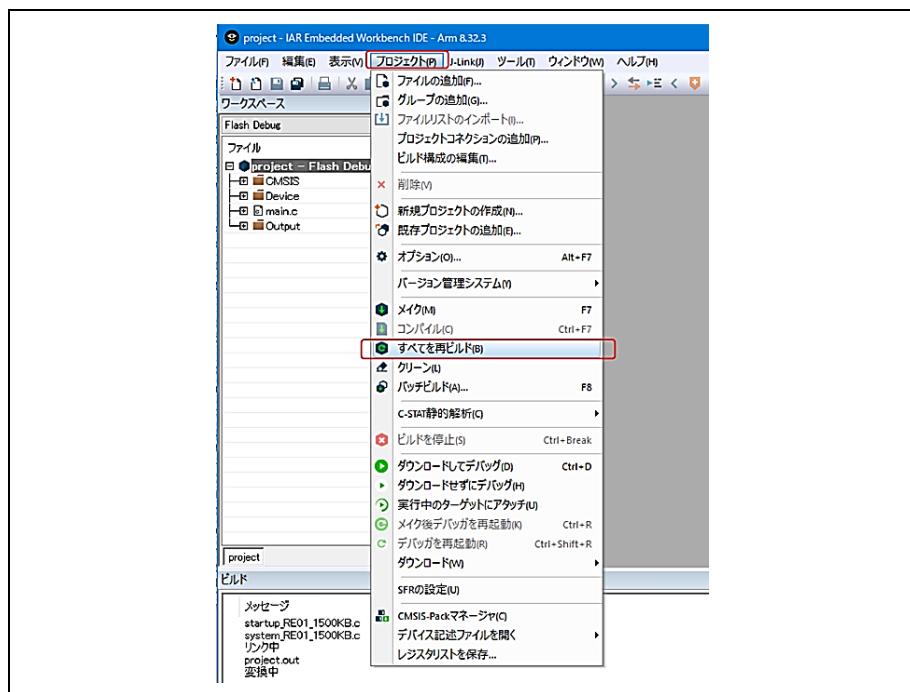


図 2-3 コンパイルメニュー

③ J-Link の選択

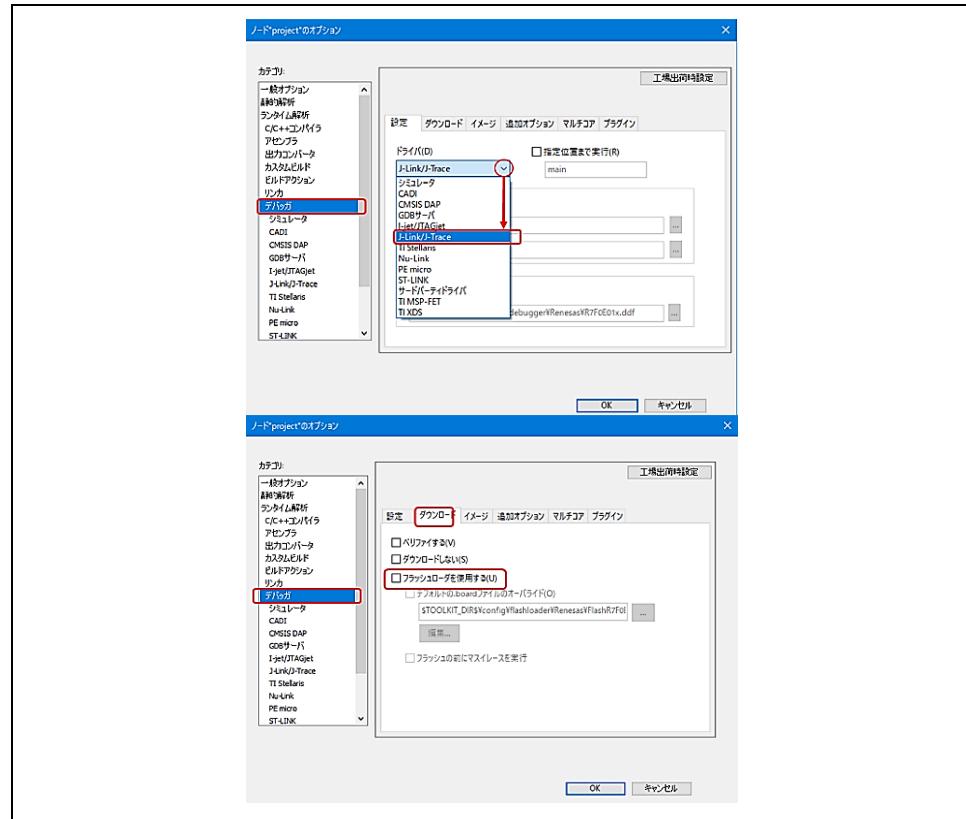


図 2-4 J-Link の選択手順

④ J-Link の設定

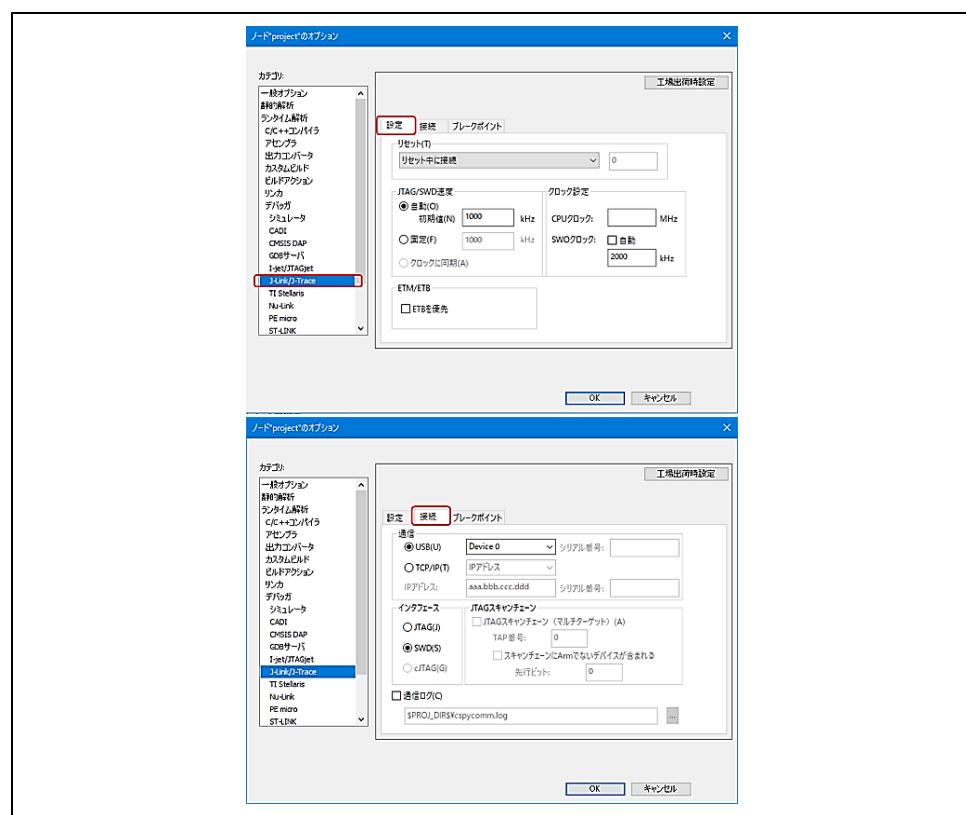


図 2-5 J-Link の設定手順

⑤ ボードと接続

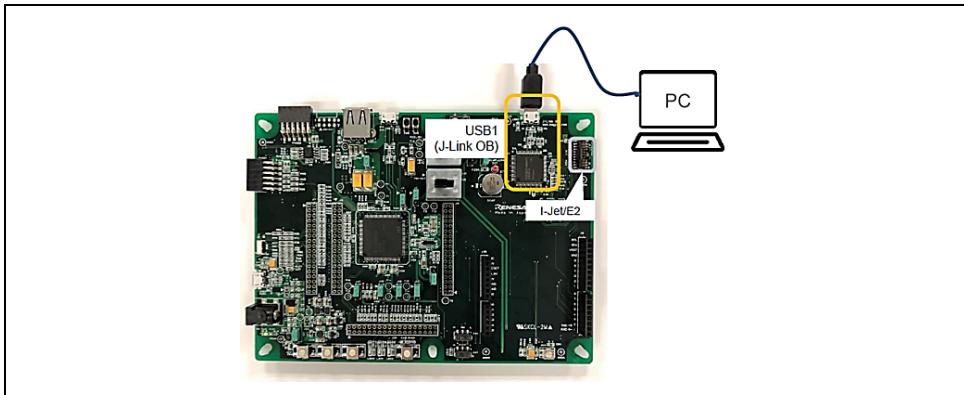


図 2-6 ボード接続例

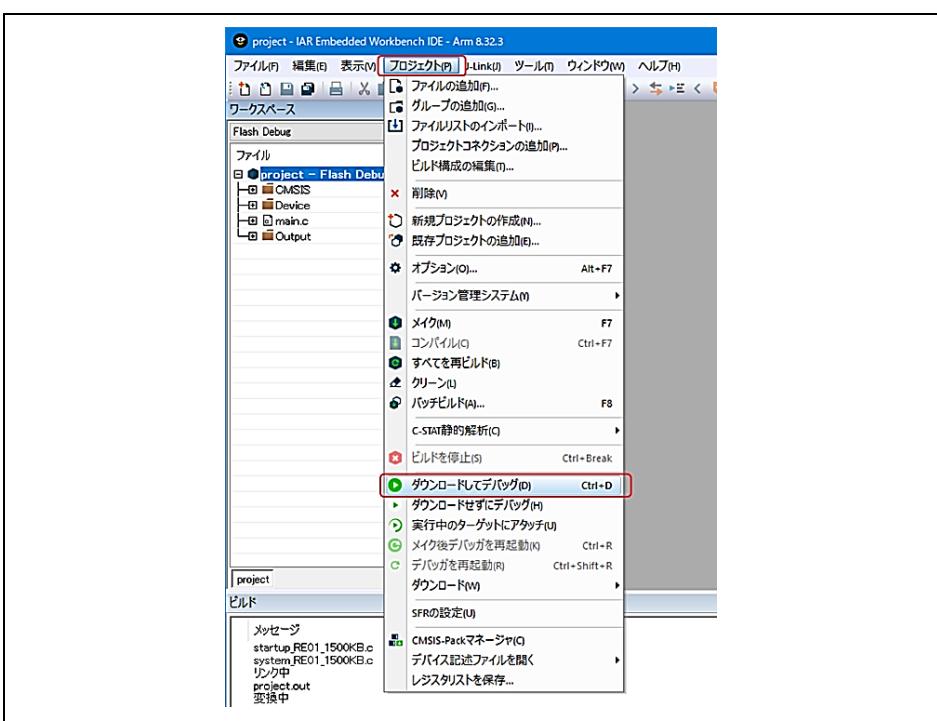
⑥ ダウンロードして
デバッガを起動

図 2-7 デバッガ起動メニュー

2.2 e² studio 編

本パッケージに同梱されているプロジェクトを動かす方法を説明します。

Evaluation Kit ボードを使用する場合の設定例を以下に示します。

① e² studio を起動



図 2-8 e² studio 起動ファイル

② インポート

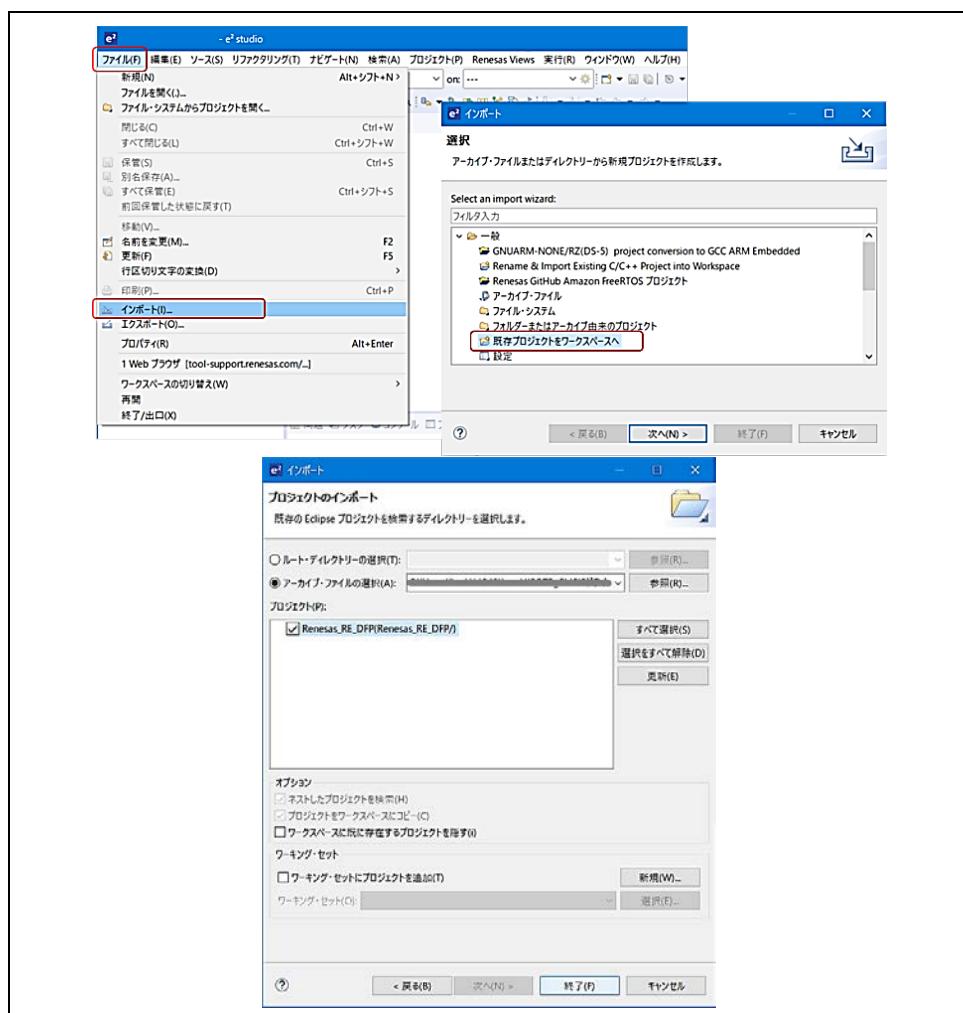


図 2-9 プロジェクトのインポート手順

③ コンパイル

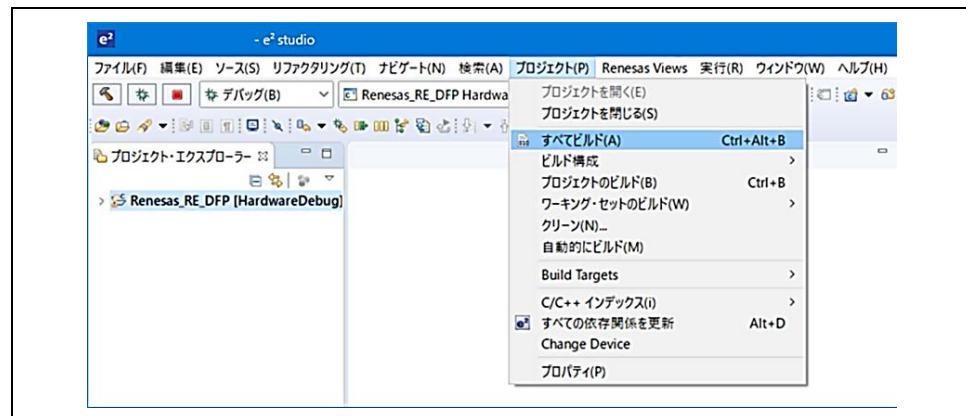


図 2-10 コンパイルメニュー

④ J-Link の設定

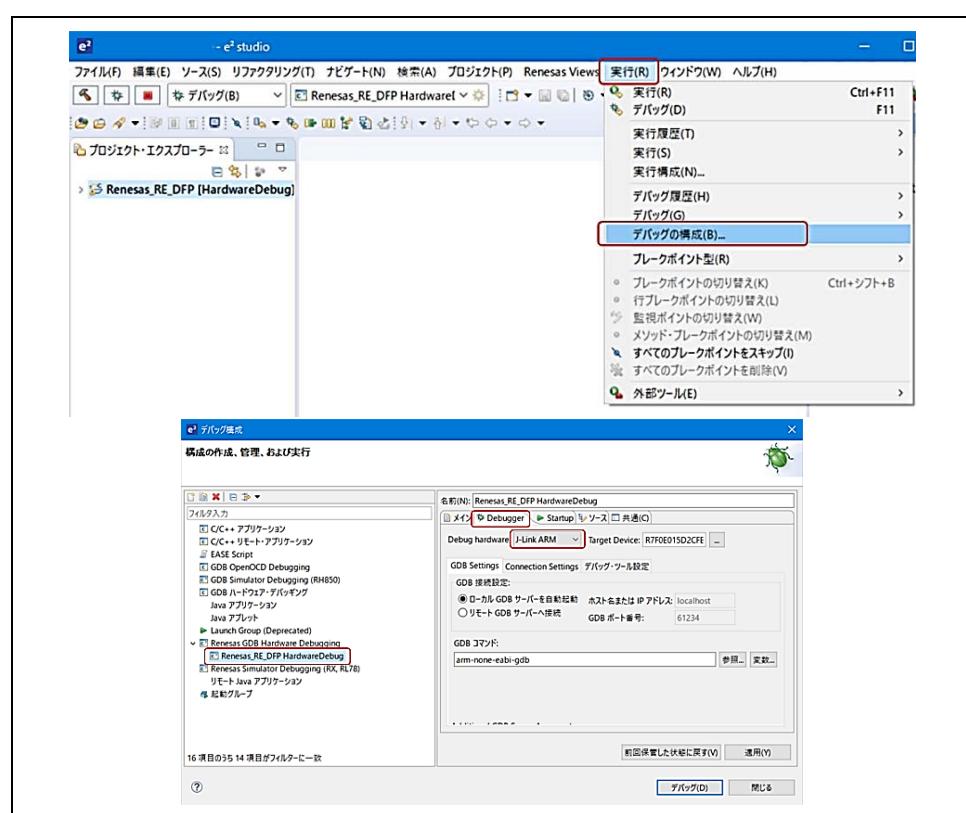


図 2-11 J-Link の設定手順

⑤ ボードと接続

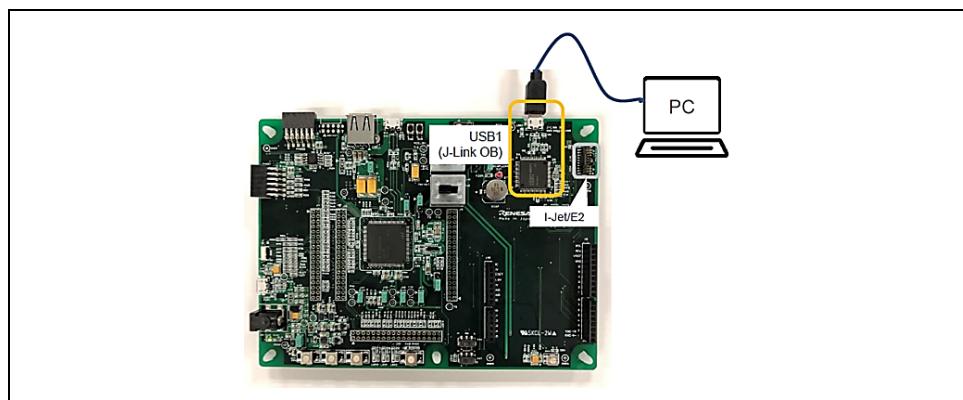


図 2-12 ボード接続例

⑥ ダウンロードして
デバッガを起動

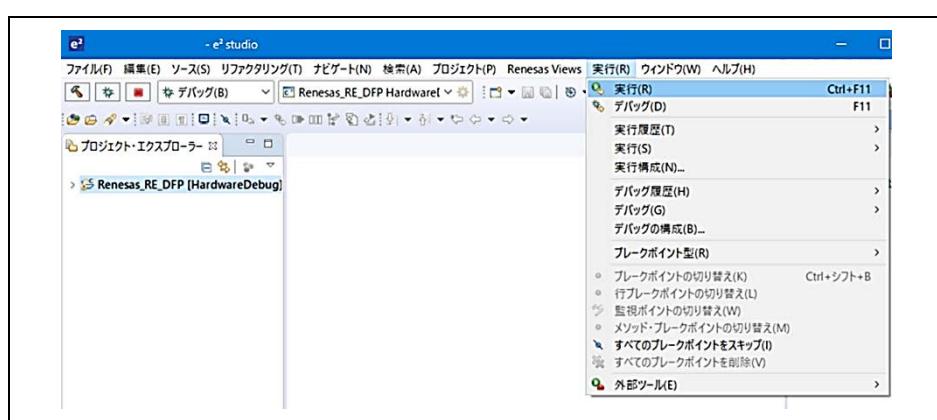


図 2-13 デバッガ起動メニュー

3. コンポーネント

本パッケージに同梱されているドライバで構成されるコンポーネントについて説明します。RE01 256KB グループは RE01 1500KB グループと同じ構成です。図内の”1500KB”を”256KB”に読み替えてください。

3.1 CMSIS-CORE

本パッケージの CMSIS-CORE は、Arm®社提供のヘッダとルネサス提供のコードによるドライバ群で構成されるコンポーネントです。

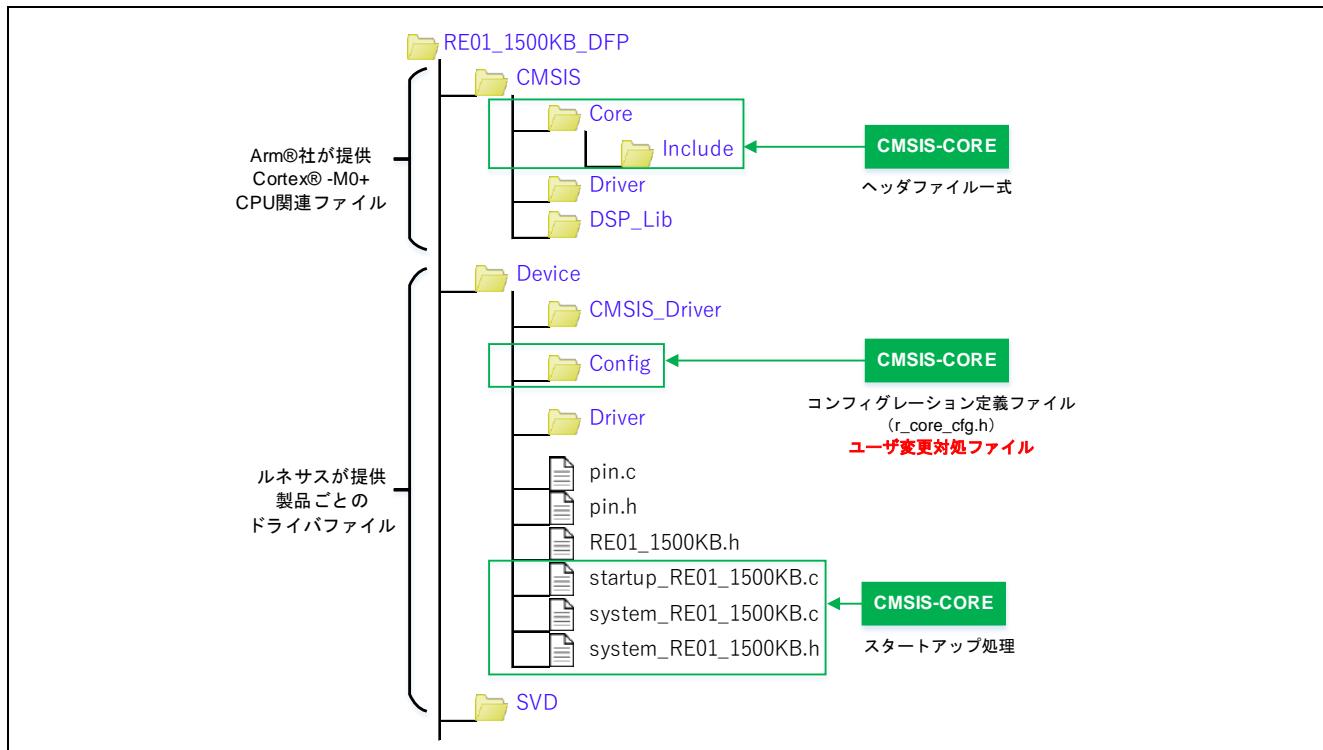


図 3-1 CMSIS-CORE 関連ファイル

3.1.1 サポートドライバ

本パッケージの CMSIS-CORE は、R_CORE ドライバを含みます。

R_CORE ドライバは、コンフィグレーション定義ヘッダを持ち、ユーザは動作環境にあわせて定義値を編集することができます。

3.1.2 R_CORE ドライバの主な役割

R_CORE ドライバの主な役割を説明します。

表 3-1 R_CORE ドライバの主な役割

役割	内容
割り込みベクタテーブル	リセット解除や IRQ などの割り込みが発生した際、エントリ関数のアドレスを管理する割り込みベクタテーブルを持ちます 詳細は、7.2.4.1 割り込み制御 を参照してください
スタートアップ処理	リセット解除後のエントリ関数で、main 関数を実行する前にスタートアップ処理を行います 本パッケージでは、CMSIS-CORE で規定されるスタートアップに加え、r_core_cfg.h の設定に従い、動作クロックおよび電力制御モードの初期設定を行います 詳細は、6.1 スタートアップ処理 を参照してください

3.2 CMSIS-Driver

本パッケージの CMSIS-Driver は、Arm®社提供ヘッダとルネサス提供コードおよび一部の拡張ヘッダによるドライバ群で構成されるコンポーネントです。

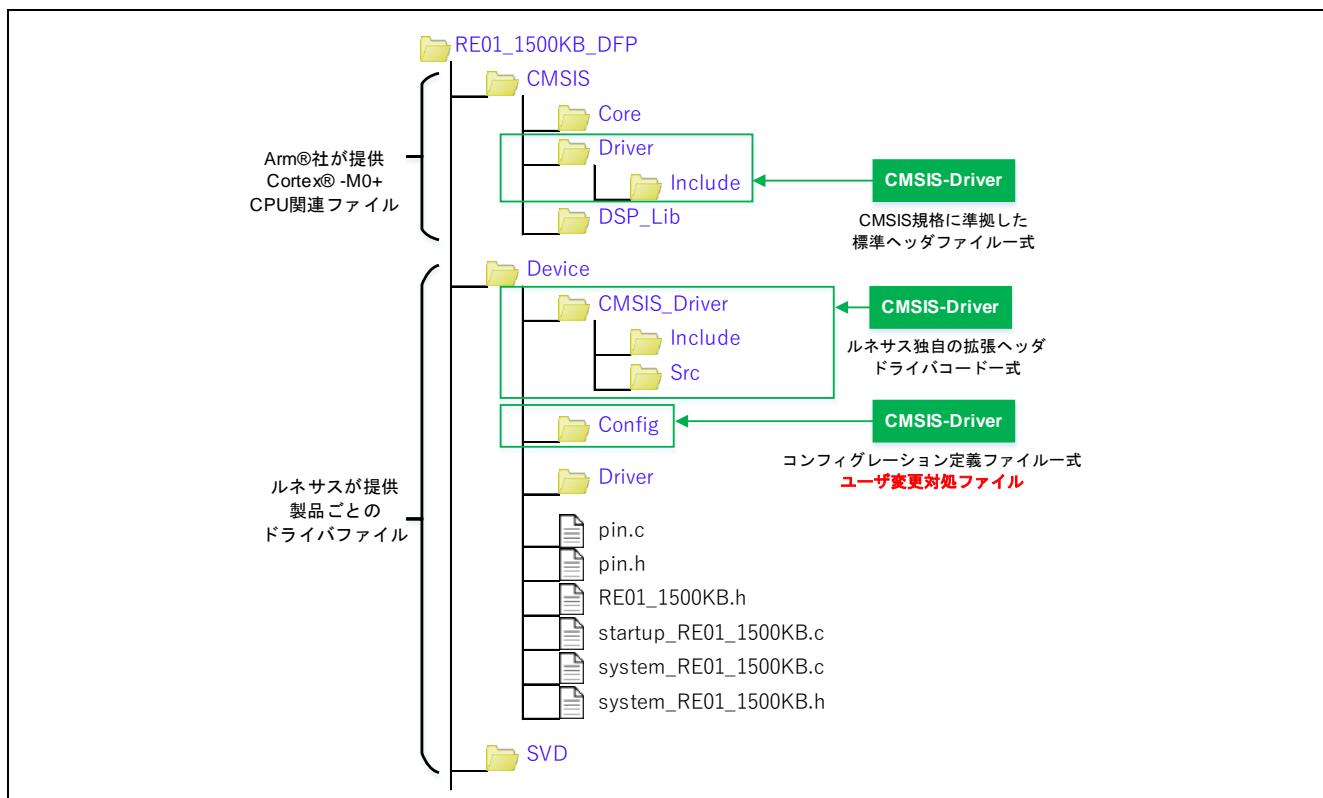


図 3-2 CMSIS-Driver 関連ファイル

3.2.1 サポートドライバ

本パッケージの CMSIS-Driver は周辺機能ドライバを含みます。本パッケージがサポートする CMSIS-Driver を表 3-2 に示します。

各ドライバは、Config フォルダ内にコンフィグレーション定義ヘッダを持ち、ユーザは動作環境にあわせて定義値を編集することができます。

表 3-2 CMSIS-Driver サポートドライバ

ドライバ名	ルネサス独自の拡張機能
R_SPI	なし
R_I2C	あり
R_USART	あり

3.2.2 拡張機能

本パッケージの CMSIS-Driver は、一部ドライバで拡張機能を持ちます。

ルネサス独自の拡張機能を持つドライバを使用する場合は、拡張ヘッダファイルをインクルードしてください。標準ヘッダファイルは拡張ヘッダファイル内でインクルードしているため、標準ヘッダファイルのインクルードは不要です。

3.3 CMSIS-DSP

本パッケージの CMSIS-DSP は、Arm®社が提供するヘッダのみで構成されるコンポーネントです。

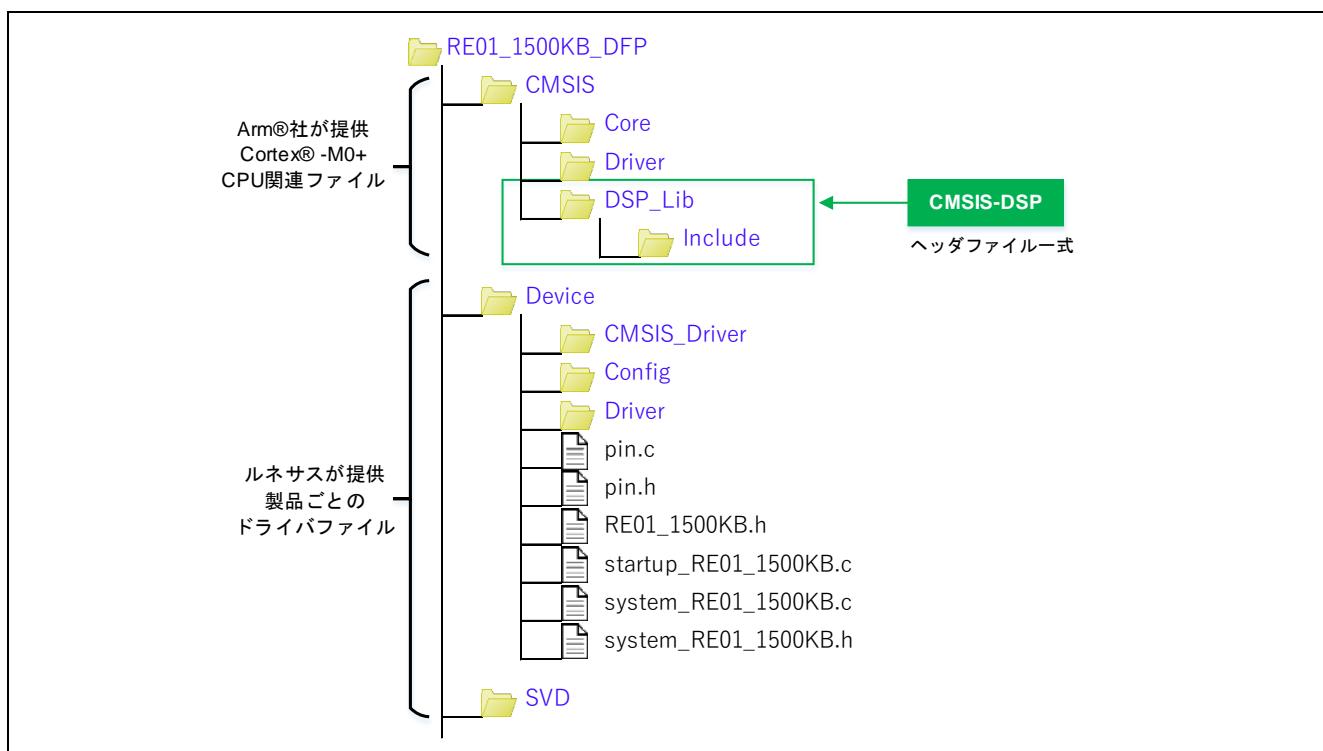


図 3-3 CMSIS-DSP 関連ファイル

3.3.1 DSP ライブラリ取り込み手順

Arm®社が提供する CMSIS パッケージ内の DSP ライブラリをプロジェクトに取り込む手順について説明します。

3.3.1.1 EWARM 編

Arm®社が提供する DSP ライブラリは、一部関数のコードサイズが大きく、EWARM の無償評価版ライセンス（コードサイズ制限版）を使用した場合にコンパイルができない問題があります。

この問題は CMSIS パッケージ内の DSP ソースコードをコンパイルして生成されるライブラリを使用することで解消できます。

DSP ソースコードからライブラリを生成し、プロジェクトに取り込む手順を説明します。

各ファイルの所在は図 3-4 を参照してください。図 3-4 は、左側に Arm®社が提供する CMSIS パッケージ、右側に本パッケージのプロジェクト (RE01_1500KB_DFP) を示しています。

CMSIS パッケージはバージョン 5.4.0 としていますが、使用するバージョンに適宜読み替えてください。

- ① Arm®社が提供する CMSIS パッケージ内にある「arm_cortexM_math.eww」をダブルクリックし、プロジェクトを立ち上げます。
- ② プロジェクトのオプションで、プロセッサに「Cortex-M0」が選択されていることを確認し、コンパイルを実行します。DSP ライブラリ「iar_cortexM0I_math.a」が生成されます。
- ③ DSP ヘッダファイル「arm_common_tables.h」「arm_const_structs.h」「arm_math.h」および生成された DSP ライブラリ「iar_cortexM0I_math.a」を本パッケージのプロジェクトに取り込みます。

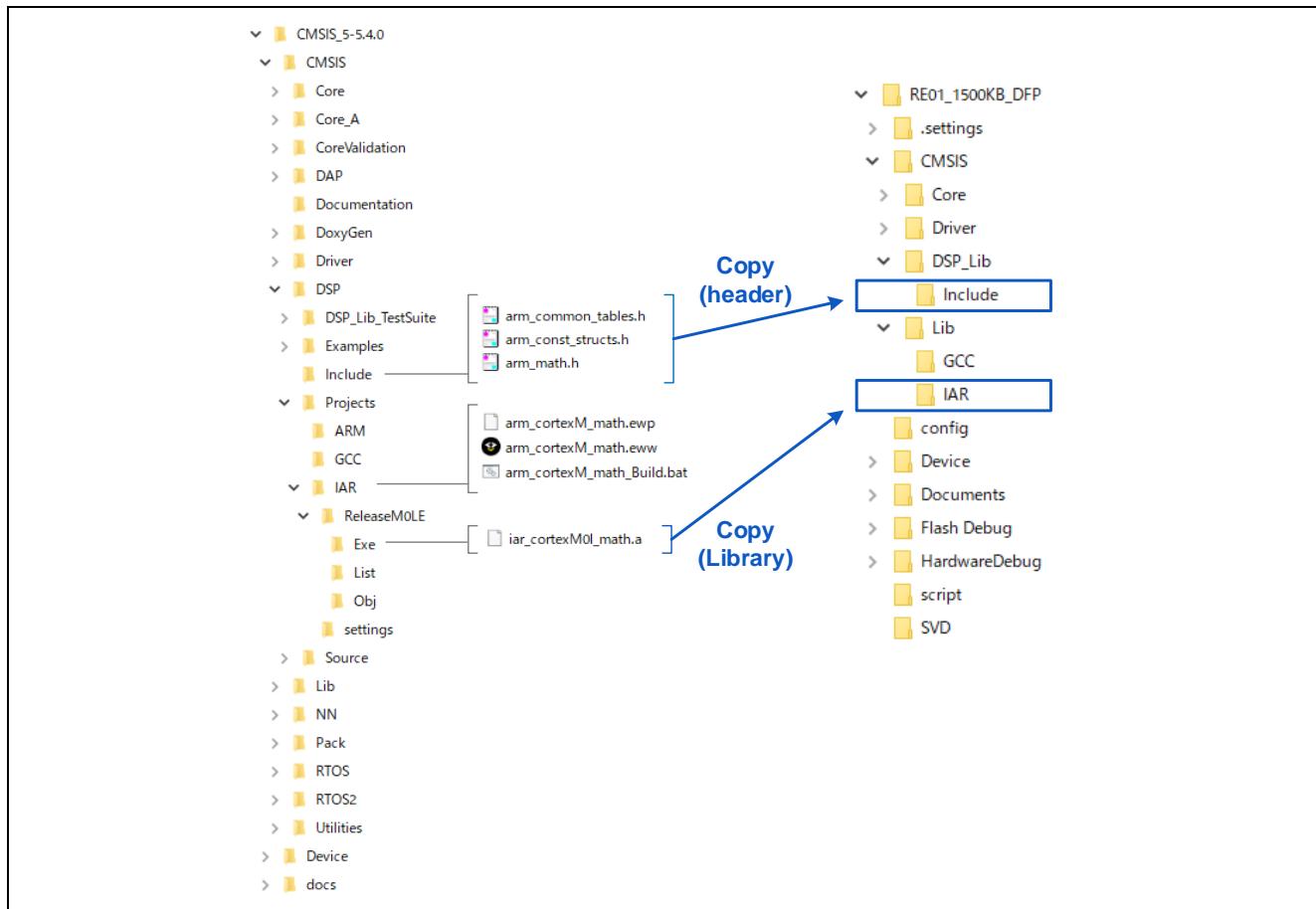


図 3-4 DSP ライブラリの取り込み (IAR コンパイラ用)

④ 本パッケージのプロジェクトに、DSP ライブラリをビルド対象ファイルとして追加します。

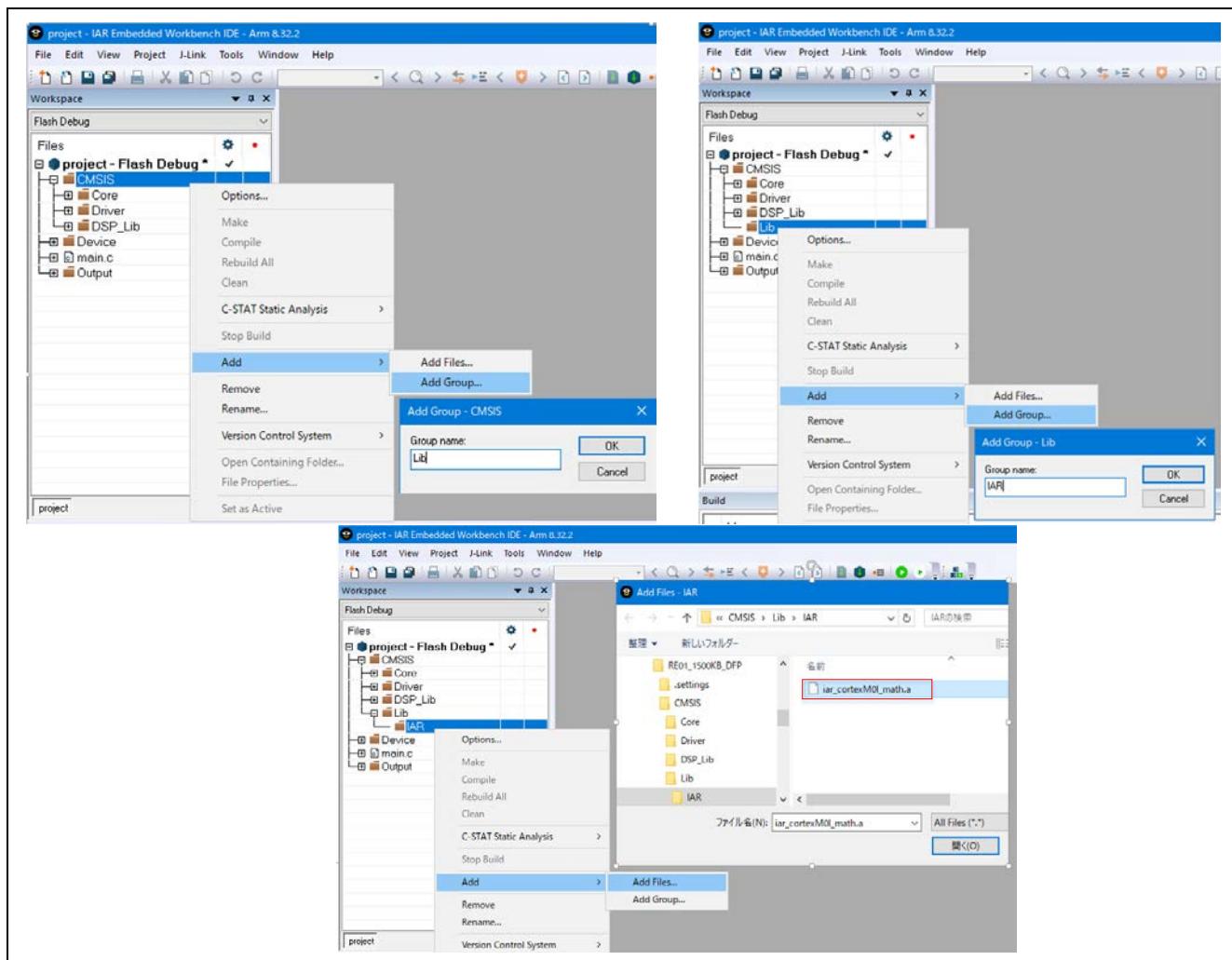


図 3-5 EWARM におけるビルド対象ファイルの追加

⑤ プロジェクトのオプションで以下の設定を行います。

- ・ インクルードパスに、DSP ヘッダファイルを格納したフォルダパスを追加
- ・ プリプロセッサに、「ARM_MATH_CM0PLUS」を追加

EWARM におけるオプション設定については図 3-6 を参照してください。

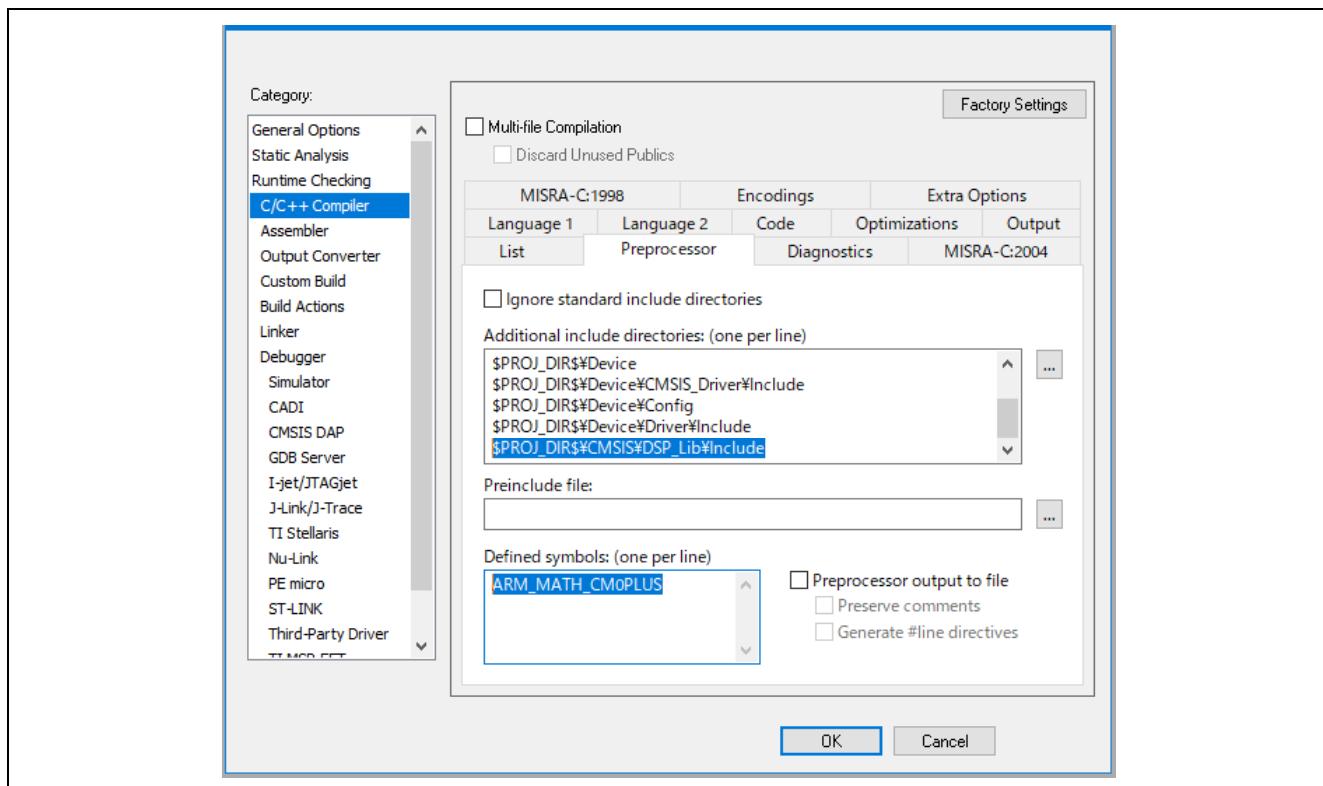


図 3-6 EWARM におけるインクルードパスおよびプリプロセッサの設定

3.3.1.2 e² studio 編

各ファイルの所在は図 3-7 を参照してください。図 3-7 は、左側に Arm®社が提供する CMSIS パッケージ、右側に本パッケージのプロジェクト (RE01_1500KB_DFP) を示しています。

CMSIS パッケージはバージョン 5.4.0 としていますが、使用するバージョンに適宜読み替えてください。

- ① DSP ヘッダファイル「arm_common_tables.h」「arm_const_structs.h」「arm_math.h」、および DSP ライブラリ「iar_cortexM0I_math.a」を本パッケージのプロジェクトに取り込みます。

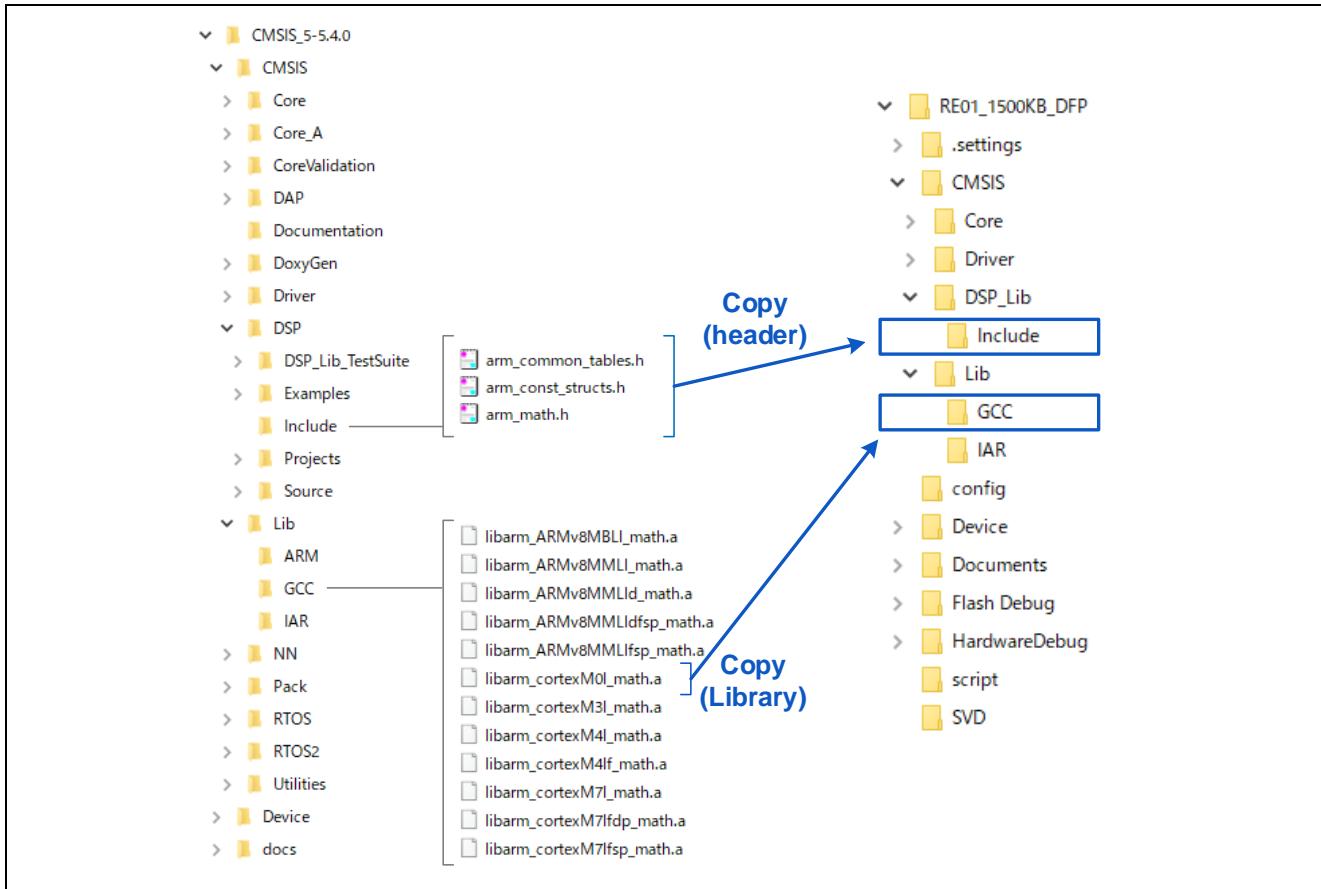


図 3-7 DSP ライブラリの取り込み (GCC コンパイラ用)

② プロジェクトのプロパティで以下の設定を行います。

インクルードパスに、DSP ヘッダファイルを格納したフォルダパスを追加

- プリプロセッサに、プリプロセッサに、「ARM_MATH_CM0PLUS」を追加
- ライブラリに、DSP ファイル名および DSP ライブラリファイルを格納したフォルダパスを追加

e² studio におけるプロパティの設定については図 3-8～図 3-10 を参照してください。

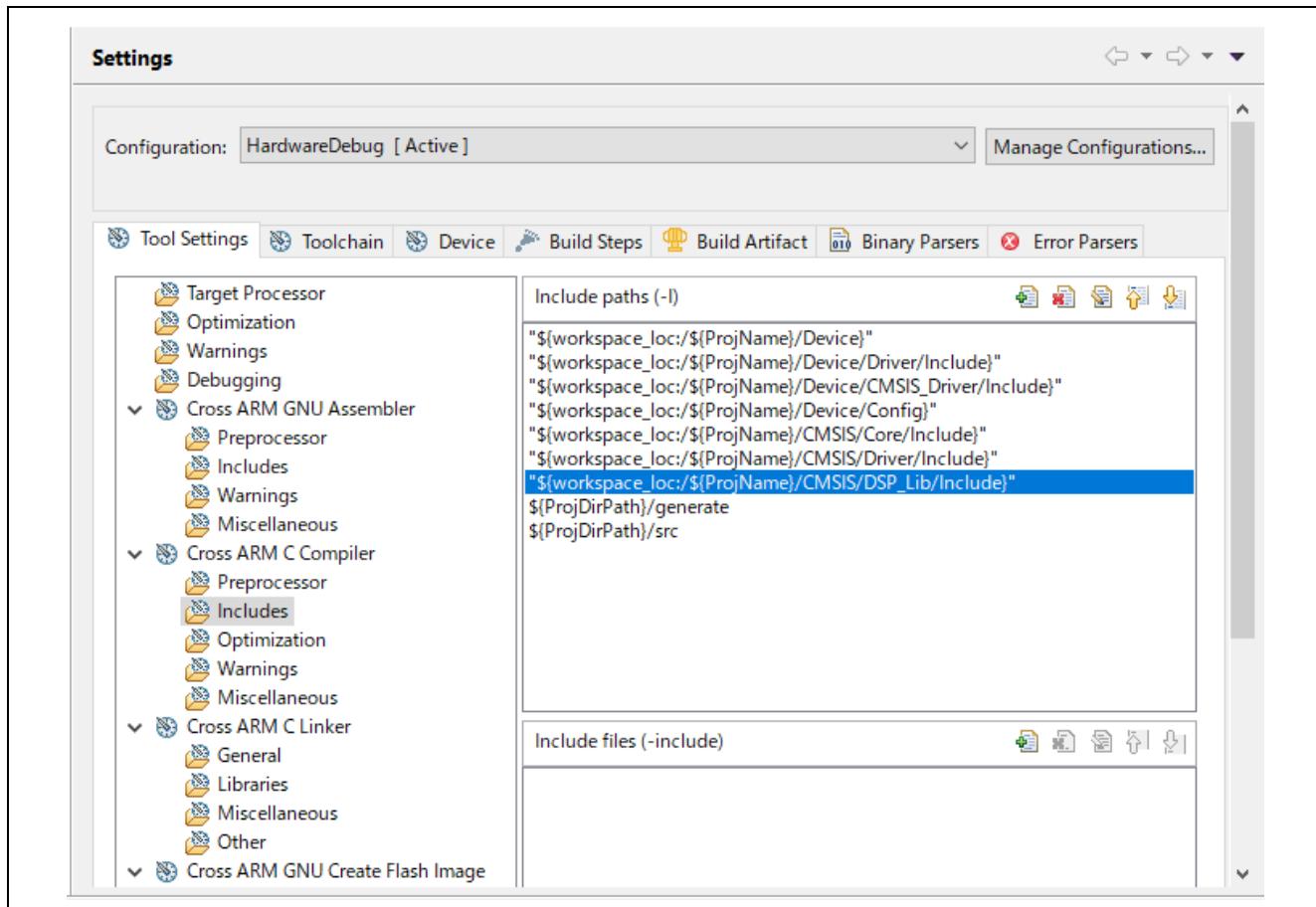
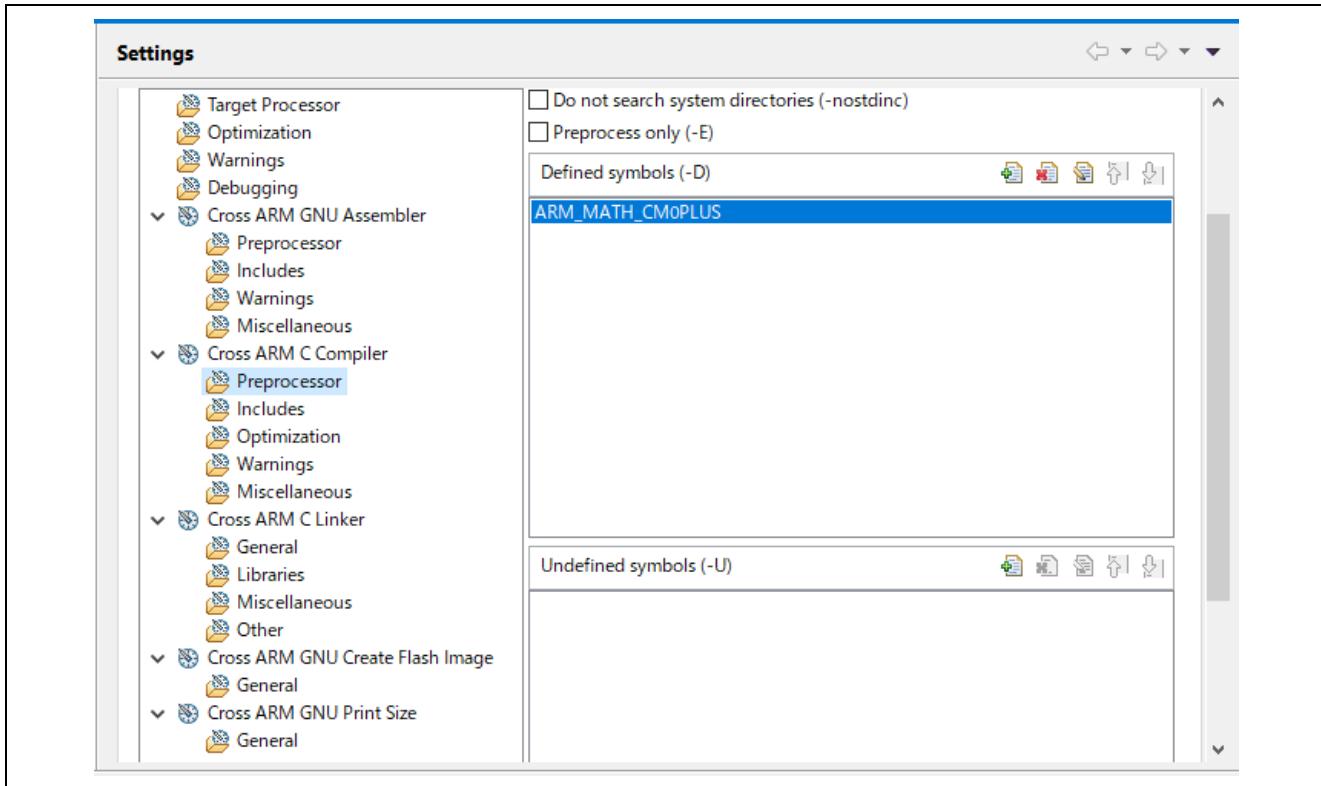
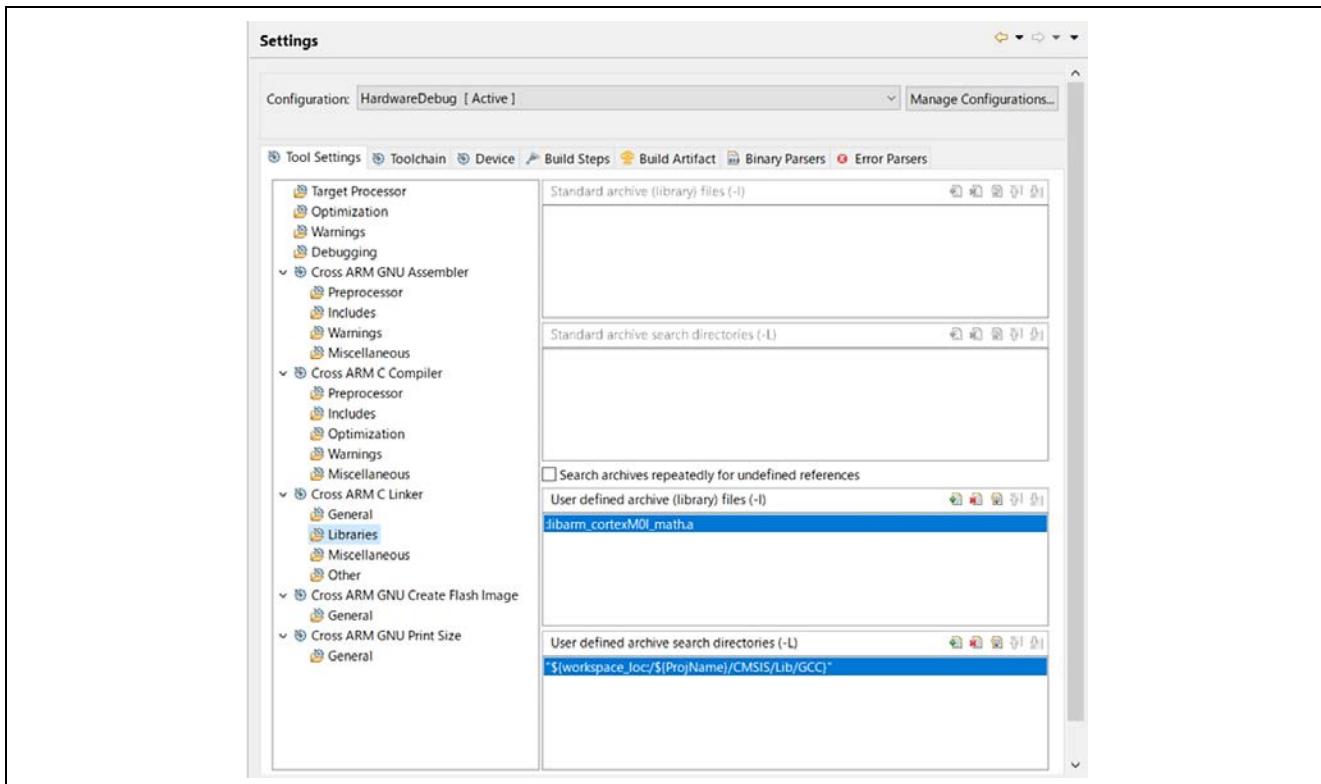


図 3-8 e² studio におけるインクルードパスの設定

図 3-9 e² studio におけるプリプロセッサの設定図 3-10 e² studio におけるライブラリの設定

3.4 HAL-Driver

本パッケージの HAL-Driver は、ヘッダ、コード共にルネサスが提供するドライバで構成されるコンポーネントです。

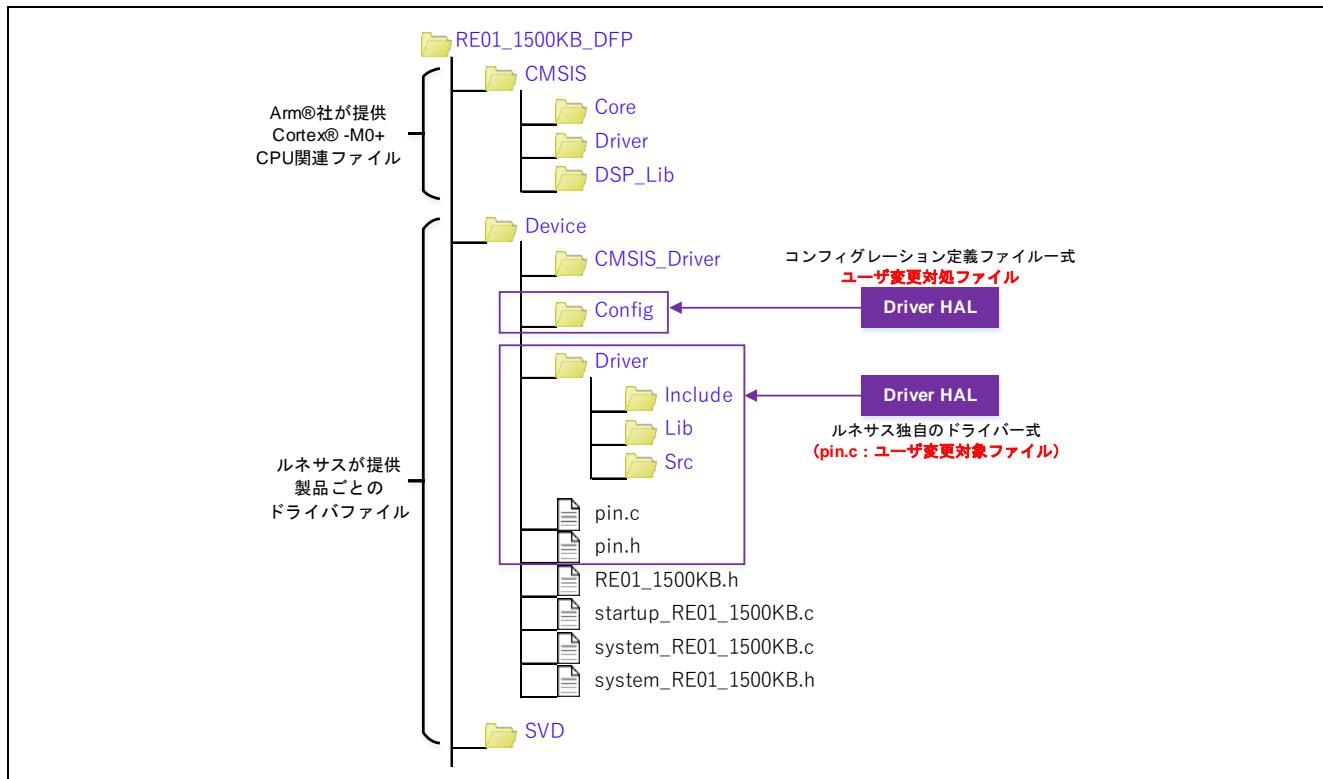


図 3-11 HAL-Driver 関連ファイル

3.4.1 サポートドライバ

本パッケージの Driver HAL は共通機能ドライバと周辺機能ドライバを含みます。本パッケージでサポートする Driver-HAL を表 3-3 に示します。

各ドライバは、Config フォルダ内にコンフィグレーション定義ヘッダを持ち、ユーザは動作環境にあわせて定義値を編集することができます。

表 3-3 HAL-Driver サポートドライバ

ドライバ名	分類
R_SYSTEM	共通機能 ドライバ
R_LPM	
R_PIN	
R_ADC	周辺機能 ドライバ
R_DMAC	
R_DTC	
R_FLASH	
R_GDT	
R_SMIP	
R_PMIP	
R_USB ^注	

【注】 RE01 256KB グループ CMSIS パッケージはサポートしていません。

3.4.2 共通機能ドライバ

共通機能ドライバは、ユーザプログラムやドライバが使用する共通機能関数を持ちます。

3.4.2.1 R_SYSTEM ドライバの主な役割

R_SYSTEM ドライバの主な役割を表 3-4 に示します。

表 3-4 R_SYSTEM ドライバの主な役割

役割	内容
クロックの設定	クロック設定を行う関数を用意しています 詳細は、6.4 クロック設定 を参照してください
割り込みの設定	割り込み制御用の関数および定義ファイルを用意しています 詳細は、6.3 割り込み制御 を参照してください
プログラムの RAM 展開	RAM 配置用セクションに配置されたプログラムを、RAM 領域にコピーする関数を用意しています プログラムの RAM 配置については、6.6 プログラムの RAM 配置 を参照してください

3.4.2.2 R_LPM ドライバの主な役割

R_LPM ドライバの主な役割を表 3-5 に示します。

表 3-5 R_LPM ドライバの主な役割

役割	内容
IO 電源 ドメインの不定値伝搬抑制機能の設定	IO 電源 ドメイン不定値伝搬抑制制御用の関数を用意しています 詳細は、6.2 IO 電源 ドメイン不定値伝搬抑制制御 を 参照してください
低消費電力モードの設定	低消費電力モード制御用の関数を用意しています

3.4.2.3 R_PIN ドライバの主な役割

R_PIN ドライバの主な役割を表 3-6 に示します。

表 3-6 R_PIN ドライバの主な役割

役割	内容
端子設定	周辺機能で使用する端子制御用の関数を用意しています 詳細は、6.5 端子設定 を 参照してください

3.4.3 周辺機能ドライバ

周辺機能ドライバは、ユーザプログラムが使用する周辺機能関数を持ちます。

4. ドライバ仕様書

本パッケージには、各ドライバの仕様書が含まれています。ドライバ仕様書の所在を以下に示します。
RE01 256KB グループは RE01 1500KB グループと同じ構成です。図内の”1500KB”を”256KB”に読み替えてください。

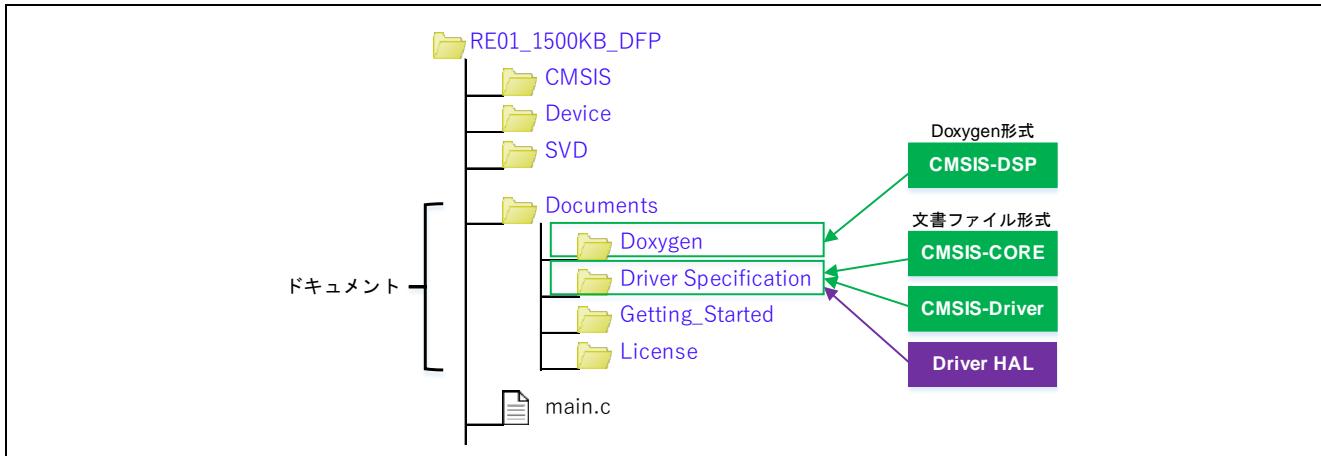


図 4-1 ドライバ仕様書関連ファイル

Arm®社が提供する DSP ライブラリは Doxygen に対応しています。本パッケージでは、Doxygen 関連ファイルは圧縮されています。

DSP ライブラリの仕様を表示するための手順を図 4-2 に示します。

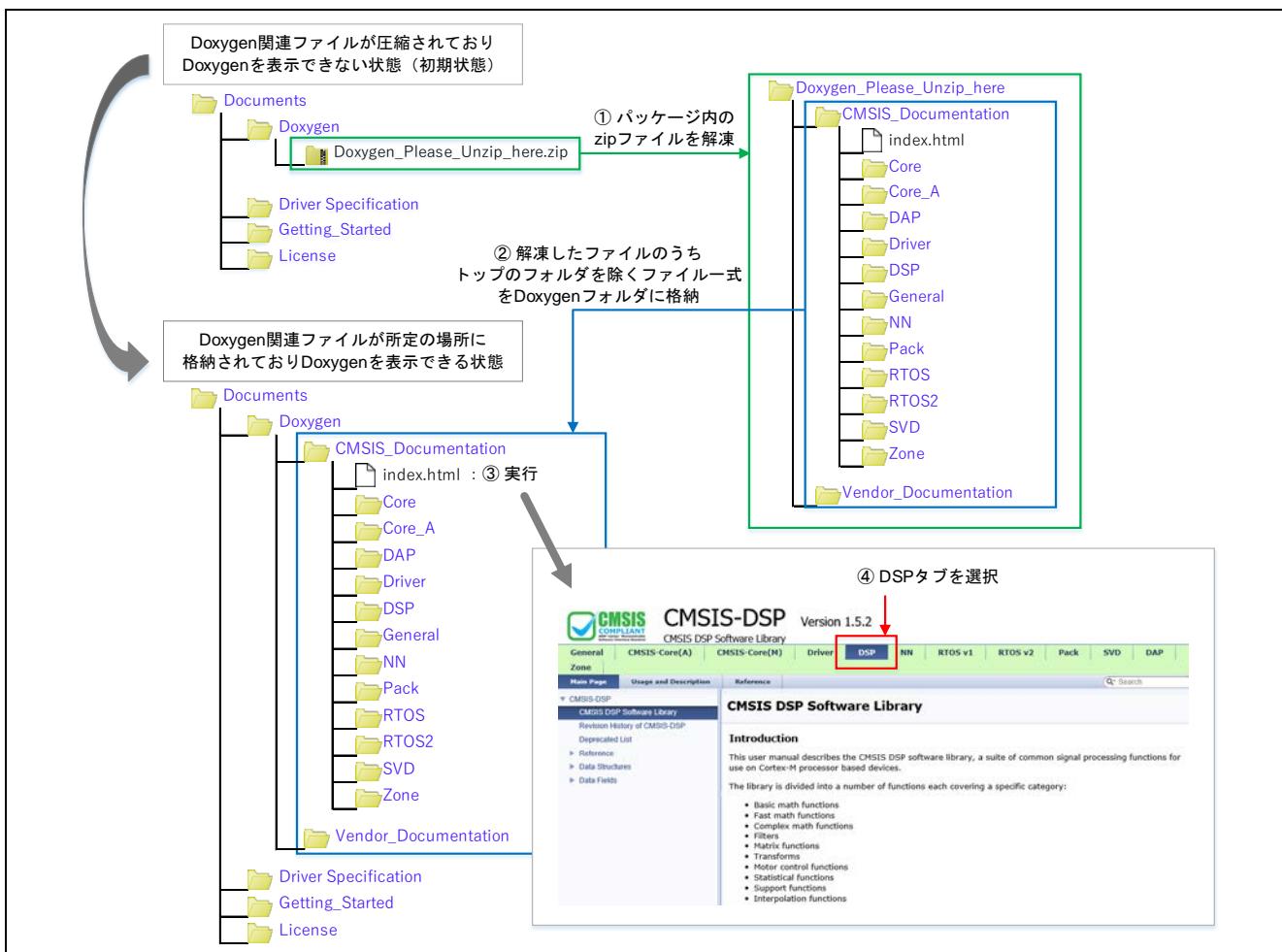


図 4-2 Doxygen による CMSIS-DSP 仕様表示方法

5. ドライバ基本概念

5.1 共通機能ドライバと周辺機能ドライバ

本パッケージに含まれるドライバは共通機能ドライバと周辺機能ドライバの2種類が含まれます。

表 5-1 ドライバの分類

分類	内容
共通機能ドライバ	<p>ユーザプログラムやドライバが使用する共通機能関数を持つドライバ 対象ドライバ：</p> <ul style="list-style-type: none"> • R_CORE ドライバ • R_SYSTEM ドライバ • R_PIN ドライバ • R_LPM ドライバ
周辺機能ドライバ	<p>ユーザプログラムが使用する周辺機能関数を持つドライバ 対象ドライバ：</p> <ul style="list-style-type: none"> • 上記、共通機能ドライバを除くすべてのドライバ

5.2 ドライバの構成

各ドライバは3種類のファイルで構成されます（一部例外あり）。

表 5-2 ドライバを構成するファイル

ファイル名	内容
r_***_api.c	ドライバのコード本体
r_***_api.h	ドライバを使用する場合に必要な定義を行う、インクルードヘッダ ユーザはドライバを使用する場合にインクルードする必要があります
r_***_cfg.h	ドライバの動作条件を定義した、コンフィグレーション定義ヘッダ ユーザは動作環境にあわせて本ファイルの定義値を編集することができます ユーザは本ファイルをインクルードする必要はありません R_CORE ドライバのコンフィグレーション定義ヘッダは、スタートアップ処理および R_SYSTEM ドライバで使用されます

【注】ファイル名の「***」は、ドライバ機能名を示します。

5.3 共通機能の設定

周辺機能ドライバは、ドライバ内で共通機能ドライバ関数を実行して共通機能の設定を行います。

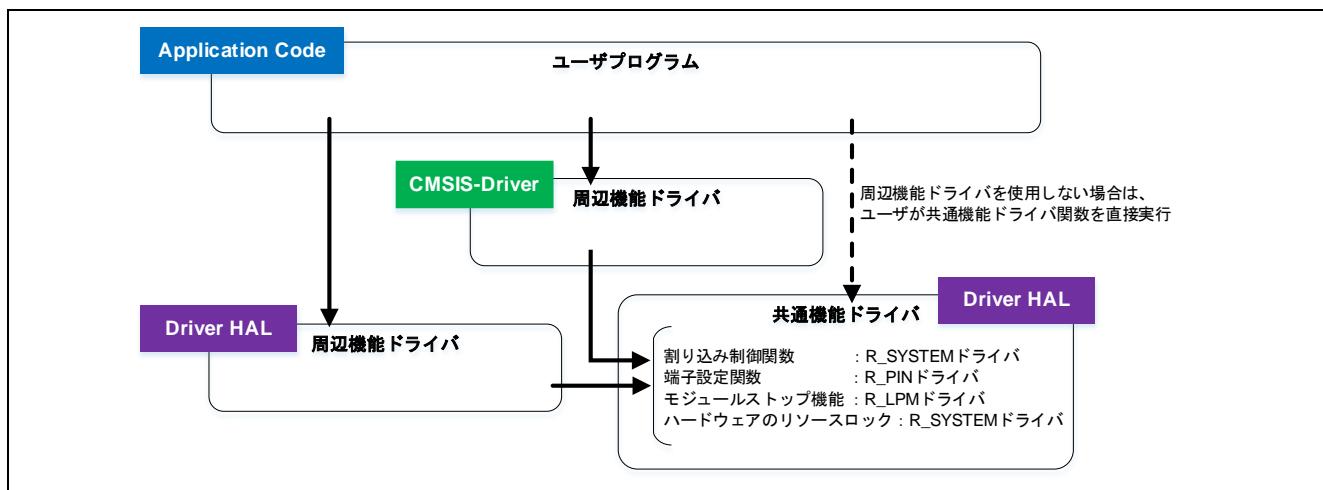


図 5-1 共通機能の設定

6. 基本機能

6.1 スタートアップ処理

本パッケージを使用する場合、リセット解除後のエントリ関数として登録されている Reset_Handler 関数が呼ばれます。Reset_Handler 関数は main 関数を実行するまでの間にスタートアップ処理を行います。スタートアップ処理の動作フローを図 6-1 に示します。RE01 256KB グループは RE01 1500KB グループと同じ構成です。図内の”1500KB”を”256KB”に読み替えてください。

スタートアップ処理では、主に以下の処理を行います。

- 動作開始時の端子設定
- 動作開始時のクロック・電力制御モードの設定

main 関数の実行前にスタートアップ処理を行います。main 関数が実行された時点では、ハードウェアレジスタのリセット解除後の値から変更されている場合がありますのでご注意ください。

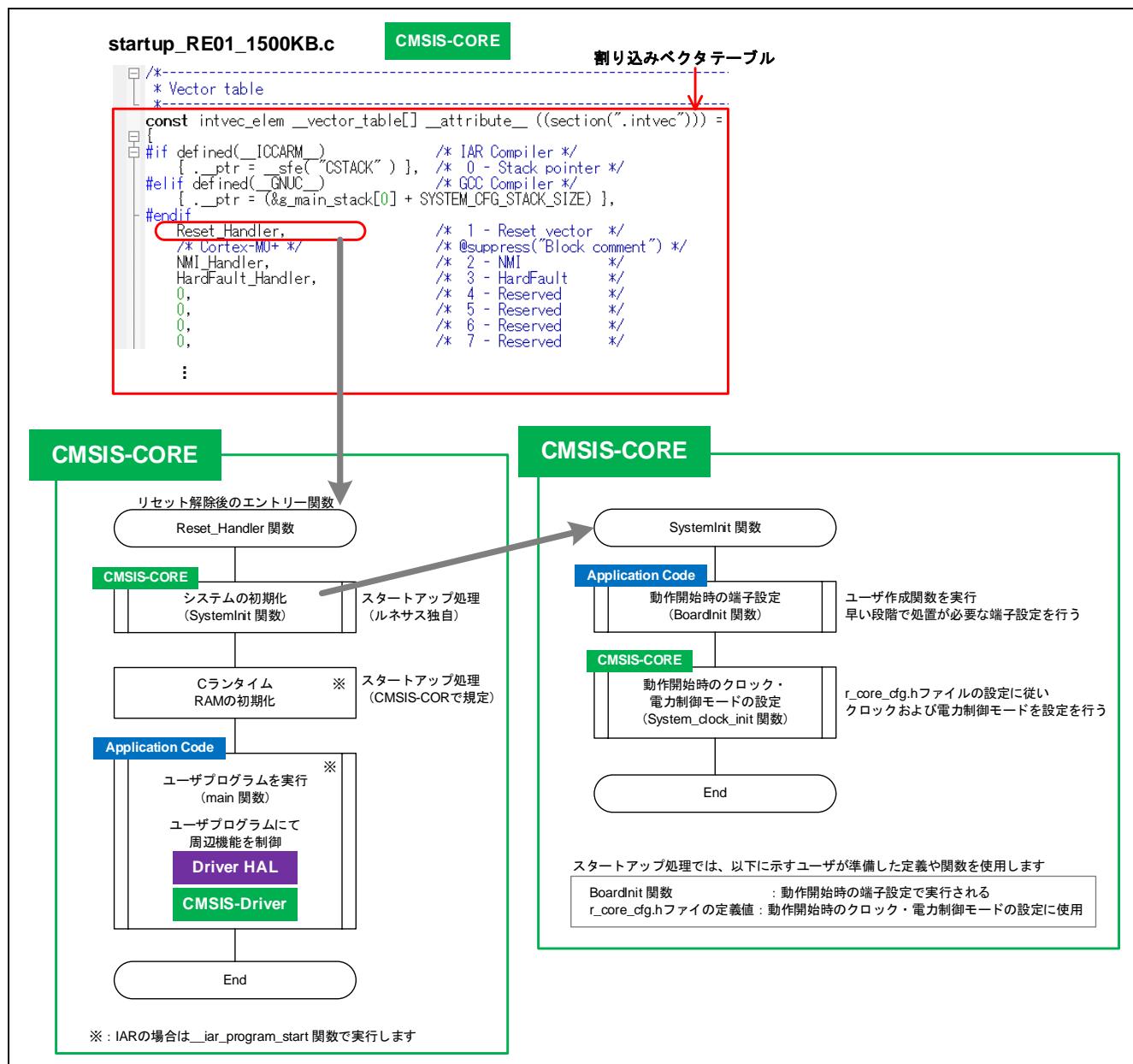


図 6-1 スタートアップ処理の動作フロー

6.1.1 動作開始時の端子設定

スタートアップ処理ではユーザ作成の BoardInit 関数を実行します。リセット解除後、早い段階で端子設定が必要な場合は、BoardInit 関数を作成し端子処置を行ってください。

BoardInit 関数

- 関数形式 : void BoardInit(void)
- 内容 : main 関数実行前にスタートアップ処理で実行されるユーザ作成関数です。
- 特徴 : BoardInit 関数は、R_CORE ドライバで Weak 関数を用意しているため、ユーザが作成しない場合でもコンパイルエラーにはなりません。
- 設定例 : 本関数の作成例を図 6-2 に示します。

ユーザプログラム

Application Code

```

/* **** Function Name: BoardInit
 * **** Description : Configure board initial setting.
 * **** This function is called by SystemInit() function in system_RE01_1500KB.c file.
 * **** This is reference to perform BoardInit process. Sample code of target is E015 SDK board (RTK70E015D)
 * **** on Renesas. Please modify this function to suit your board.
 * **** Arguments : none
 * **** Return Value : none
 ****
void BoardInit(void)
{
    /* **** This function performs at beginning of start-up after released reset pin ****/
    /* **** Please set pins here if your board is needed pins setting at the device start-up. ****/
    /* **** This function is suiting RE01_1500KB SDK board. Please change to your board pin setting ****/

    /* Handling of Unused ports (IOVCC domain) */
    /* PORT4 Setting: RE01_1500KB SDK has DCDCs. Those are connect P404 and P405. */
    /* This perform to disable those output.
    /* Those are needed to enable when using EHC start up of this
    /* Set P404 and P405 not to be used as DCDC_EN (output low: D
    /* PODR - Port Output Data
    b15-b0 PODR15 - PORD00      - Output Low Level */
    PORT4->PODR = 0x0000;

    /* PDR - Port Direction
    b15-b6 PODR15 - PRD06      - Input
    b5 -b4 PODR05 - PRD04      - Output
    b3 -b0 PODR03 - PRD00      - Input */
    PORT4->PDR = 0x0030;

    /* Handling of Unused ports (AVCC1 domain) */
    /* PORT0 Setting */
    /* Set P009, P008 and P007 as LEDs (output high) */

    /* PODR - Port Output Data
    b15-b10 PODR15 - PORD10      - Output Low Level
    b9 -b7 PODR09 - PORD07      - Output High Level
    b6 -b0 PODR06 - PORD00      - Output Low Level */
    PORT0->PODR = 0x0380;

    /* PDR - Port Direction
    b15-b10 PODR15 - PRD10      - Input PORT
    b9 -b7 PODR09 - PRD07      - Output PORT
    b6 -b0 PODR06 - PRD00      - Input PORT */
    PORT0->PDR = 0x0380;
}
/* End of function BoardInit */

```

ボード上でDCDCに接続されているP404,P405を
汎用ポートLow出力に設定

ボード上でLEDに接続されているP007,P008, P009を
汎用ポートHigh出力に設定

図 6-2 動作開始時の端子設定関数の作成例

6.1.2 動作開始時のクロック・電力制御モードの設定

スタートアップ処理では、`r_core_cfg.h` の設定に従い、クロックおよび電力制御モードの初期設定を行います。ユーザは、動作環境にあわせて `r_core_cfg.h` の定義値を編集してください。

`r_core_cfg.h` : 動作開始時のクロック／電力制御モードの設定

- 内容 : 動作開始時のクロック／電力制御モードを設定
- 定義名および定義値については、6.4.1 クロック定義 を参照してください。
動作開始時のモードとして選択可能な電力制御モードおよび
初期値で選択されているクロック／電力制御モードの状態を図 6-3、図 6-4 に
示します。
- 設定例 : 図 6-5 に示します。

(1) RE01 1500KB グループ

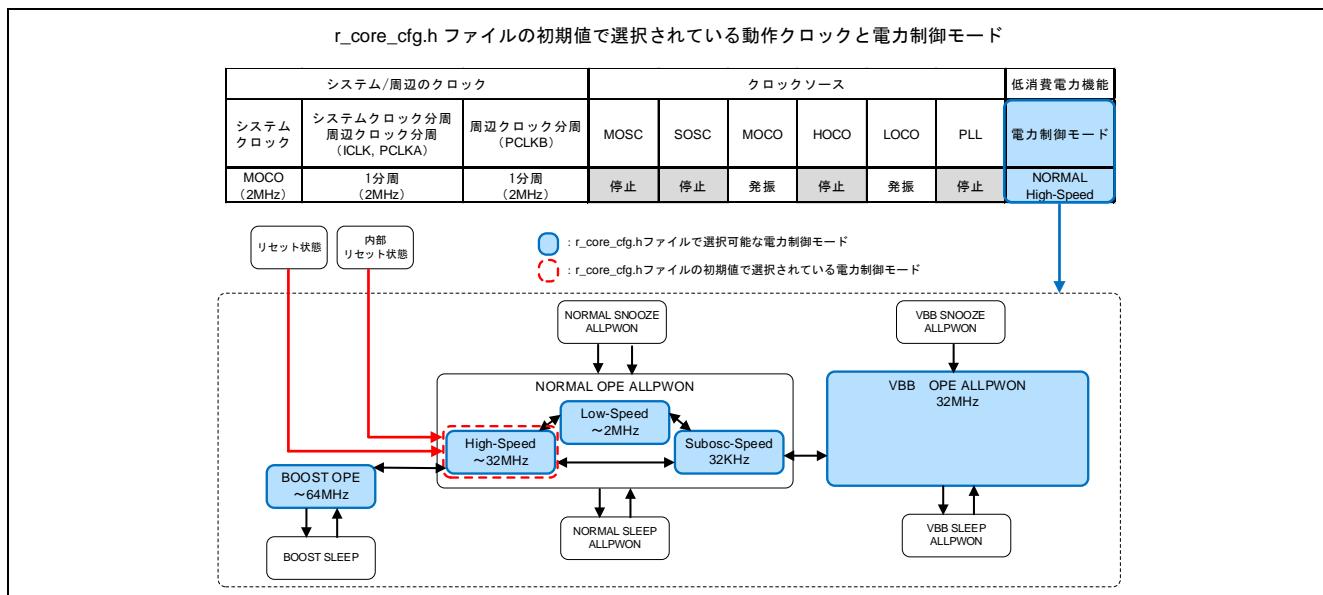


図 6-3 `r_core_cfg.h` で選択可能なクロックおよび電力制御モード

(2) RE01 256KB グループ

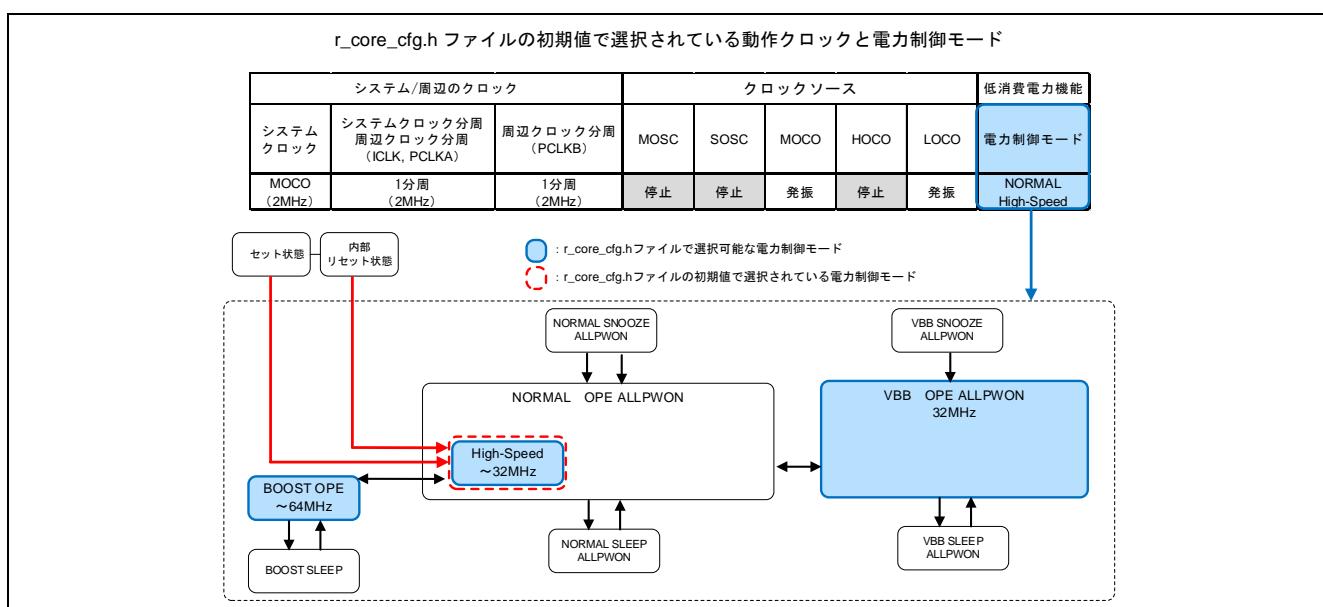


図 6-4 `r_core_cfg.h` で選択可能なクロックおよび電力制御モード

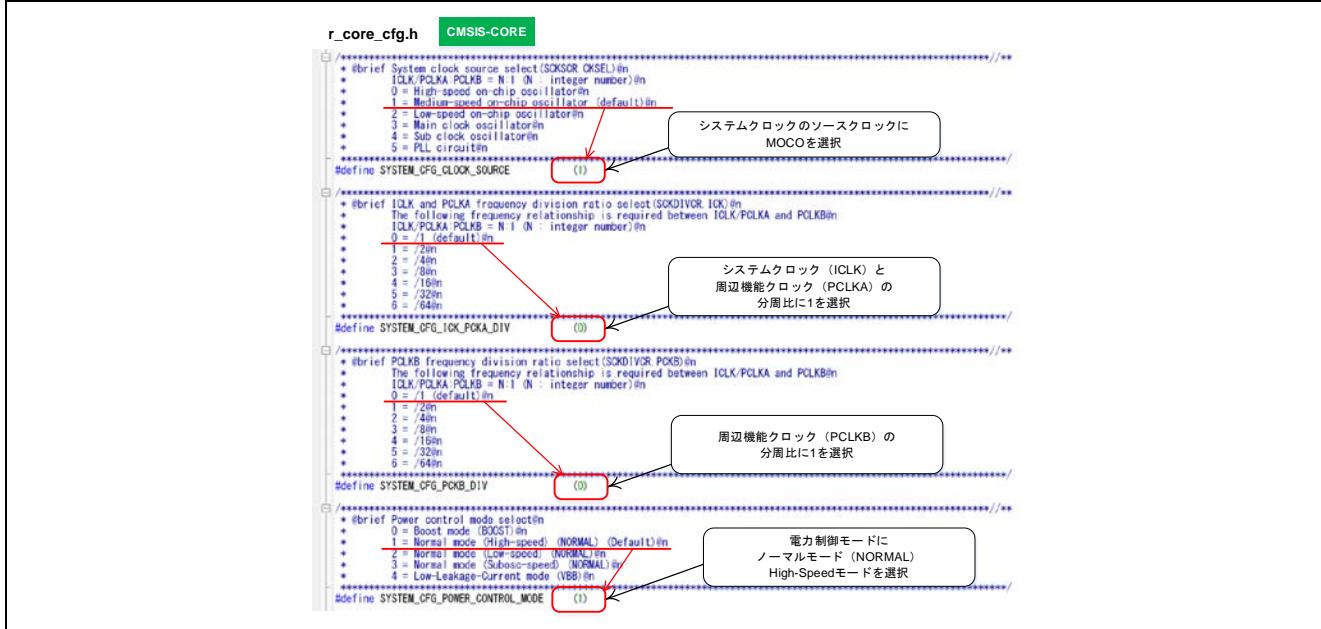


図 6-5 動作開始時のクロック・電力制御モードの設定例

6.2 IO 電源ドメイン不定値伝搬抑止制御

本デバイスは複数の IO 電源ドメインを持ち、ドメインごとに電源の供給／遮断することができます。また電源供給のないドメインからの不定値が伝搬しないよう抑制する、不定値伝搬抑止機能があります。

リセット解除後は、IOVCC ドメインを除くすべての IO 電源ドメインの不定値伝搬抑止機能が有効であり、IO 電源ドメインへ電源を給電しても端子を使用することはできません。各電源の接続状態にあわせて、本機能を制御する必要があります。

- 電源供給されているドメイン： 不定値伝搬抑止機能を無効にすること
- 電源供給されていないドメイン： 不定値伝搬抑止機能を有効にすること

なお Evaluation Kit は、ジャンパ設定により以下を選択可能です

- IOVCC を除く全電源ドメインへの電源供給あり（通常の起動用）
- 外部 DCDC を有効にするまで IOVCC を除く全電源ドメインの電源供給なし
(エナジーハーベスト起動用)

6.2.1 対応電源ドメイン

各 IO 電源ドメインに配置されるハードウェア機能および端子を表 6-1、表 6-2 に示します。各端子の対応電源ドメインは、対象デバイスの「ユーザーズマニュアル ハードウェア編 機能別端子一覧」を参照してください。

(1) RE01 1500KB グループ

表 6-1 各 IO 電源ドメインに配置されるハードウェア機能および端子

IO 電源ドメイン	ハードウェア機能／端子
IOVCC	以下を除くすべてのハードウェア機能／端子
AVCC0	14 ビット A/D コンバータ (S14AD) 温度センサ回路 (TEMPS) 基準電圧生成回路 (VREF)
AVCC1	12 ビット D/A コンバータ (R12DA) アナログコンパレータ (ACMP)
IOVCC0	ポート 8 に割り当てられた入出力機能
IOVCC1	ポート 3, ポート 6, ポート 7, P202～P204 に割り当てられた入出力機能
IOVCC2	ポート 1 に割り当てられた入出力機能
IOVCC3	P010～P015, ポート 5 に割り当てられた入出力機能
USB	USB2.0FS ホスト／ファンクションモジュール (USB)
VPM	モータドライバ制御回路 (MTDV)

(2) RE01 256KB グループ

表 6-2 各 IO 電源ドメインに配置されるハードウェア機能および端子

IO 電源ドメイン	ハードウェア機能／端子
IOVCC	以下を除くすべてのハードウェア機能／端子
AVCC0	P000～P007 に割り当てられた入出力機能
IOVCC0	P010～P015, ポート 8 に割り当てられた入出力機能
IOVCC1	ポート 1, ポート 3, ポート 5, ポート 6, ポート 7, P202～P205 に割り当てられた入出力機能

各端子の対応電源ドメイン

↓

ピン 番号 144 LFQFP	電源 クロック システム制御	I/Oポート	タイマ (CAC, GPT, POE, AGT, TMR, RTC, LPG)	通信 (SCI, SPI, I2C, USB, QSPI)	表示系 (MLCD, LED)	割り込み (IRQ, KINT)	アナログ (S14AD, R12DA, ACMP)	対応電源
1		P810	CACREF_B/GTI0Q0_A/ GTIOC2A_B	SCK3_B/SCL0				IOVCC0
2	VSS							
3	IOVCC0							
4		P809	AGTEEE0_A/ GTETRGA_B/ GTIOC2B_B	TXD3_B/SDA0				IOVCC0
5		P808	AGTO0_A/GTETRGB_B	RXD3_B		IRQ2_B		IOVCC0
6		P807	AGTOAO_A/ GTIOC1A_B	CTS3_B/SSLB3_C		IRQ3_B	VCOUT_B	IOVCC0
:								
42	VSS_USB							
43				USB_DM				VCC_USB_B
44				USB_DP				VCC_USB_B
45	VCC_USB							
46		P205		CTS4_B		IRQ8_B		IOVCC1
47		P204	ADTRG0_A/GTIU_A/ RTCIC0_B	USB_VBUS/ SCK4_B		IRQ9_B		IOVCC1
48		P203	GTIV_A/RTTCIC1_B	USB_OVRCURA_A/ TXD4_B				IOVCC1
49		P202	CACREF_A/GTIW_A/ CCCOOUT_B/ RTCOOUT_B	USB_OVRCURB_A/ RXD4_B		IRQ4_A		IOVCC1

RE01グループ (1.5Mバイトフラッシュメモリ搭載製品) ユーザーズマニュアルハードウェア編 概要より抜粋

図 6-6 各端子の対応電源ドメインの確認方法

6.2.2 ドライバ関数

R_LPM ドライバは、IO 電源 ドメイン不定値伝搬抑止制御関数を用意しています。

R_LPM_IOPowerSupplyModeSet 関数

- 内容： 指定ドメインの不定値伝搬抑止機能を有効にします
- 引数： 不定値伝搬抑止機能を有効にする IO 電源 ドメインを設定します
0x00 : IOVCC を除くすべての電源 ドメインの不定値伝搬抑止機能を無効にします
- 設定例： IOVCC および IOVCC0 のみ給電しているため、IOVCC0 ドメインの不定値伝搬
抑止機能を無効に設定する場合の例を図 6-7 に示します

ユーザプログラム
Application Code

```

/* **** */
* Function Name : main
* Description   : main function
* Arguments     : none
* Return Value  : none
***** */
int main (void)
{
    R_SYS_CodeCopy();
    R_SYS_Initialize();
    R_LPM_Initialize(); // R_LPM ドライバを初期化
    /* Power IOVCC0 to use PORT_8 */
    R_LPM_IOPowerSupplyModeSet(LPM_IOPOWER_SUPPLY_IOVCC0); // 電源 ドメインIOVCC0が給電されているため
    /* USART Driver Setup */
    gsp_usart_drv->Initialize(usart_callback);
    gsp_usart_drv->PowerControl(ARM_POWER_FULL);
    gsp_usart_drv->Control(ARM_USART_MODE_ASYNCHRONOUS |
                           ARM_USART_DATA_BITS_8 |
                           ARM_USART_PARITY_NONE |
                           ARM_USART_STOP_BITS_1 |
                           ARM_USART_FLOW_CONTROL_NONE,
                           9600);
    gsp_usart_drv->Control(ARM_USART_CONTROL_TX, 1);
    gsp_usart_drv->Send(&gs_send_data[0], sizeof(gs_send_data));
    while (1)
    {
        ; /* main loop */
    }
    return 0;
} /* End of function main() */

```

図 6-7 IO 電源 ドメイン不定値伝搬抑止機能の制御例

6.3 割り込み制御

6.3.1 割り込みベクターテーブルとエントリ関数

本パッケージは複数のドライバで割り込み制御を行っています。割り込み発生時のエントリ関数を定義するベクターテーブルは、R_CORE ドライバが用意しています。割り込み発生時のエントリ関数は、割り込みの種類によって R_CORE ドライバと R_SYSTEM ドライバのいずれかで用意します。

割り込みベクターテーブルとエントリ関数の関係を図 6-8 に示します。RE01 256KB グループは RE01 1500KB グループと同じ構成です。図内の"1500KB"を"256KB"に読み替えてください。

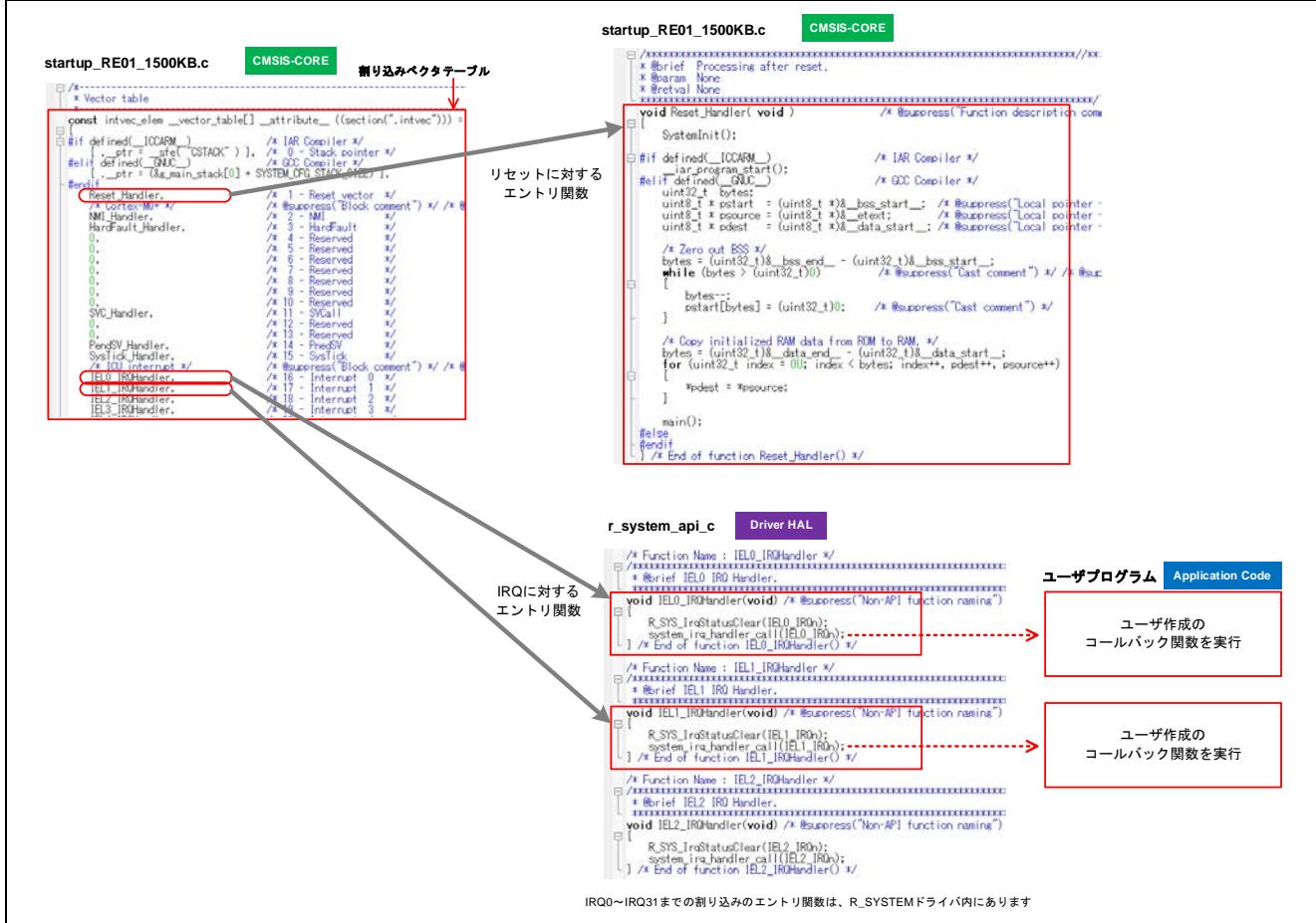


図 6-8 割り込みベクターテーブルとエントリ関数

6.3.2 IRQ 番号の割り当て

本デバイスで割り込みを使用する場合、周辺機能からの割り込みを IRQ 番号に割り当てる必要があります。AGT0 からの割り込み (AGT0_AGTI) を使用する場合を例に、周辺機能割り込みと Cortex®-M0+の接続イメージを図 6-9 に示します。

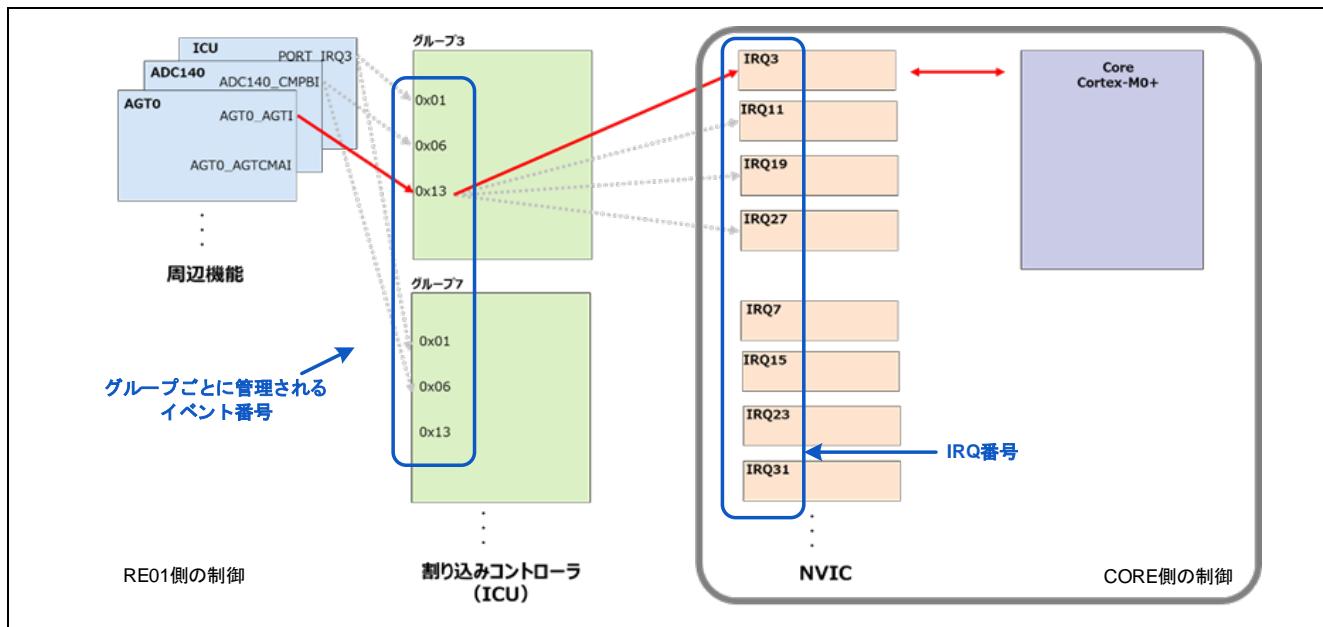


図 6-9 周辺機能割り込みと Cortex®-M0+の接続イメージ

周辺機能からの割り込みが接続可能な IRQ 番号は、対象デバイスの「ユーザーズマニュアル ハードウェア編 割り込みコントローラユニット (ICU)」を参照してください。

AGT1_AGTI を例に、各周辺機能割り込みが接続可能な IRQ 番号の確認方法を図 6-10 に示します。

表 16.7 各イベント選択のレジスタ設定値 (1 / 4)

名称	IELSRn.IELS[4:0]								SELSRD. SELS [7:0]
	グループ0 (n = 0/8/ 16/24)	グループ1 (n = 1/9/ 17/25)	グループ2 (n = 2/10/ 18/26)	グループ3 (n = 3/11/ 19/27)	グループ4 (n = 4/12/ 20/28)	グループ5 (n = 5/13/ 21/29)	グループ6 (n = 6/14/ 22/30)	グループ7 (n = 7/15/ 23/31)	
PORT_IRQ0	01h				01h				01h
:		①				②			
AGT0_AGTI				13h					1Bh
AGT0_AGTCMBI		13h							1Ah
AGT1_AGTI	06h				06h				1Bh
AGT1_AGTCMAI		05h				05h			1Ch
AGT0_AGTCMAI		04h				04h			1Dh

RE01 グループ (1.5M バイト フラッシュメモリ搭載製品) ユーザーズマニュアル ハードウェア編
割り込みコントローラユニット (ICU) より抜粋

上記、ユーザーズマニュアル ハードウェア編 割り込みコントローラユニット (ICU) から、以下のことがわかります

- ① AGT1_AGTI 割り込みは ICU のグループ 0 に属し、グループ 0 における AGT1_AGTI 割り込みのイベント番号は 0x06 (グループ 0 は IRQ0, IRQ8, IRQ16, IRQ24 に接続可能)
- ② AGT1_AGTI 割り込み ICU のグループ 4 にも属し、グループ 4 における AGT1_AGTI 割り込みのイベント番号は 0x06 (グループ 4 は IRQ4, IRQ12, IRQ20, IRQ28 に接続可能)

図 6-10 IRQ 番号の確認方法

6.3.3 r_system_cfg.h の編集

R_SYSTEM ドライバは、周辺機能の各割り込みと IRQ 番号を関連づける定義を `r_system_cfg.h` に用意しています。初期状態ではすべての周辺機能割り込みが”IRQ に割り当てない”に設定されています。割り込みを使用する場合は、動作環境にあわせて `r_system_cfg.h` の定義値を編集してください。

r_system_cfg.h : IRQ 番号の定義

- 内容： 周辺機能割り込みに対し IRQ 番号を関連づける
 - 定義名： SYSTEM_CFG_EVENT_NUMBER_XXX
“XXX”は周辺機能の割り込み名
 - 定義値：
 - SYSTEM_IRQ_EVENT_NUMBER_NOT_USED : 割り込みを IRQ に割り当てない（初期値）
 - SYSTEM_IRQ_EVENT_NUMBERn : 割り込みを IRQn に割り当てる (n=0~31)
複数の割り込みに対して、同一の IRQ 番号を割り当てることはできません
 - 設定例： AGT1 の割り込み AGT1_AGT1 を、IRQ0 に割り当てる場合の設定例を図 6-11 に示します

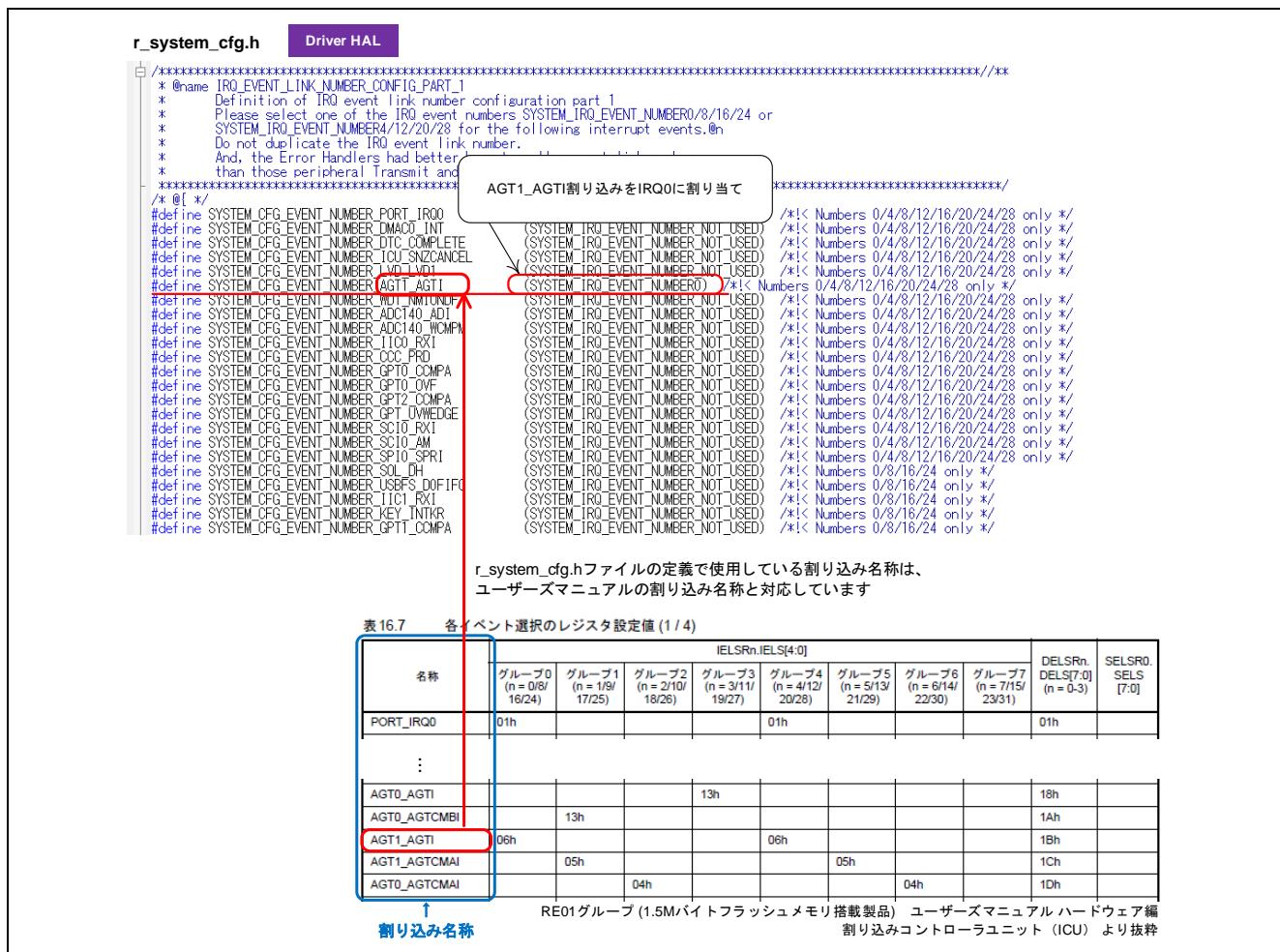


図 6-11 割り込み名称の確認方法と IRQ 番号の割り当て例

6.3.4 ドライバ関数

R_SYSTEM ドライバは、割り込み制御関数を用意しています。主な関数を説明します。

R_SYS_IrqEventLinkSet 関数 : 周辺機能割り込みと IRQn を接続し、コールバック関数を登録

第一引数 : IRQ 番号 (IRQn)

第二引数 : 周辺機能割り込みのイベント番号

第三引数 : コールバック関数のアドレス

R_SYS_IrqStatusClear 関数 : IRQn のステータスフラグをクリア

第一引数 : IRQ 番号 (IRQn)

R_NVIC_EnableIRQ 関数 : IRQn の割り込みを許可

第一引数 : IRQ 番号 (IRQn)

R_NVIC_SetPriority 関数 : IRQn の割り込み優先レベルを設定

第一引数 : IRQ 番号 (IRQn)

第二引数 : 優先度 (0:高～3:低)

R_NVIC_ClearPendingIRQ 関数 : IRQn の割り込み保留状態をクリア

第一引数 : IRQ 番号 (IRQn)

"R_NVIC" から始まる関数は、Arm®社が提供する NVIC 制御関数と同様の処理を行っています。低消費電力モード使用時に、内蔵フラッシュメモリの電源を遮断した後でも割り込み制御関数が使用できるよう、強制インライン関数として再定義した関数です。"R_NVIC" から始まる関数は、低消費電力モード以外でも使用可能です。

周辺機能ドライバを使用して割り込みを制御する場合 :

ユーザプログラムで、R_SYSTEM ドライバの割り込み制御関数を実行する必要はありません。

r_system_cfg.h に設定された IRQ 番号に従い、周辺機能ドライバ内で R_SYSTEM ドライバの割り込み制御関数を実行します。動作環境にあわせて r_system_cfg.h の定義値を編集してください。

各ドライバが使用する割り込みは、4 章で示す各ドライバ仕様書を参照してください。一例として、ドライバが使用する割り込み一覧を 10 付録の図 10-1 に示します。

周辺機能ドライバを使用せずに割り込みを制御する場合 :

ユーザプログラムで、R_SYSTEM ドライバの割り込み制御関数を実行してください。動作環境にあわせて r_system_cfg.h の定義値を編集し、割り込み制御関数実行時の IRQ 番号設定に使用することで、IRQ 番号を一括管理することができます。

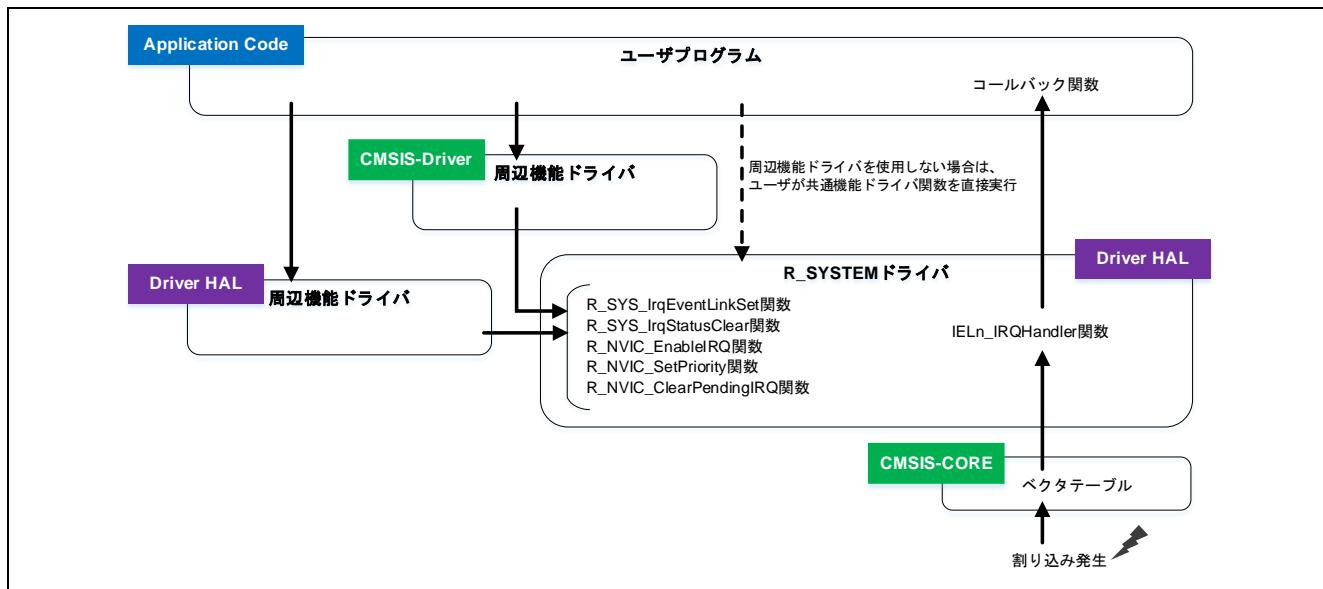


図 6-12 割り込み制御

6.4 クロック設定

6.4.1 クロック定義

R_CORE ドライバは、r_core_cfg.h にクロックの定義を用意しています。ユーザは動作環境にあわせて r_core_cfg.h の定義値を編集してください。

r_core_cfg.h におけるクロック関連の定義を図 6-13 に示します。

表 4-1 R_CORE コンフィグレーションの初期設定一覧			
章番号	コンフィグレーション	設定内容	初期値
4.1.1	SYSTEM_CFG_MOSC_ENABLE	メインクロック(MOSC)の動作/停止を設定	0
4.1.2	SYSTEM_CFG_MOSC_FREQUENCY_HZ	メインクロック(MOSC)の動作周波数を設定	32000000
4.1.3	SYSTEM_CFG_MOSC_DRIVE	メインクロック発振器駆動能力を設定	7
4.1.4	SYSTEM_CFG_MOSC_CLOCK_SOURCE	メインクロック発振器の発振源を設定	0
4.1.5	SYSTEM_CFG_MOSC_LOW_POWER_ENABLE	メインクロック発振器低消費発振機能を設定	0
4.1.6	SYSTEM_CFG_MOSC_WAIT_TIME	メインクロック発振器安定待ち時間を設定	5
4.1.7	SYSTEM_CFG_HOCO_ENABLE	高速オンチップオシレータ(HOCO)の動作/停止を設定	0
4.1.8	SYSTEM_CFG_HOCO_FREQUENCY	高速オンチップオシレータ(HOCO)の発振周波数を設定	0
4.1.9	SYSTEM_CFG_FLL_ENABLE 【注2】	FLL の有効/無効を設定	0
4.1.10	SYSTEM_CFG_MOCO_ENABLE	中速オンチップオシレータ(MOCO)の動作/停止を設定	1
4.1.11	SYSTEM_CFG_LOCO_ENABLE	低速オンチップオシレータ(LOCO)の動作/停止を設定	1
4.1.12	SYSTEM_CFG_SOSC_ENABLE	サブクロック(SOSC)の動作/停止を設定	0
4.1.13	SYSTEM_CFG_SOSC_DRIVE	サブクロック発振器駆動能力を設定	0
4.1.14	SYSTEM_CFG_SOSC_NF_STOP	サブクロック発振器ノイズフィルタの動作/停止を設定	0
4.1.15	SYSTEM_CFG_PLL_ENABLE 【注1】	PLL 回路の動作/停止を設定	0
4.1.16	SYSTEM_CFG_PLL_DIV 【注1】	PLL クロックソースの分周比を設定	1
4.1.17	SYSTEM_CFG_PLL_MUL 【注1】	PLL の周波数倍率を設定	1
4.1.19	SYSTEM_CFG_CLCOK_SOURCE	システムクロックのクロックソースを設定	1
4.1.20	SYSTEM_CFG_ICK_PCKA_DIV	システムクロック(iCLK)と周辺モジュールクロック A(PCLKA)の分周値を設定	0
4.1.21	SYSTEM_CFG_PCKB_DIV	周辺モジュールクロック B(PCLKB)を設定	0
4.1.22	SYSTEM_CFG_POWER_CONTROL_MODE	電力制御モードを設定	1

【注1】：このコンフィグレーションは 256KB フラッシュメモリ搭載製品向けドライバパッケージにはありません。
 【注2】：このコンフィグレーションは 1500KB フラッシュメモリ搭載製品向けドライバパッケージにはありません。

RE01 256KB グループ CMSIS パッケージ内 R_CORE ドライバ仕様書 コンフィグレーションより抜粋

R_CORE ドライバのスタートアップ処理：すべての定義を参照
 R_SYSTEM ドライバのクロック設定関数：黒文字の定義のみ参照
 青文字は、スタートアップ処理でのみ参照されます

図 6-13 r_core_cfg.h のクロック関連定義

6.4.2 ドライバ関数

R_SYSTEM ドライバは、クロック制御関数を用意しています。R_SYSTEM ドライバのクロック制御関数は、r_core_cfg.h の設定に従いクロックを制御します。

R_SYSTEM ドライバのクロック制御関数の一部を以下に示します。

- R_SYS_MainOscSpeedClockStart 関数 : メインクロックを発振
- R_SYS_MainOscSpeedClockStop 関数 : メインクロックを停止
- R_SYS_SystemClockMOSCSel 関数 : システムクロックにメインクロックを設定

その他、多数の関数があります。（詳細は、4 章に示す R_SYSTEM ドライバ仕様書を参照してください。）

6.5 端子設定

本デバイスは、周辺機能、割り込み、汎用入出力ポートで使用する端子を複数の端子から選択することができます。R_PIN ドライバは、pin.c に周辺機能で使用する端子を設定する関数を用意しています。リセット解除後は、一部端子を除き汎用入力ポートに設定されているため、動作環境にあわせて pin.c の関数処理を編集してください。

6.5.1 ドライバ関数

R_PIN ドライバは、周辺機能で使用する端子設定関数を pin.c ファイルに用意しています。本パッケージでは、pin.c ファイルのオブジェクトを RAM に配置しているため、低消費電力モードで内蔵フラッシュメモリの電源を遮断した後も端子設定関数を使用することができます。

オブジェクトの RAM 配置については 6.6 プログラムの RAM 配置 を参照してください。

R_XXX_Pinset_YYY 関数 : 周辺機能で使用する端子を設定

XXX : 周辺機能名

YYY : チャネル／機能名

R_XXX_Pinctr_YYY 関数 : 周辺機能で使用しなくなった端子を汎用入出力ポートに設定

XXX : 周辺機能名

YYY : チャネル／機能名

各周辺機能で使用する端子を表 6-3, 表 6-4 に示します。端子割り当ての詳細について、RE01 1500KB グループは「ユーザーズマニュアル ハードウェア編 マルチファンクションピンコントローラ (MPC) マルチ端子の割り当て一覧」を、RE01 256KB グループは「ユーザーズマニュアル ハードウェア編 IO ポート 製品ごとの周辺選択設定」を参照してください。

(1) RE01 1500KB グループ

表 6-3 ハードウェア機能で使用する端子

ハードウェア機能	対応 ドライバ	周辺 機能名	チャネル ／機能名	端子機能
端子割り込み	—	ICU	NMI	NMI
			CHn (n=0~9)	IRQn (n=0~9)
汎用 PWM タイマ (GPT)	—	GPT	CHn (n=0~5)	GTIOCnA, GTIOCnB (n=0~5)
			COM ^注 (共通)	GTETRGA, GTETRGB
			OPS ^注	GTIU, GTOULO, GTOUUP, GTIV, GTOVLO, GTOVUP, GTIW, GTOWLO, GTOWUP
非同期汎用タイマ (AGT)	—	AGT	CHn (n=0, 1)	AGTEEn, AGTIOOn, AGTOOn, AGTOAn, AGTOBn (n=0, 1)
シリアルコミュニケーションインターフェース (SCl _g 、SCl _i)	R_USART	SCI	CHn (n=0~5, 9)	CTS _n , RXD _n , SCK _n , TXD _n , RTS _n (n=0~5, 9)
シリアルペリフェラルインタフェース (SPI)	R_SPI	SPI	CH0	MISOA, MOSIA, RSPCKA, SSLAn (n=0~3)
			CH1	MISOB, MOSIB, RSPCKB, SSLBn (n=0~3)
クレッドシリアルペリフェラルインターフェース (QSPI)	—	QSPI	—	QSPCLK, QSSL, QIOn (n=0~3)
I2C バスインターフェース (RIIC)	R_I2C	RIIC	CHn (n=0, 1)	SCL _n , SDAn (n=0, 1)
クロック周波数精度測定回路 (CAC)	—	CAC	—	CACREF
14 ビット A/D コンバータ (S14AD)	R_ADC	S14AD	—	ADTRG0, AN0nn (nn=00~06, 16, 17, 20~28)
12 ビット D/A コンバータ (R12DA)	—	R12DA	—	DA0
アナログコンバータ (ACMP)	—	ACMP	—	CMPIN, CMPREF, VCOUT
MIP 液晶コントローラ (MLCD)	R_PMIP	MLCD	—	MLCD_DEN, MLCD_ENBG, MLCD_ENBS, MLCD_SCLK, MLCD_SIn (n=0~7), MLCD_VCOM, MLCD_XRST
キー割り込み機能 (KINT)	—	KINT	—	KRMnn (nn=00~07)
USB2.0FS ホスト／ファンクションモジュール (USB)	R_USB	USB	—	USB_EXICEN, USB_ID, USB_VBUS, USB_VBUSEN, USB_OVRCURA, USB_OVRCURB
リアルタイムクロック (RTC)	—	RTC	—	RTCICh (n=0~2), RTCOUT
クロック補正回路 (CCC)	—	CCC	—	CCCOUT
低速パルスジェネレータ (LPG)	—	LPG	—	LPGOUT
8 ビットタイマ (TMR)	—	TMR	—	TMCl _n , TMOn, TMRIn (n=0, 1)
クロック発生回路 (CLKOUT)	—	CLKOUT	—	CLKOUT, CLKOUT32
LED ドライバ (LED)	—	LED	—	LEDn (n=1~3)

【注】関数名の「周辺機能名」と「チャネル/機能名」が逆になります。

R_COM_Pinset_GPT, R_COM_Pinclr_GPT

R_OPS_Pinset_GPT, R_OPS_Pinclr_GPT

(2) RE01 256KB グループ

表 6-4 ハードウェア機能で使用する端子

ハードウェア機能	対応 ドライバ	周辺 機能名	チャネル ／機能名	端子機能
端子割り込み	—	ICU	NMI	NMI
			CHn (n=0~9)	IRQn (n=0~9)
汎用 PWM タイマ (GPT)	—	GPT	CHn (n=0~5)	GTIOCnA, GTIOCnB (n=0~5)
			COM ^注 (共通)	GTETRGA, GTETRGB
			OPS ^注	GTIU, GTOULO, GTOUUP, GTIV, GTOVLO, GTOVUP, GTIW, GTOWLO, GTOWUP
非同期汎用タイマ (AGT)	—	AGT	CHn (n=0, 1)	AGTEEn, AGTIOn, AGTOOn, AGTOAn, AGTOBn (n=0, 1)
非同期汎用タイマ (AGTW)	—	AGTW	CHn (n=0, 1)	AGTWEEn, AGTWIOn, AGTWOn, AGTWOAn, AGTWOBn (n=0, 1)
シリアルコミュニケーションインターフェース (SCIg、SCl)	R_USART	SCI	CHn (n=0~5, 9)	CTS _n , RXD _n , SCK _n , TXD _n , RTS _n (n=0~5, 9)
シリアルペリフェラルインタフェース (SPI)	R_SPI	SPI	CH0	MISOA, MOSIA, RSPCKA, SSLAn (n=0~3)
			CH1	MISOB, MOSIB, RSPCKB, SSLBn (n=0~3)
クレッドシリアルペリフェラルインターフェース (QSPI)	—	QSPI	—	QSPCLK, QSSL, QIO _n (n=0~3)
I2C バスインタフェース (IIC)	R_I2C	IIC	CHn (n=0, 1)	SCL _n , SDAn (n=0, 1)
クロック周波数精度測定回路 (CAC)	—	CAC	—	CACREF
14 ビット A/D コンバータ (ADC14)	R_ADC	ADC14	—	ADTRG0, AN0nn (nn=00~07, 16, 17, 20, 21)
MIP 液晶コントローラ (MLCD)	R_PMIP	MLCD	—	MLCD_DEN, MLCD_ENBG, MLCD_ENBS, MLCD_SCLK, MLCD_SIn (n=0~7), MLCD_VCOM, MLCD_XRST
キー割り込み機能 (KINT)	—	KINT	—	KRMnn (nn=00~07)
リアルタイムクロック (RTC)	—	RTC	—	RTCI _n (n=0~2), RTCOUT
クロック補正回路 (CCC)	—	CCC	—	CCCO _n
低速パルスジェネレータ (LPG)	—	LPG	—	LPGOUT
8 ビットタイマ (TMR)	—	TMR	—	TMCIn, TMOn, TMRIn (n=0, 1)
ウェイクアップタイマ (WUPT)	—	WUPT	—	TMWO
クロック発生回路 (CLKOUT)	—	CLKOUT	—	CLKOUT, CLKOUT32

【注】関数名の「周辺機能名」と「チャネル/機能名」が逆になります。

R_COM_Pinset_GPT, R_COM_Pinclr_GPT
R_OPS_Pinset_GPT, R_OPS_Pinclr_GPT

周辺機能ドライバを使用して端子設定を行う場合 :

周辺機能ドライバ内で R_PIN の端子制御関数を実行しているためユーザプログラムで、R_PIN ドライバの端子設定関数を実行する必要はありません。動作環境にあわせて pin.c の関数処理を編集してください。

周辺機能ドライバを使用せずに端子設定を行う場合 :

ユーザプログラムで端子制御を行ってください。動作環境にあわせて pin.c の関数処理を編集し、端子制御時に使用することで、使用端子の情報を一括管理することができます。

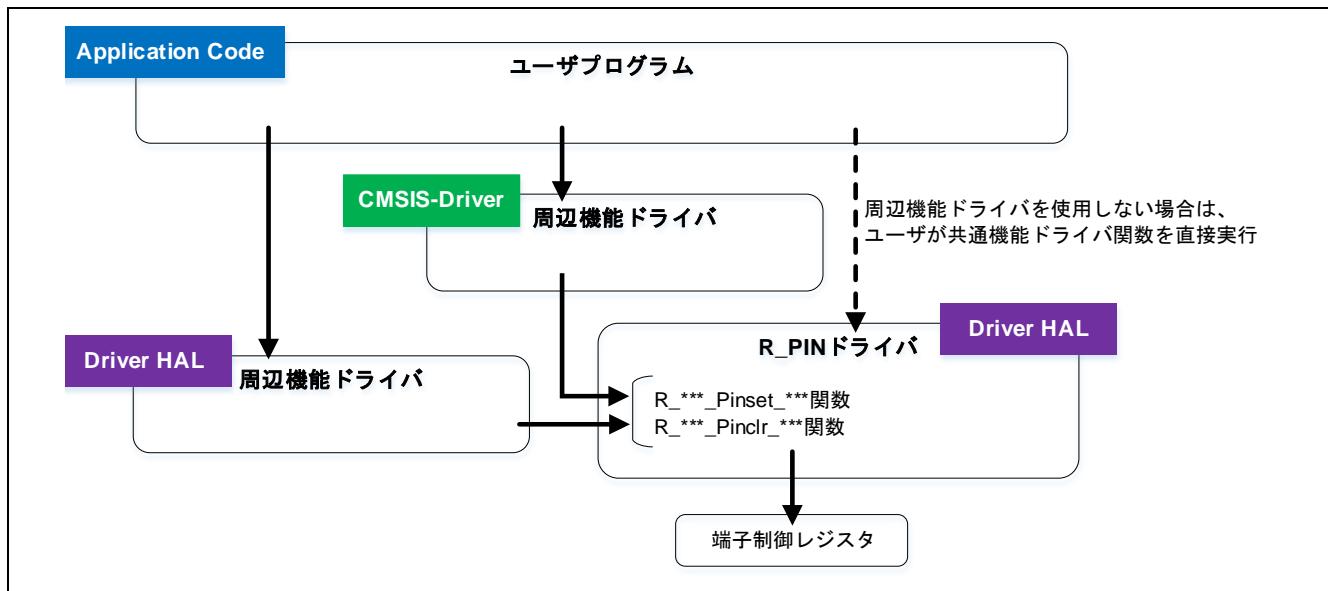


図 6-14 端子制御関数の呼び出し

6.5.2 ドライバ関数の編集

R_PIN ドライバ関数は、動作環境にあわせてユーザが編集して使用する関数です。各関数は、周辺機能で使用可能な端子の設定を記述しています。

シリアルコミュニケーションインターフェース SCI4 の送受信データ端子を以下のポートに割り当てて使用する場合の、R PIN ドライバ関数の編集例を図 6-15 に示します。

P812 : TXD4

P813 : RXD4

```

pin.c Driver HAL

/* **** This function sets Pin of SCI4. **** */
* @brief This function sets Pin of SCI4.
**** Function Name : R_SCI_Pinset_CH4 */
void R_SCI_Pinset_CH4(void) // @suppress("Function length")
{
    /* Disable protection for PFS function (Set to PMPR register) */
    R_SYS_RegisterProtectDisable(SYSTEM_REG_PROTECT_MPC);

    /* CTS4 : P111 */
    // PFS->P111PFS_b.ASEL = 0U; /* 0: Do not use as an analog pin, 1: Use as an analog pin. */
    // PFS->P111PFS_b.ISEL = 0U; /* 0: Do not use as an IRQn input pin, 1: Use as an IRQn input pin. */
    // PFS->P111PFS_b.PSEL = R_PIN_PRV_SCI_PSEL_04;
    // PFS->P111PFS_b.PMR = 10; /* 7: 0: Use the pin as a general I/O port, 1: Use the pin as a peripheral module. */

    /* P812をTXD4用端子に設定 */
    /* When using SCI in I2C mode, set the pin to NMOS Open drain. */
    // PFS->P203PFS_b.NODDR = 1U; /* 0: CMOS output, 1: NMOS open-drain output. */
    // PFS->P203PFS_b.PDDR = 0U; /* 0: CMOS output, 1: NMOS open-drain output. */
    // PFS->P203PFS_b.PSEL = R_PIN_PRV_SCI_PSEL_04;
    // PFS->P203PFS_b.PMR = 1U; /* 7: 0: Use the pin as a general I/O port, 1: Use the pin as a peripheral module. */

    /* TXD4 : P812 */
    PFS->P812PFS_b.ASEL = 0U; /* 0: Do not use as an analog pin, 1: Use as an analog pin. */
    PFS->P812PFS_b.ISEL = 0U; /* 0: Do not use as an IRQn input pin, 1: Use as an IRQn input pin. */

    /* When using SCI in I2C mode, set the pin to NMOS Open drain. */
    // PFS->P812PFS_b.NODDR = 1U; /* 0: CMOS output, 1: NMOS open-drain output. */
    // PFS->P812PFS_b.PDDR = 0U; /* 0: CMOS output, 1: NMOS open-drain output. */
    PFS->P812PFS_b.PSEL = R_PIN_PRV_SCI_PSEL_04;
    PFS->P812PFS_b.PMR = 10; /* 7: 0: Use the pin as a general I/O port, 1: Use the pin as a peripheral module. */

    /* RXD4 : P112 */
    // PFS->P112PFS_b.ASEL = 0U; /* 0: Do not use as an analog pin, 1: Use as an analog pin. */
    // PFS->P112PFS_b.ISEL = 0U; /* 0: Do not use as an IRQn input pin, 1: Use as an IRQn input pin. */

    /* P813をRXD4用端子に設定 */
    /* When using SCI in I2C mode, set the pin to NMOS Open drain. */
    // PFS->P202PFS_b.NODDR = 1U; /* 0: CMOS output, 1: NMOS open-drain output. */
    // PFS->P202PFS_b.PDDR = 0U; /* 0: CMOS output, 1: NMOS open-drain output. */
    // PFS->P202PFS_b.PSEL = R_PIN_PRV_SCI_PSEL_04;
    // PFS->P202PFS_b.PMR = 1U; /* 7: 0: Use the pin as a general I/O port, 1: Use the pin as a peripheral module. */

    /* RXD4 : P813 */
    PFS->P813PFS_b.ASEL = 0U; /* 0: Do not use as an analog pin, 1: Use as an analog pin. */
    PFS->P813PFS_b.ISEL = 0U; /* 0: Do not use as an IRQn input pin, 1: Use as an IRQn input pin. */

    /* When using SCI in I2C mode, set the pin to NMOS Open drain. */
    // PFS->P813PFS_b.NODDR = 1U; /* 0: CMOS output, 1: NMOS open-drain output. */
    // PFS->P813PFS_b.PDDR = 0U; /* 0: CMOS output, 1: NMOS open-drain output. */
    PFS->P813PFS_b.PSEL = R_PIN_PRV_SCI_PSEL_04;
    PFS->P813PFS_b.PMR = 10; /* 7: 0: Use the pin as a general I/O port, 1: Use the pin as a peripheral module. */

    /* SCK4 : P108 */
    // PFS->P108PFS_b.ASEL = 0U; /* 0: Do not use as an analog pin, 1: Use as an analog pin. */
    // PFS->P108PFS_b.ISEL = 0U; /* 0: Do not use as an IRQn input pin, 1: Use as an IRQn input pin. */
    // PFS->P108PFS_b.PSEL = R_PIN_PRV_SCI_PSEL_04;
    // PFS->P108PFS_b.PMR = 10; /* 7: 0: Use the pin as a general I/O port, 1: Use the pin as a peripheral module. */

    /* SCK4 : P204 */
    // PFS->P204PFS_b.ASEL = 0U; /* 0: Do not use as an analog pin, 1: Use as an analog pin. */
    // PFS->P204PFS_b.ISEL = 0U; /* 0: Do not use as an IRQn input pin, 1: Use as an IRQn input pin. */
    // PFS->P204PFS_b.PSEL = R_PIN_PRV_SCI_PSEL_04;
    // PFS->P204PFS_b.PMR = 10; /* 7: 0: Use the pin as a general I/O port, 1: Use the pin as a peripheral module. */

    /* SCK4 : P814 */
    // PFS->P814PFS_b.ASEL = 0U; /* 0: Do not use as an analog pin, 1: Use as an analog pin. */
    // PFS->P814PFS_b.ISEL = 0U; /* 0: Do not use as an IRQn input pin, 1: Use as an IRQn input pin. */
    // PFS->P814PFS_b.PSEL = R_PIN_PRV_SCI_PSEL_04;
    // PFS->P814PFS_b.PMR = 10; /* 7: 0: Use the pin as a general I/O port, 1: Use the pin as a peripheral module. */

    /* Enable protection for PFS function (Set to PMPR register) */
    R_SYS_RegisterProtectEnable(SYSTEM_REG_PROTECT_MPC);

    /* End of function R_SCI_Pinset_CH4() */
}

```

図 6-15 端子設定閾数の編集例

表24.1 マルチ端子の割り当て一覧 (4 / 9)

モジュール／機能	チャネル	端子機能	割り当てポート	パッケージ		
				156ピン WLBGA	144ピン	100ピン
シリアルコミュニケーションインターフェース (SCIg, SCII)	SCI0	CTS0_A	P107	○	○	○
		CTS0_B	P500	○	○	○
		CTS0_C	P704	○	○	○
⋮						
シリアルコミュニケーションインターフェース (SCIg, SCII)	SCI4	CTS4_A	P111	○	○	○
		CTS4_B	P205	○	○	○
		CTS4_C	P815	○	○	×
⋮						
シリアルコミュニケーションインターフェース (SCIg, SCII)	SCI4	RXD4_A	P112	○	○	○
		RXD4_B	P202	○	○	○
		RXD4_C	P813	○	○	×
		SCK4_A	P108	○	○	○
		SCK4_B	P204	○	○	○
		SCK4_C	P814	○	○	×
		TXD4_A	P113	○	○	○
		TXD4_B	P203	○	○	○
		TXD4_C	P812	○	○	×
		CTS5_A	P109	○	○	○

RE01グループ (1.5Mバイトフラッシュメモリ搭載製品) ユーザーズマニュアルハードウェア編
マルチファンクションピンコントローラ (MPC) より抜粋

上記、ユーザーズマニュアルハードウェア編 マルチピンファンクションコントローラ (MPC) から、以下のことがわかりります

SCI4で使用する端子は下記から選択することができる

- ・CTS用端子 : P111, P205 P815
- ・RXD用端子 : P112, P202, P813
- ・SCK用端子 : P108, P204, P814
- ・TXD用端子 : P113, P203, P812

【注】 RE01 256KB グループはユーザーズマニュアル ハードウェア編 表 22.4～表 22.18 入出力端子機能のレジスタ設定を参照してください。

図 6-16 周辺機能で使用できる端子の確認方法

6.6 プログラムの RAM 配置

本デバイスは、内蔵フラッシュメモリの電源を遮断することで消費電力を低減することができます。

内蔵フラッシュメモリの電源を遮断した後に RAM に展開したプログラムで動作をさせる場合は、任意のプログラムを RAM に配置する必要があります。

本パッケージで行っている、任意のプログラムを RAM に配置する方法を紹介します。

1. RAM 配置セクションによる RAM 配置方法

各ドライバが行っている方法で、コンフィグレーション定義ヘッダにおける RAM 配置定義に従い、関数単位で配置場所を設定します。

R_PIN ドライバだけはオブジェクト単位で配置場所を設定します。

また標準関数でも使用可能な方法です。

2. 強制インライン展開による RAM 配置方法

R_SYSTEM ドライバの “R_NVIC” から始まる割り込み制御関数で行っている方法です。

RAM に配置されたプログラムから実行された場合は RAM にインライン展開されますが、内蔵フラッシュメモリに配置されたプログラムから実行された場合は内蔵フラッシュメモリにインライン展開されます。

6.6.1 RAM 配置セクションによる RAM 配置方法

RAM 配置セクションによる RAM 配置の設定手順を説明します。

手順 1. リンカファイルにて RAM 配置セクションを定義

(RE01 1500KB グループ CMISIS パッケージで定義しているセクションを、図 6-17 に示します)

手順 2. 任意のプログラムを RAM 配置セクションに割り当てる

手順 3. リセット解除後に RAM 配置セクションに割り当てられたプログラムを RAM に展開

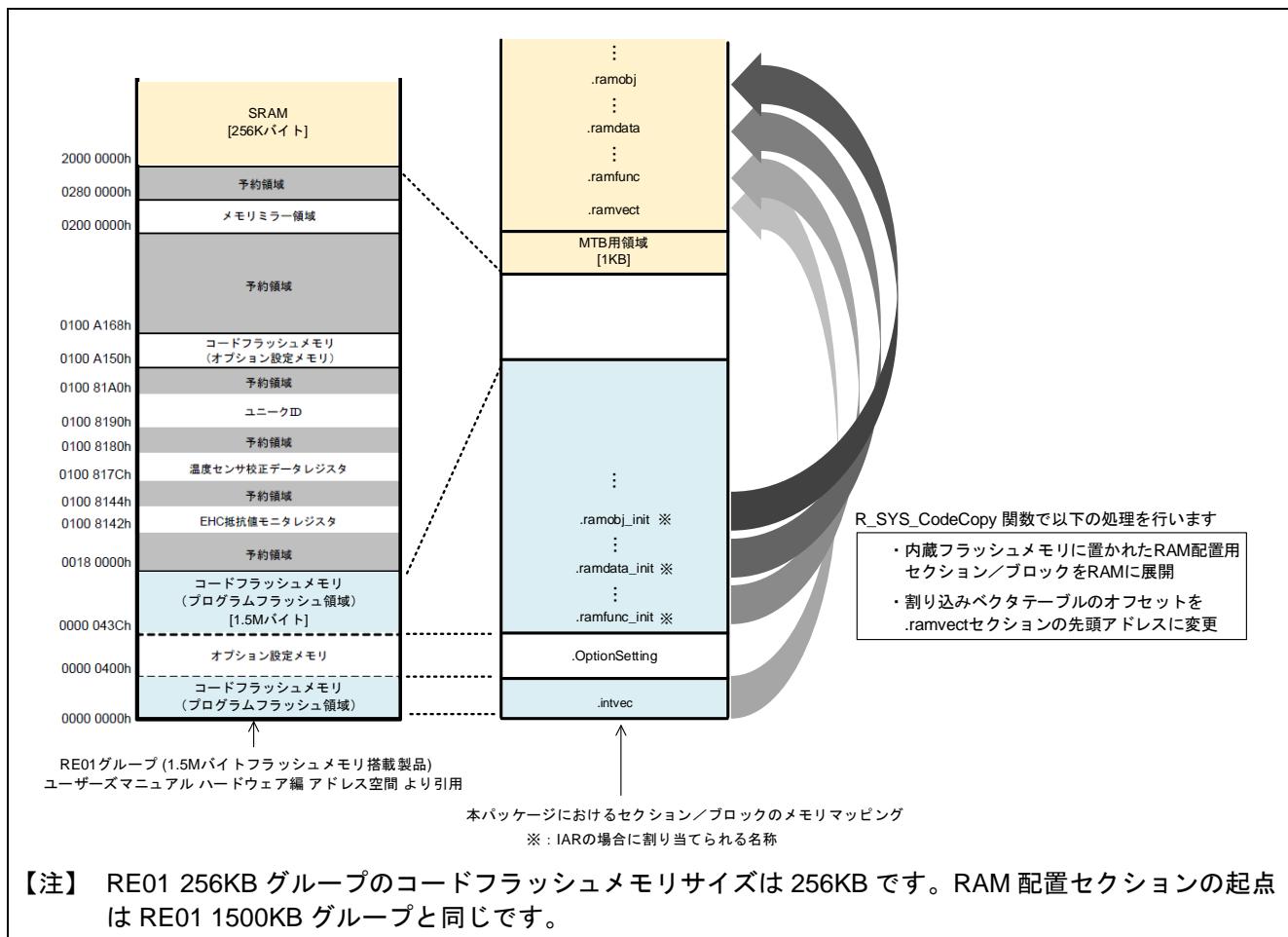


図 6-17 RAM 配置セクションを使用した場合のメモリマッピング

(1) リンカファイルにて RAM 配置セクションを定義

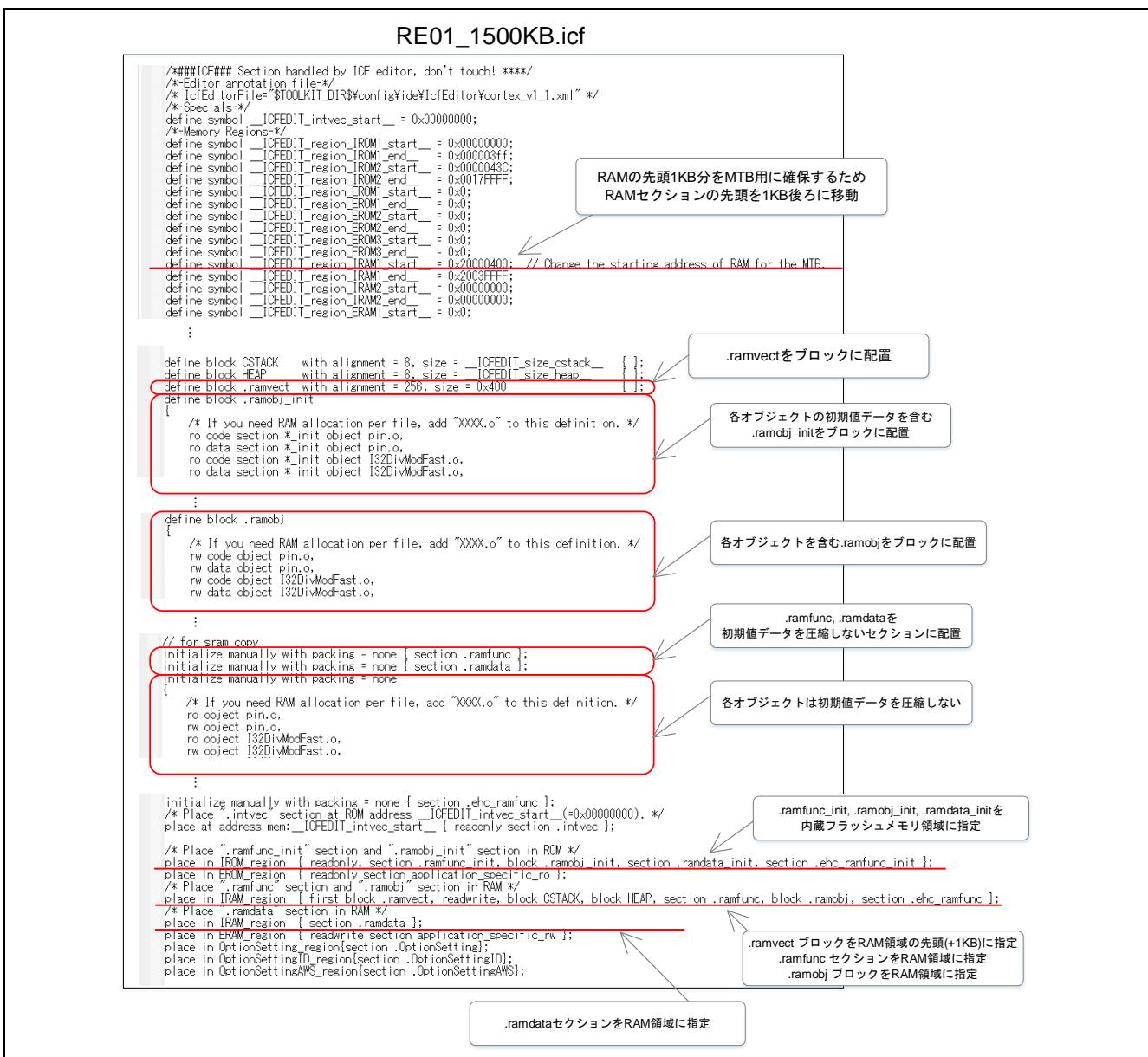
本パッケージに同梱されているリンカファイルで定義している RAM 配置用セクションを表 6-5 に示します。

表 6-5 RAM 配置用セクション

セクション名	内容	割り当て状況
.ramobj	オブジェクトの RAM 配置用セクション	R_PIN ドライバ
.ramdata	変数の RAM 配置用セクション	ドライバ関数で使用する変数
.ramfunc	関数の RAM 配置用セクション	ドライバ関数
.ramvect	ベクターテーブルの RAM 配置用セクション	ベクターテーブル

R_PIN ドライバは、オブジェクトファイル pin.o をリンカファイルにて.ramobj に割り当てています。リンカファイルで pin.o を指定しているため、pin.c のファイル名は変更しないでください。

本パッケージに同封されているリンカファイルの設定を図 6-18、図 6-19 に示します。



【注】 RE01 256KB グループは IROM2, IRAM2 の終了アドレスが異なりますが説明箇所は同じ仕様です。

図 6-18 リンカファイルでの設定例 (IAR コンパイラ)

RE01_1500KB.ld

```

/*
   Linker Script File for R7F0E015/R7F0E017 parts
*/
/* Linker script to configure memory regions. */
MEMORY
[ 
  FLASH (rx)      : ORIGIN = 0x00000000, LENGTH = 0x01000000 /* 1.5M */
  OSFS (rx)       : ORIGIN = 0x0100A150, LENGTH = 0x00000018 /* 24B */
  E2S_TRACE_BUF (rw) : ORIGIN = 0x20000000, LENGTH = 0x00004000 /* 1K */
  RAM (rwx)       : ORIGIN = 0x20000400, LENGTH = 0x003FC000 /* 255K */
  USPT_FLASH (rx) : ORIGIN = 0x60000000, LENGTH = 0x80000000 /* 128M */
]
;

SECTIONS
{
  .text :
  [
    _Vectors_Start = .;
    KEEP(*(.intvec))
    KEEP(*(.sort_by_name(.intvec)))
    _Vectors_End = .;
    _end = .;
  ]
  /* ROM Registers start at address 0x00000400 */
  . = 0x400;
  _optionSetting_start = .;
  REEP(*(.optionSetting))
  KEEP(*(.optionSetting))
  _optionSetting_end = .;
  /* Reserving 0x100 bytes of space for ROM registers. */
  . = . + 0x100;
  *EXCLUDE_FILE(*pin.o *libgcc.a:* *libc.a*) .text*
  KEEP(*(.version))
  KEEP(*(.init))
  KEEP(*(.fini))

  /* .ctors */
  *crtbegin.o(.ctors)
  *crtbegini.o(.ctors)
  *EXCLUDE_FILE(*crtend2.o *crtend.o) .ctors
  *(.sort(.ctors))
  *(.ctors)

  /* .dtors */
  *crtbegin.o(.dtors)
  *crtbegini.o(.dtors)
  *EXCLUDE_FILE(*crtend2.o *crtend.o) .dtors
  *(.sort(.dtors))
  *(.dtors)

  /* (EXCLUDE_FILE(*pin.o *libgcc.a:* *libc.a*) .rodata*)
  ROM_End = .;
] > FLASH = 0xFF
;

/* secure vector area on RAM to be copied from ROM */
[.ramvect :
  . = ALIGN(256);
  .ramvect_start = .;
  . = . + 0x400;
  . = ALIGN(4);
] > RAM
;

.ramfunc :
[.ramfunc_start = .;
  REEP(*(.ramfunc))
  . = ALIGN(4);
  .ramfunc_end = .;
] > RAM AT> FLASH
._ramfunc_init_start = LOADADDR(.ramfunc);

.ehc_ramfunc :
[.ehc_ramfunc_start = .;
  REEP(*(.ehc_ramfunc))
  . = ALIGN(4);
  .ehc_ramfunc_end = .;
] > RAM AT> FLASH
._ehc_ramfunc_init_start = LOADADDR(.ehc_ramfunc);

.ramdata :
[.ramdata_start = .;
  REEP(*(.ramdata))
  . = ALIGN(4);
  .ramdata_end = .;
] > RAM AT> FLASH
._ramdata_init_start = LOADADDR(.ramdata);

.ramobj :
[.ramobj_start = .;
  REEP(*(.ramobj))
  REEP(*(.text*.text))
  REEP(*(.text*.text))
  REEP(*(.text*.text))
  REEP(*(.text*.text))
  . = ALIGN(4);
  .ramobj_end = .;
] > RAM AT> FLASH
._ramobj_init_start = LOADADDR(.ramobj);

.poinit (NOLOAD):
[. = ALIGN(4);
  .poinit_start = .;
  REEP(*(.poinit))
  .poinit_end = .;
] > RAM AT> RAM
;
```

RAMの先頭1KB分をMTB用に確保するため
RAMセクションの先頭を1KB後ろに移動

オブジェクトおよびアセンブラファイルを除外

オブジェクトおよびアセン布拉ファイルを除外

.ramvectセクションをRAMに配置し
セクションの各アドレスを保存

.ramfuncセクションを内蔵フラッシュメモリに配置し
RAMで実行セクションの各アドレスを保存

.ramdataセクションを内蔵フラッシュメモリに配置し
RAMで実行セクションの各ドレスを保存

各オブジェクトおよびアセン布拉ファイルを含む
.ramobjセクションを内蔵フラッシュメモリに配置し
RAMで実行セクションの各アドレスを保存

【注】 RE01 256KB グループは FLASH, RAM のサイズが異なりますが説明箇所は同じ仕様です。

図 6-19 リンカファイルでの設定例 (GCC コンパイラ)

(2) 任意のプログラムを RAM 配置用セクションに割り当てる

プログラムを RAM に配置するために、任意のプログラムを RAM 配置用セクションに指定します。本パッケージに同梱されているドライバは、各ドライバ関数の RAM 配置用セクション指定を容易にするための仕組みが入っています。

➤ ドライバを RAM 配置用セクションに設定する方法：

各ドライバは、コンフィグレーション定義ヘッダの定義値に従い、任意のプログラムを RAM 配置用セクションに指定しています。

プログラムの RAM 配置定義：

- 内容： ドライバ関数の RAM 配置の有無を設定
- 定義名： XXX_CFG_SECTION_YYY
XXX：ドライバ名
YYY：関数名（すべて大文字）
- 定義値：
SYSTEM_SECTION_CODE：関数を内蔵フラッシュメモリのコード用セクションに割り当てる
SYSTEM_SECTION_RAM_FUNC：関数を RAM 配置用セクションに割り当てる
- 設定例： R_SYSTEM ドライバ関数の設定例を図 6-20 に示します

```

r_system_cfg.h   Driver HAL

/* ****@[ */ *!/***** @[ */
* @name R_SYSTEM_API_LOCATION_CONFIG
* Definition of R_System API location configuration.
* Please select "SYSTEM_SECTION_CODE" or "SYSTEM_SECTION_RAM_FUNC".
* ****@[ */ */

/* @[ */
#define SYSTEM_CFG_SECTION_R_SYS_INITIALIZE (SYSTEM_SECTION_CODE) /*!< R_SYS_Initialize関数を内蔵フラッシュメモリに配置 */
#define SYSTEM_CFG_SECTION_R_SYS_BOOSTSPEEDMODESET (SYSTEM_SECTION_CODE) /*!< R_SYS_HighSpeedModeSet関数をRAMに配置 */
#define SYSTEM_CFG_SECTION_R_SYS_HIGHSPEDMODESET (SYSTEM_SECTION_CODE) /*!< R_SYS_LowSpeedModeSet() section */
#define SYSTEM_CFG_SECTION_R_SYS_LOWSPEDMODESET (SYSTEM_SECTION_RAM_FUNC) /*!< R_SYS_32kHzSpeedModeSet() section */
#define SYSTEM_CFG_SECTION_R_SYS_32KHZSPEEDMODESET (SYSTEM_SECTION_RAM_FUNC) /*!< R_SYS_SpeedModeGet() section */
#define SYSTEM_CFG_SECTION_R_SYS_SPEEDMODEGET (SYSTEM_SECTION_RAM_FUNC) /*!< R_SYS_SystemClockHOCOSet() section */
#define SYSTEM_CFG_SECTION_R_SYS_SYSTEMCLOCKHOCOSET (SYSTEM_SECTION_RAM_FUNC) /*!< R_SYS_SystemClockMOCOSet() section */
#define SYSTEM_CFG_SECTION_R_SYS_SYSTEMCLOCKMOCOSET (SYSTEM_SECTION_RAM_FUNC) /*!< R_SYS_SystemClockLCOSet() section */
#define SYSTEM_CFG_SECTION_R_SYS_SYSTEMCLOCKLOCSET (SYSTEM_SECTION_RAM_FUNC) /*!< R_SYS_SystemClockMOSCSet() section */
#define SYSTEM_CFG_SECTION_R_SYS_SYSTEMCLOCKMOSCSET (SYSTEM_SECTION_RAM_FUNC) /*!< R_SYS_SystemClockSOSCSet() section */
#define SYSTEM_CFG_SECTION_R_SYS_SYSTEMCLOCKSOSCSET (SYSTEM_SECTION_RAM_FUNC) /*!< R_SYS_SystemClockPLLSet() section */
#define SYSTEM_CFG_SECTION_R_SYS_SYSTEMCLOCKPLLSET (SYSTEM_SECTION_CODE) /*!< R_SYS_SystemClockFreqGet() section */
#define SYSTEM_CFG_SECTION_R_SYS_SYSTEMCLOCKFREQGET (SYSTEM_SECTION_RAM_FUNC) /*!< R_SYS_PeripheralClockFreqGet() section */
#define SYSTEM_CFG_SECTION_R_SYS_PERIPHERALCLOCKFREQGET (SYSTEM_SECTION_RAM_FUNC) /*!< R_SYS_SystemClockDividerSet() section */
#define SYSTEM_CFG_SECTION_R_SYS_SYSTEMCLOCKDIVIDERSET (SYSTEM_SECTION_RAM_FUNC) /*!< R_SYS_MainOscSpeedClockStart() section */
#define SYSTEM_CFG_SECTION_R_SYS_MAINOSSPEEDCLOCKSTART (SYSTEM_SECTION_RAM_FUNC) /*!< R_SYS_MainOscSpeedClockStop() section */
#define SYSTEM_CFG_SECTION_R_SYS_HIGHSPEEDCLOCKSTART (SYSTEM_SECTION_RAM_FUNC) /*!< R_SYS_HighSpeedClockStart() section */
#define SYSTEM_CFG_SECTION_R_SYS_HIGHSPEEDCLOCKSTOP (SYSTEM_SECTION_RAM_FUNC) /*!< R_SYS_HighSpeedClockStop() section */

```

図 6-20 ドライバ関数を配置するセクションの設定例

➤ ユーザプログラムを RAM 配置用セクションに設定する方法 :

関数および変数のプロトタイプ宣言『`__attribute__((section("xxxx")))`』を使用し、任意のプログラムを RAM 配置用セクションに指定することができます。

以下に、任意のセクションに変数および関数を配置する設定例を示します。
(IAR コンパイラ、GCC コンパイラ共に使用可能)

- 変数 `sample_data` : 変数の RAM 配置用セクション".ramdata"に配置
- 関数 `sample_func` : 関数の RAM 配置用セクション".ramfunc"に配置

✓ `__attribute__((section("xxx")))`を使用しセクションを指定する設定例

```
static const int32_t sample_data __attribute__ ((section(".ramdata")));
static void sample_function(void) __attribute__ ((section(".ramfunc")));
```

✓ `__attribute__((noinline))`を使用し関数のインライン化を抑制する設定例

コンパイラの最適化によって、関数がインライン展開され、期待どおりに RAM に配置されない場合があります。インライン展開を禁止する必要のある関数では、『`__attribute__((noinline))`』を使用し、インライン化を抑制することができます。

```
static void sample_function(void) __attribute__ ((section(".ramfunc")))
__attribute__ ((noinline));
```

➤ 標準関数を RAM 配置用セクションに指定する方法 :

`stdio.h` の標準関数や除算、浮動小数点演算など、CPU が標準でサポートできない命令はコンパイラに同梱される標準関数で実行されます。この標準関数を RAM 配置用セクションに指定する方法を説明します。

標準関数はオブジェクトファイル(*.o)として生成されます。このオブジェクトファイルをリンクファイルにてオブジェクトの RAM 配置用セクション (.ramobj) に割り当てます。本パッケージのリンクスクリプトに予め追加するためのセクションを設けています。図 6-18~図 6-19 を参考に追加してください。`pin.o` ファイルが同様の方法で RAM 配置用セクションに割り当てられていますので、`pin.o` を参考に RAM 配置用セクションに割り当てたいオブジェクトファイルを追記してください。

(3) リセット解除後に RAM 配置セクションに割り当てられたプログラムを RAM に展開

リセット解除後、任意のプログラムを RAM に展開するために、R_SYSTEM ドライバの R_SYS_CodeCopy 関数を実行します。

R_SYS_CodeCopy 関数

- 内容： RAM 配置用セクションに割り当てた関数を内蔵フラッシュメモリから RAM に展開する関数
- 実行タイミング：
R_SYS_CodeCopy 関数はリセット解除後、RAM 配置用セクションに割り当てた関数を実行する前に 1 度だけ実行してください。（複数回呼び出す必要はありません）
R_SYSTEM ドライバの初期化関数、R_SYS_Initialize 関数より先に実行してください。
- 設定例： 本関数の使用例を図 6-21 に示します

```

ユーザプログラム Application Code



```

/* Function Name : main
 * Description : main function
 * Arguments : none
 * Return Value : none

int main (void)
{
 R_SYS_CodeCopy(); // RAM配置用セクション領域をRAMに展開
 R_SYS_Initialize();
 R_LPM_Initialize();
 R_LPM_IOPowerSupplyModeSet (LPM_IOPOWER_SUPPLY_NONE);

 /* USART Driver Setup */
 gsp_usart_drv->Initialize(usart_callback);
 gsp_usart_drv->PowerControl (ARM_POWER_FULL);

 gsp_usart_drv->Control (ARM_USART_MODE_ASYNCHRONOUS |
 ARM_USART_DATA_BITS_8 |
 ARM_USART_PARITY_NONE |
 ARM_USART_STOP_BITS_1 |
 ARM_USART_FLOW_CONTROL_NONE,
 9600);

 gsp_usart_drv->Control (ARM_USART_CONTROL_TX, 1);
 gsp_usart_drv->Send(&gs_send_data[0], sizeof(gs_send_data));

 while (1)
 {
 ; /* main loop */
 }
 return 0;
} /* End of function main() */

```


```

図 6-21 プログラムの RAM 展開関数使用例

6.6.2 強制インライン展開による RAM 配置方法

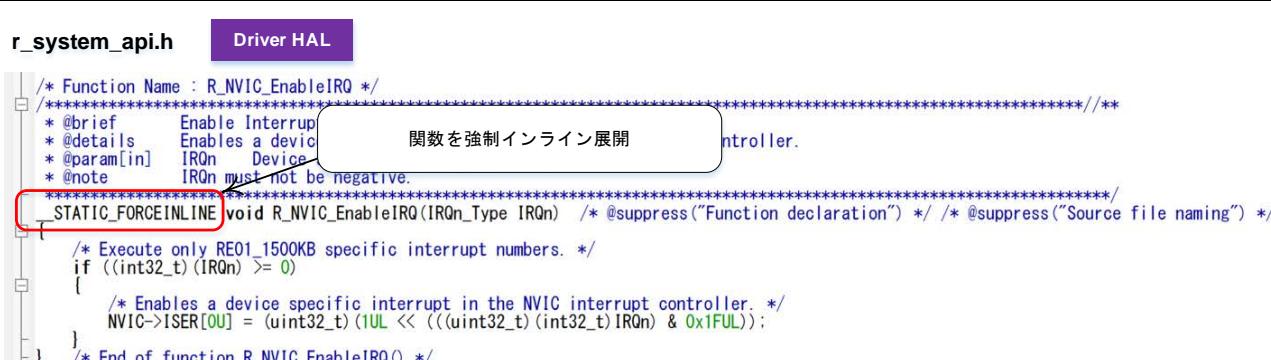
インライン関数は、コンパイラの最適化等によりインライン展開されない場合があるため、強制インライン展開の手法をとります。

RAM に配置されたプログラムから実行された場合、強制的にインライン展開することでプログラムを確実に RAM に展開することができます。

この方法は、R_SYSTEM ドライバの “R_NVIC” で始まる関数で使用しています。

CMSIS の標準関数にも同様の NVIC 関数が存在しますが、プログラムを RAM に配置する必要がある場合は、“R_NVIC” で始まるルネサス独自の関数を使用してください。

強制的にインライン展開する場合は、R_SYSTEM ドライバの設定を参考にしてください。



```

r_system_api.h      Driver HAL

/* Function Name : R_NVIC_EnableIRQ */
//*********************************************************************//**

* @brief      Enable Interrupt
* @details    Enables a device
* @param[in]   IRQn_Type IRQn
* @note       IRQn must not be negative.
*             Device controller.

STATIC_FORCEINLINE void R_NVIC_EnableIRQ(IRQn_Type IRQn) /* @suppress("Function declaration") */ /* @suppress("Source file naming") */

/* Execute only RE01_1500KB specific interrupt numbers. */
if ((int32_t)(IRQn) >= 0)
{
    /* Enables a device specific interrupt in the NVIC interrupt controller. */
    NVIC->ISER[0U] = (uint32_t)(1UL << (((uint32_t)(int32_t)IRQn) & 0x1FUL));
}
/* End of function R_NVIC_EnableIRQ() */

```

図 6-22 関数の強制インライン展開設定例

7. ユーザプログラムを作ってみよう

本章は RE01 1500KB グループを例に説明します。以降、特に記述がない限り、その手順は RE01 256KB グループも同じです。

7.1 ユーザプログラム作成準備

ユーザプログラムを作成する場合、特定関数の作成やドライバのコンフィグレーション定義ヘッダを編集するなどの準備が必要です。プログラム実行前までに必要な準備を説明します。

各処理の実施順番は問いません。プログラム実行前に動作環境にあわせて準備をしてください。

表 7-1 ユーザプログラム作成準備

周辺機能 ドライバを使用する場合	周辺機能 ドライバを使用しない場合
スタートアップ処理の準備	
① 動作開始時の端子設定 (ユーザ作成関数 : BoardInit 関数) ② 動作開始時のクロック・電力制御モードの定義	
共通機能 ドライバの準備	
③ IRQ 番号の定義 ④ 周辺機能で使用する端子の設定 ⑤ 共通機能 ドライバの動作条件の定義 ⑥ R_SYSTEM ドライバで使用するクロックの定義	
周辺機能 ドライバの準備	
⑦ 周辺機能 ドライバの動作条件の定義	—

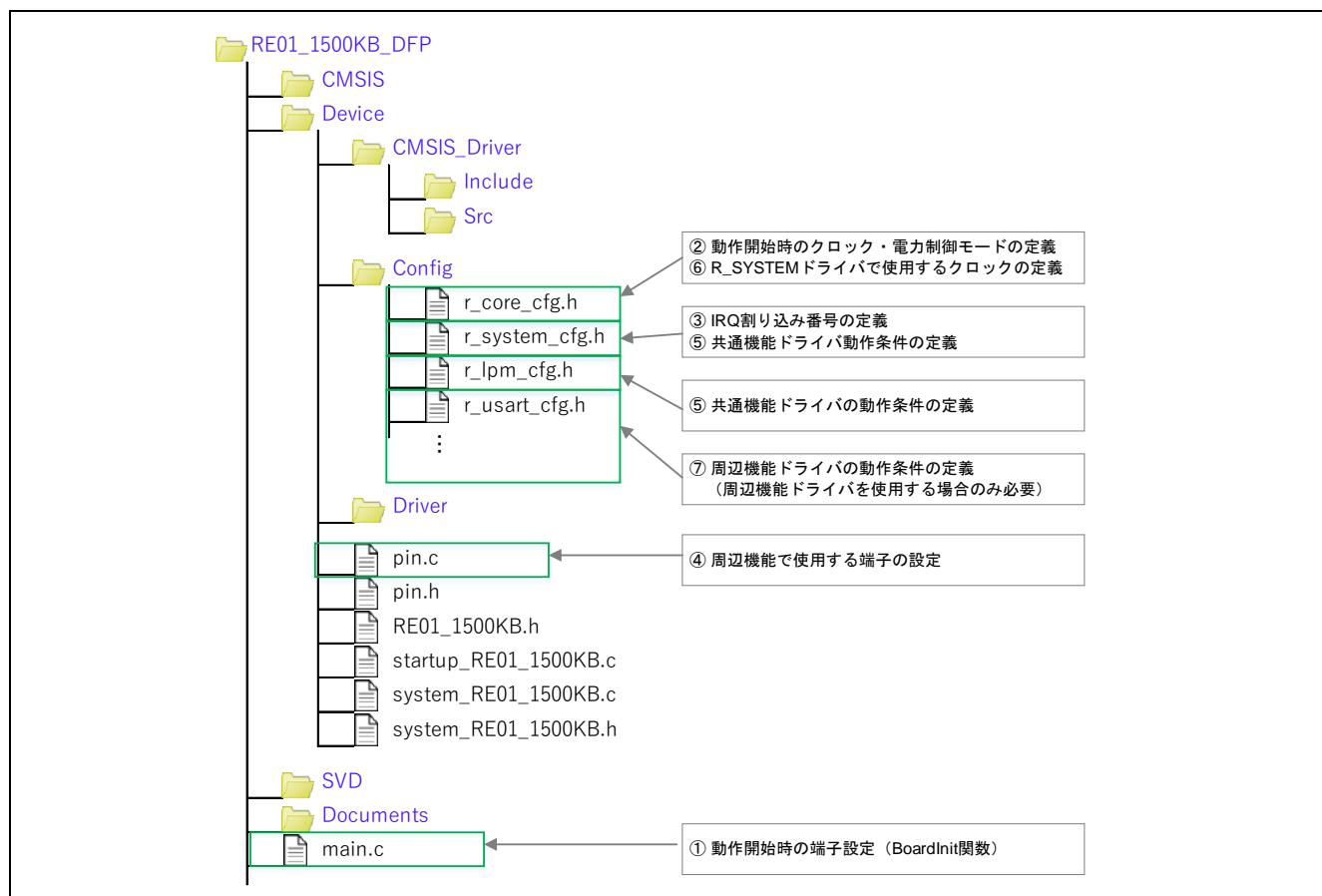


図 7-1 ユーザプログラム作成準備における編集対象ファイル

7.1.1 スタートアップ処理の準備

リセット解除後、main 関数実行までに行われるスタートアップ処理では、任意の定義および関数が使用されます。動作環境にあわせて編集または作成してください。

7.1.1.1 動作開始時の端子設定

スタートアップ処理では、ユーザ作成関数である BoardInit 関数を実行します。リセット解除後、早い段階で端子設定が必要な場合は、BoardInit 関数を作成し端子処置を行ってください。端子の初期設定については、6.1.1 動作開始時の端子設定 を参照してください。

BoardInit 関数による設定例を図 7-2 に示します。

```

ユーザプログラム Application Code

/*
 * Function Name: BoardInit
 * Description : Configure board initial setting.
 *               This function is called by SystemInit() function in system_RE01_1500KB.c file.
 *               This is reference to perform Boardinit process. Sample code of target is E015 SDK board(RTK70E015D)
 *               on Renesas. Please modify this function to suit your board.
 * Arguments   : none
 * Return Value: none
 */
void BoardInit(void)
{
    /* ***** This function performs at beginning of start-up after released reset pin *****/
    /* ***** Please set pins here if your board is needed pins setting at the device start-up. *****/
    /* ***** This function is suiting RE01_1500KB SDK board. Please change to your board pin setting *****

    /* Handling of Unused ports (IOVCC domain) */
    /* PORT4 Setting: RE01_1500KB SDK has DCDCs. Those are connect P404 and P405. */
    /* This perform to disable those output.
    /* Those are needed to enable when using EHC start up of this
    /* Set P404 and P405 not to be used as DCDC_EN (output low: D
    /* PORT4->PDR = 0x0000;

    /* PDR - Port Output Data
    b15-b0  PDR15 - PRD00      - Output Low Level */
    PORT4->PDR = 0x0000;

    /* PDR - Port Direction
    b15-b6  PDR15 - PRD06      - Input
    b5-b4   PDR05 - PRD04      - Output
    b3-b0   PDR03 - PRD00      - Input */
    PORT4->PDR = 0x0030;

    /* Handling of Unused ports (AVCC1 domain) */
    /* PORT0 Setting */
    /* Set P009, P008 and P007 as LEDs (output high) */

    /* PDR - Port Output Data
    b15-b10 PDR15 - PRD10      - Output Low Level
    b9-b7   PDR09 - PRD07      - Output High Level
    b6-b0   PDR06 - PRD00      - Output Low Level */
    PORT0->PDR = 0x0380;

    /* PDR - Port Direction
    b15-b10 PDR15 - PRD10      - Input PORT
    b9-b7   PDR09 - PRD07      - Output PORT
    b6-b0   PDR06 - PRD00      - Input PORT */
    PORT0->PDR = 0x0380;
}
/* End of function BoardInit */

```

図 7-2 動作開始時の端子設定例

7.1.1.2 動作開始時のクロック・電力制御モードを定義

スタートアップ処理では `r_core_cfg.h` の設定に従い、クロックおよび電力制御モードの初期設定を行います。動作環境にあわせて `r_core_cfg.h` の定義値を編集してください。`r_core_cfg.h` の設定については、6.4.1 クロック定義 を参照してください。

動作開始時のクロック・電力制御モードの設定例を図 7-3 に示します。

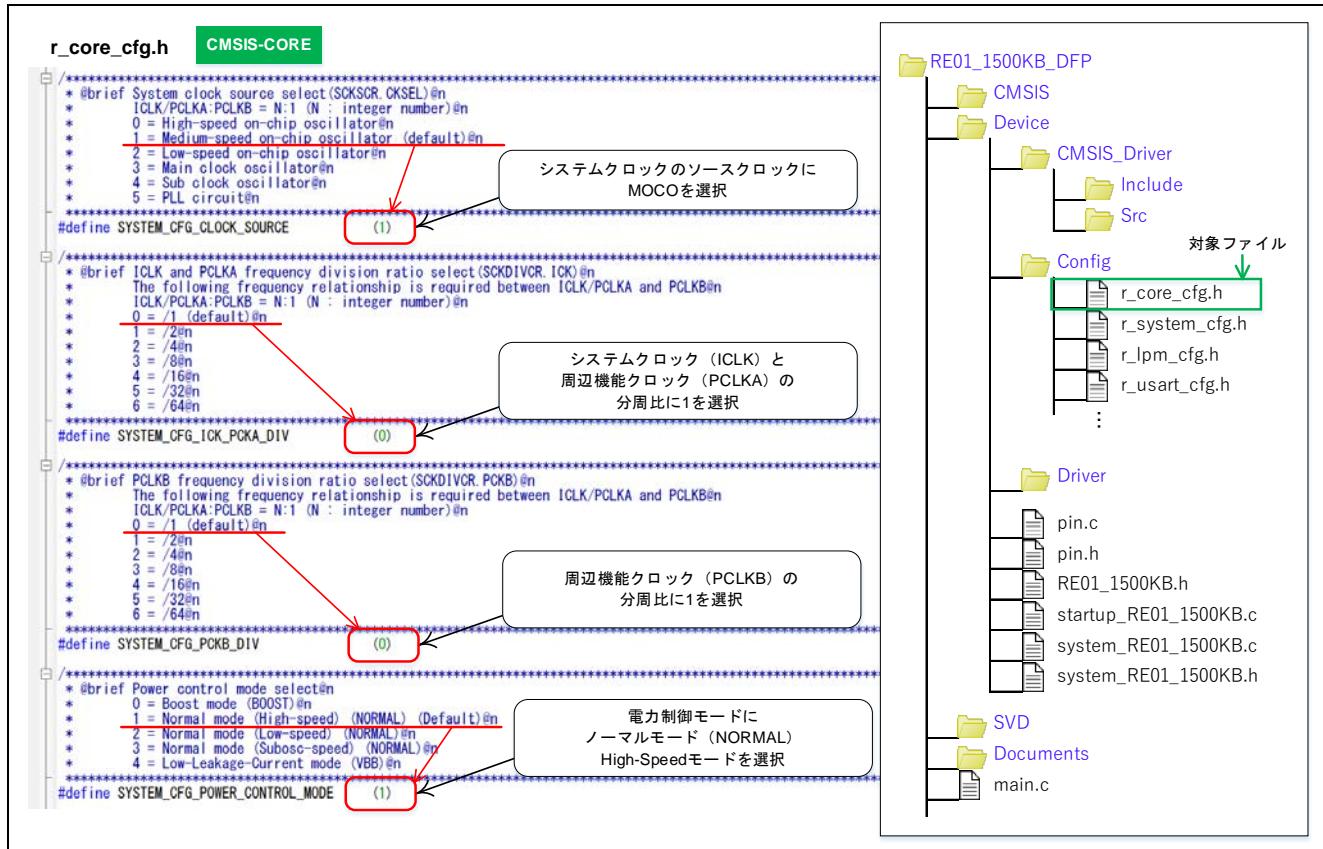


図 7-3 動作開始時のクロック・電力制御モードの設定例

7.1.2 共通機能ドライバの準備

割り込み、端子設定などの共通機能について、動作環境にあわせてドライバ関数の編集、およびコンフィグレーション定義ヘッダを編集してください。

7.1.2.1 IRQ 番号の定義

割り込みを使用する場合、周辺機能からの割り込みを IRQ 番号に割り当てる必要があります。

動作環境にあわせて、r_system_cfg.h の各割り込みに対する IRQ 番号の定義値を編集してください。IRQ 番号の設定については、6.3.2 IRQ 番号の割り当て を参照してください。

IRQ 番号の設定例を図 7-4 に示します。

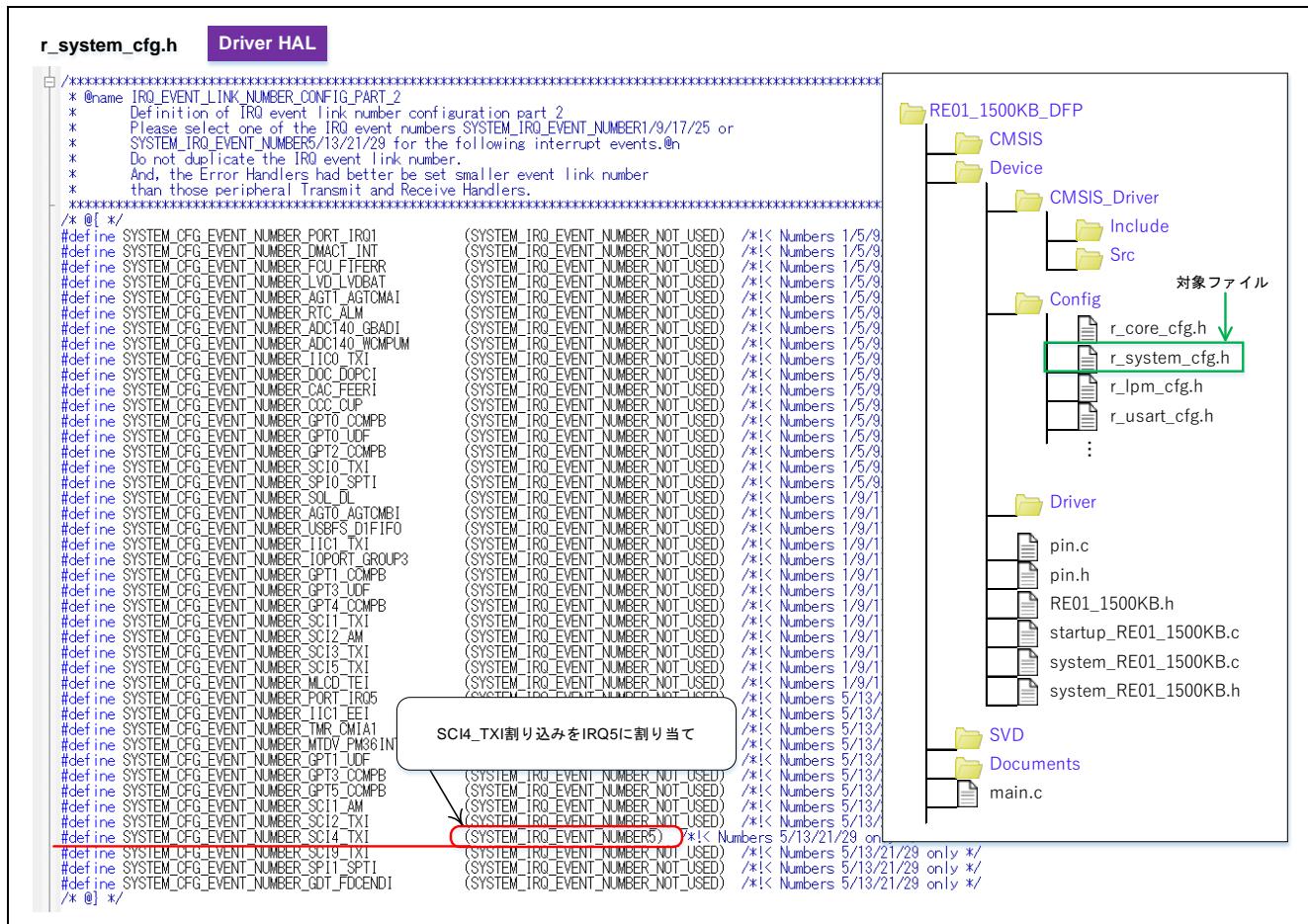


図 7-4 IRQ 番号の設定例

7.1.2.2 周辺機能で使用する端子の設定

動作環境にあわせて、pin.c の端子設定関数の処理を編集してください。端子設定関数の編集については 6.5.2 ドライバ関数の編集 を参照してください。

周辺機能で使用する端子の設定例を図 7-5 に示します。

```

pin.c Driver HAL
#include "r_core.h"
#include "r_system.h"
#include "r_lpm.h"
#include "r_usart.h"

void R_SCI_Pinset_CH4(void) // @suppress("API function naming") @suppress("Function length")
{
    /* Disable protection for PFS function (Set to PWPR register) */
    R_SYS_RegisterProtectDisable(SYSTEM_REG_PROTECT_MPC);

    /* CTS4 : P111 */
    PFS->P111PFS_b.ASEL = 0U; /* 0: Do not use as an analog pin, 1: Use as an analog pin. */
    PFS->P111PFS_b.ISEL = 0U; /* 0: Do not use as an IRQn input pin, 1: Use as an IRQn input pin. */
    PFS->P111PFS_b.PSEL = R_PIN_PRV_SCI_PSEL_04;
    PFS->P111PFS_b.PMR = 1U; /* 0: Use the pin as a general I/O port, 1: Use the pin as a peripheral module. */

    /* TXD4 : P112 */
    PFS->P112PFS_b.ASEL = 0U; /* 0: Do not use as an analog pin, 1: Use as an analog pin. */
    PFS->P112PFS_b.ISEL = 0U; /* 0: Do not use as an IRQn input pin, 1: Use as an IRQn input pin. */

    /* When using SCI in I2C mode, set the pin to NMOS Open drain. */
    PFS->P203PFS_b.NODR = 1U; /* 0: CMOS output, 1: NMOS open-drain output. */
    PFS->P203PFS_b.PODR = 0U; /* 0: CMOS output, 1: PMOS open-drain output. */
    PFS->P203PFS_b.PSEL = R_PIN_PRV_SCI_PSEL_04;
    PFS->P203PFS_b.PMR = 1U; /* 0: Use the pin as a general I/O port, 1: Use the pin as a peripheral module. */

    /* RXD4 : P112 */
    PFS->P112PFS_b.ASEL = 0U; /* 0: Do not use as an analog pin, 1: Use as an analog pin. */
    PFS->P112PFS_b.ISEL = 0U; /* 0: Do not use as an IRQn input pin, 1: Use as an IRQn input pin. */

    /* When using SCI in I2C mode, set the pin to NMOS Open drain. */
    PFS->P203PFS_b.NODR = 1U; /* 0: CMOS output, 1: NMOS open-drain output. */
    PFS->P203PFS_b.PODR = 0U; /* 0: CMOS output, 1: PMOS open-drain output. */
    PFS->P203PFS_b.PSEL = R_PIN_PRV_SCI_PSEL_04;
    PFS->P203PFS_b.PMR = 1U; /* 0: Use the pin as a general I/O port, 1: Use the pin as a peripheral module. */

    /* RXD4 : P113 */
    PFS->P113PFS_b.ASEL = 0U; /* 0: Do not use as an analog pin, 1: Use as an analog pin. */
    PFS->P113PFS_b.ISEL = 0U; /* 0: Do not use as an IRQn input pin, 1: Use as an IRQn input pin. */

    /* When using SCI in I2C mode, set the pin to NMOS Open drain. */
    PFS->P203PFS_b.NODR = 1U; /* 0: CMOS output, 1: NMOS open-drain output. */
    PFS->P203PFS_b.PODR = 0U; /* 0: CMOS output, 1: PMOS open-drain output. */
    PFS->P203PFS_b.PSEL = R_PIN_PRV_SCI_PSEL_04;
    PFS->P203PFS_b.PMR = 1U; /* 0: Use the pin as a general I/O port, 1: Use the pin as a peripheral module. */

    /* SCK4 : P108 */
    PFS->P108PFS_b.ASEL = 0U; /* 0: Do not use as an analog pin, 1: Use as an analog pin. */
    PFS->P108PFS_b.ISEL = 0U; /* 0: Do not use as an IRQn input pin, 1: Use as an IRQn input pin. */
    PFS->P108PFS_b.PSEL = R_PIN_PRV_SCI_PSEL_04;
    PFS->P108PFS_b.PMR = 1U; /* 0: Use the pin as a general I/O port, 1: Use the pin as a peripheral module. */

    /* SCK4 : P204 */
    PFS->P204PFS_b.ASEL = 0U; /* 0: Do not use as an analog pin, 1: Use as an analog pin. */
    PFS->P204PFS_b.ISEL = 0U; /* 0: Do not use as an IRQn input pin, 1: Use as an IRQn input pin. */
    PFS->P204PFS_b.PSEL = R_PIN_PRV_SCI_PSEL_04;
    PFS->P204PFS_b.PMR = 1U; /* 0: Use the pin as a general I/O port, 1: Use the pin as a peripheral module. */

    /* SCK4 : P814 */
    PFS->P814PFS_b.ASEL = 0U; /* 0: Do not use as an analog pin, 1: Use as an analog pin. */
    PFS->P814PFS_b.ISEL = 0U; /* 0: Do not use as an IRQn input pin, 1: Use as an IRQn input pin. */
    PFS->P814PFS_b.PSEL = R_PIN_PRV_SCI_PSEL_04;
    PFS->P814PFS_b.PMR = 1U; /* 0: Use the pin as a general I/O port, 1: Use the pin as a peripheral module. */

    /* Enable protection for PFS function (Set to PWPR register) */
    R_SYS_RegisterProtectEnable(SYSTEM_REG_PROTECT_MPC);
}

/* End of function R_SCI_Pinset_CH4() */

```

```

RE01_1500KB_DFP
  CMSIS
  Device
    CMSIS_Driver
      Include
      Src
  Config
    r_core_cfg.h
    r_system_cfg.h
    r_lpm_cfg.h
    r_usart_cfg.h
  :
  Driver
    pin.c
    pin.h
  RE01_1500KB.h
  startup_RE01_1500KB.c
  system_RE01_1500KB.c
  system_RE01_1500KB.h
  SVD
  Documents
  main.c

```

対象ファイル

図 7-5 周辺機能で使用する端子の設定例

7.1.2.3 共通機能ドライバ動作条件の定義

R_SYSTEM ドライバ、R_LPM ドライバには、動作条件を定義するコンフィグレーション定義ヘッダ `r_system_cfg.h`、`r_lpm_cfg.h` があります。動作環境にあわせてコンフィグレーション定義ヘッダの定義値を編集してください。

共通機能ドライバ動作条件の設定例を図 7-6 に示します。

r_system_cfg.h

Driver HAL

```

/* @brief Parameter check enable@n
 * 0 = Disable@n
 * 1 = Enable (default)@n
 */
#define SYSTEM_CFG_PARAM_CHECKING_ENABLE (1)

/* @brief Entering the critical section by masking interrupt@n
 * 0 = Disable@n
 * 1 = Enable (default)@n
 * If this setting is enabled, it masks the interrupt.
 * If not, the user need to use R_SYS_EnterCriticalSection() to enter the critical section in the user application.
 */
#define SYSTEM_CFG_ENTER_CRITICAL_SECTION_ENABLE (1)

/* @brief Register protection enable setting@n
 * 0 = Disable@n
 * 1 = Enable (default)@n
 * If this setting is enabled, register protection will be performed.
 * If not, the user need to use R_SYS_RegisterProtect() to perform protection settings in the user application.
 */
#define SYSTEM_CFG_REGISTER_PROTECTION_ENABLE (1)

```

RE01_1500KB_DFP

- CMSIS
- Device
- CMSIS_Driver
 - Include
 - Src
- Config
 - r_core_cfg.h
 - r_system_cfg.h
 - r_lpm_cfg.h
 - r_usart_cfg.h
 - ⋮
- Driver
 - pin.c
 - pin.h
- RE01_1500KB.h
- startup_RE01_1500KB.c
- system_RE01_1500KB.c
- system_RE01_1500KB.h
- SVD
- Documents
- main.c

R_SYS_Initialize関数を内蔵フラッシュメモリに配置

```

/* @name R_SYSTEM_API_LOCATION_CONFIG
 * Definition of R_System API location configuration.
 * Please select "SYSTEM_SECTION_CODE" or "SYSTEM_SECTION_RAM".
 */
/* @*/
#define SYSTEM_CFG_SECTION_R_SYS_INITIALIZE (SYSTEM_SECTION_CODE)
#define SYSTEM_CFG_SECTION_R_SYS_BOOSTSPEEDMODESET (SYSTEM_SECTION_CODE)
#define SYSTEM_CFG_SECTION_R_SYS_HIGHSPED MODESET (SYSTEM_SECTION_CODE)
#define SYSTEM_CFG_SECTION_R_SYS_LOWSPED MODESET (SYSTEM SECTION RAM FUNC)
#define SYSTEM_CFG_SECTION_R_SYS_32KHZSPED MODESET (SYSTEM SECTION RAM FUNC)
#define SYSTEM_CFG_SECTION_R_SYS_SPEED MODEGET (SYSTEM SECTION RAM FUNC)
#define SYSTEM_CFG_SECTION_R_SYS_SYSTEMCLOCKHOCSE T (SYSTEM SECTION RAM FUNC)
#define SYSTEM_CFG_SECTION_R_SYS_CLOCK (SYSTEM SECTION RAM FUNC)
#define SYSTEM_CFG_SECTION_R_SYS_SYS R_SYS_HighSpeedModeSet関数をRAMIに配置
#define SYSTEM_CFG_SECTION_R_SYS_SYS (SYSTEM SECTION RAM FUNC)
#define SYSTEM_CFG_SECTION_R_SYS_SYSTEMCLOCKPPLLSET (SYSTEM SECTION CODE)
#define SYSTEM_CFG_SECTION_R_SYS_SYSTEMCLOCKFREQGET (SYSTEM SECTION RAM FUNC)
#define SYSTEM_CFG_SECTION_R_SYS_PERIPHERALCLOCKFREQGET (SYSTEM SECTION RAM FUNC)
#define SYSTEM_CFG_SECTION_R_SYS_SYSTEMCLOCKDIVIDERSET (SYSTEM SECTION RAM FUNC)
#define SYSTEM_CFG_SECTION_R_SYS_MAINOSCSPEDCLOCKSTART (SYSTEM SECTION RAM FUNC)
#define SYSTEM_CFG_SECTION_R_SYS_MAINOSCSPEDCLOCKSTOP (SYSTEM SECTION RAM FUNC)
#define SYSTEM_CFG_SECTION_R_SYS_HIGHSPEDCLOCKSTART (SYSTEM SECTION RAM FUNC)
#define SYSTEM_CFG_SECTION_R_SYS_HIGHSPEDCLOCKSTOP (SYSTEM SECTION RAM FUNC)

```

パラメータチェックを有効に設定

クリティカルセクションを有効に設定

レジストライテプロテクト制御を有効に設定

R_SYS_Initialize() section */
R_SYS_BoostSpeedModeGet() section */
R_SYS_HighSpeedModeGet() section */
R_SYS_LowSpeedModeGet() section */
R_SYS_32KHzSpeedModeGet() section */
R_SYS_SpeedModeGet() section */
R_SYS_SystemClockHOCSE T() section */
R_SYS_SystemClockMOC() section */
R_SYS_SystemClockLOC() section */
R_SYS_SystemClockKOMS() section */
R_SYS_SystemClockKOS() section */
R_SYS_SystemClockPPLL() section */
R_SYS_SystemClockFreqGet() section */
R_SYS_PeripheralClockFreqGet() section */
R_SYS_SystemClockDividerSet() section */
R_SYS_MainOscSpeedClockStart() section */
R_SYS_MainOscSpeedClockStop() section */
R_SYS_HighSpeedClockStart() section */
R_SYS_HighSpeedClockStop() section */

図 7-6 共通機能ドライバ動作条件の設定例

7.1.2.4 R_SYSTEM ドライバで使用するクロックを定義

R_SYSTEM ドライバのクロック制御関数は、r_core_cfg.h の設定に従いクロック設定を行います。動作環境にあわせて r_core_cfg.h の定義値を編集してください。r_core_cfg.h の設定については、6.4.1 クロック定義 を参照してください。なお、r_core_cfg.h のクロック定義は、スタートアップ処理でも参照されます。

R_SYSTEM ドライバでの使用クロック設定例を図 7-7 に示します。

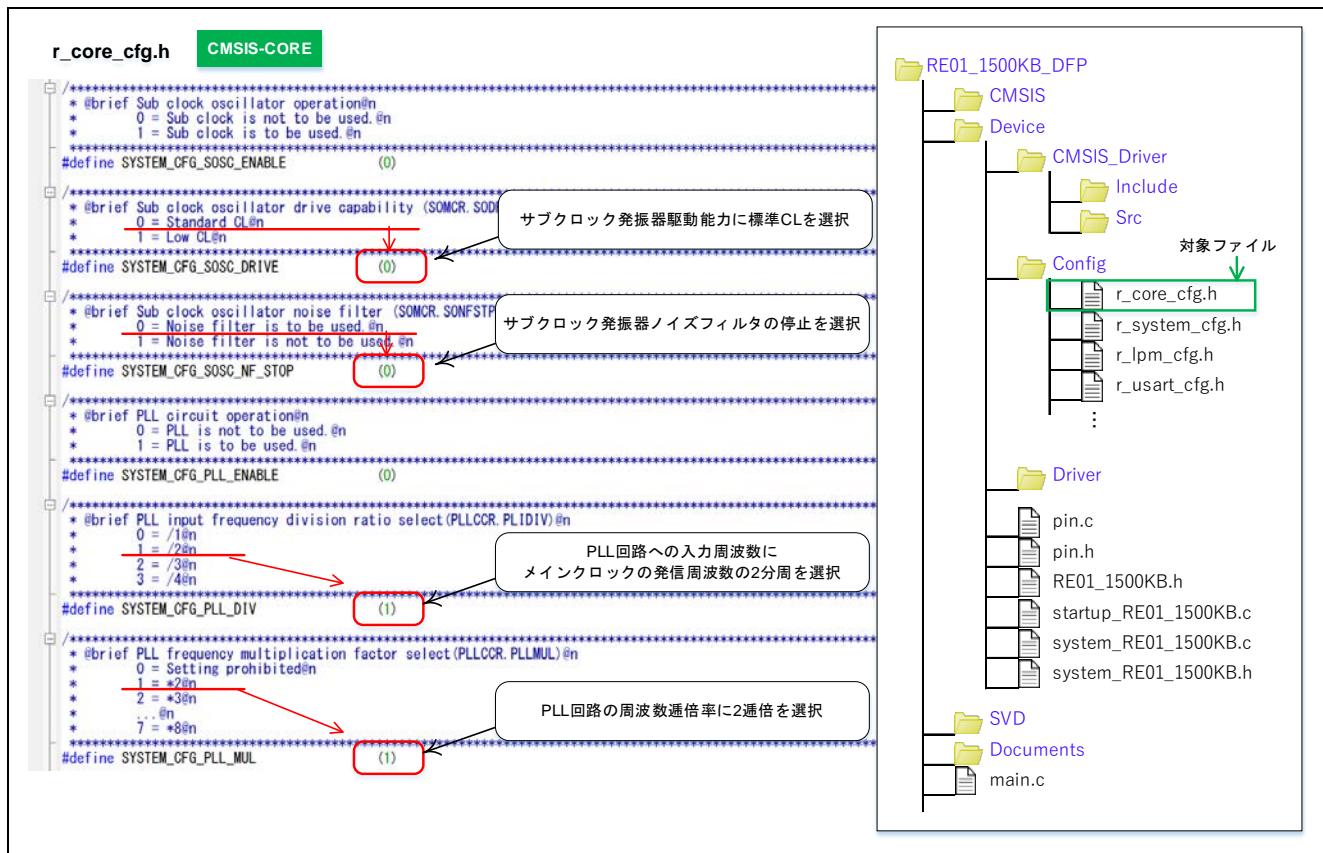


図 7-7 R_SYSTEM ドライバでの使用クロック設定例

7.1.3 周辺機能ドライバの準備

7.1.3.1 周辺機能ドライバの動作条件の定義

各ドライバには、動作条件を定義するコンフィグレーション定義ヘッダ `r_xxx_cfg.h` があります（`xxx` はドライバ機能名）。動作環境にあわせて `r_xxx_cfg.h` ファイルの定義値を編集してください。

設定例は、7.1.2.3 共通機能ドライバ動作条件の定義 を参照してください。

7.2 ユーザプログラム作成

ユーザプログラムを作成する場合、周辺機能を使用するために必要な設定があります。

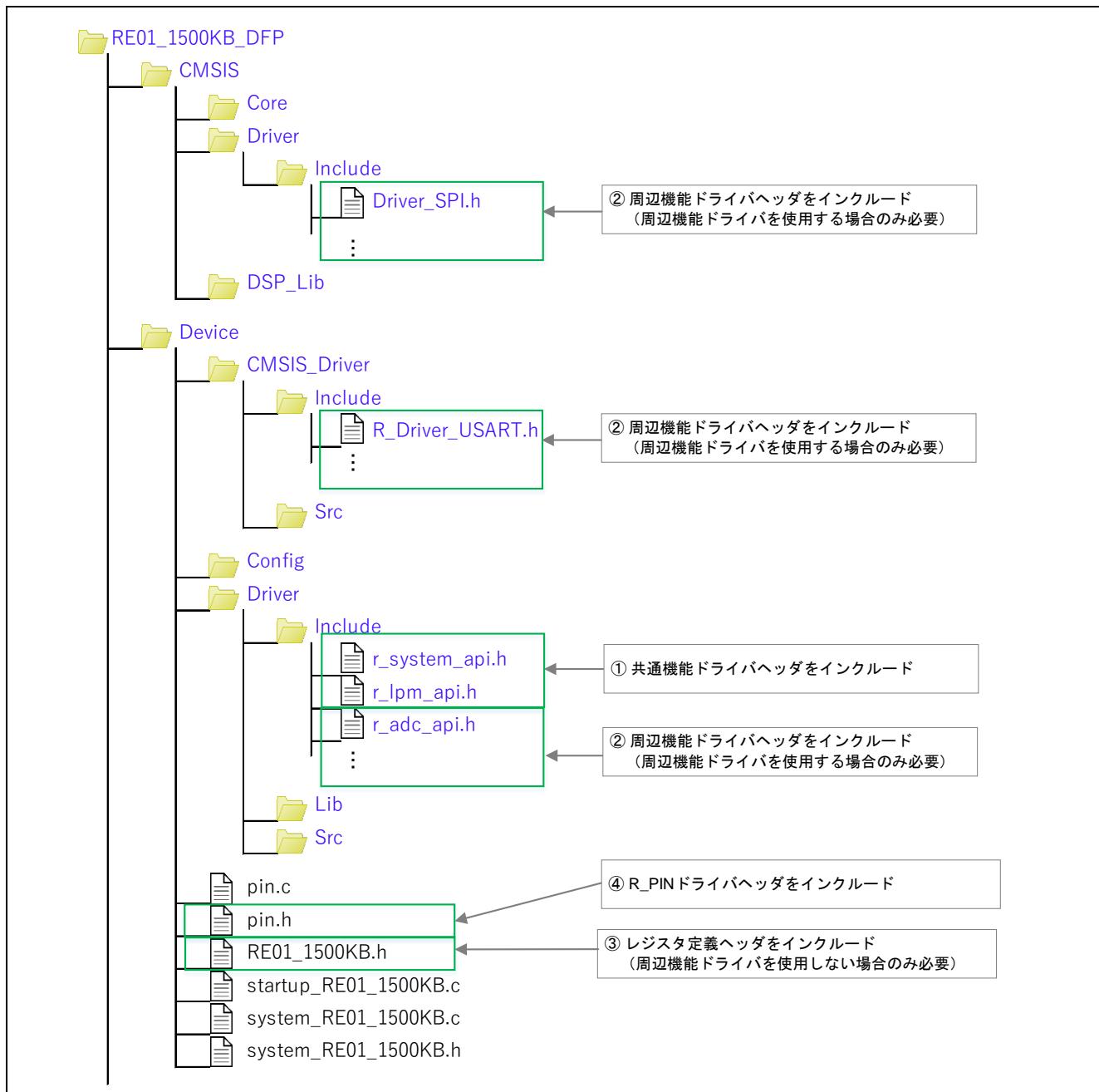
本パッケージでドライバがサポートしていない周辺機能を使用する場合、該当機能のプログラムを作成する必要があります。同一プロジェクトにおいて、ドライバを使用する機能と、ドライバを使用しない機能が混在しても問題ありません。

周辺機能を使用する場合、周辺機能ドライバを使用する場合と、周辺機能ドライバを使用しない場合で必要な処理が異なります。

表 7-2 ユーザプログラムの初期設定と周辺機能を使用する場合に必要な制御

周辺機能ドライバを使用する場合	周辺機能ドライバを使用しない場合	関連関数・ファイル
初期設定		
共通機能ドライバヘッダをインクルード	—	対象ファイルは図 7-8 の①を参照
任意のプログラムを RAM に展開	—	R_SYS_CodeCopy 関数
IO 電源ドメインに給電	—	R_LPM_IOPowerSupplyModeSet 関数
周辺機能を制御		
周辺機能ドライバヘッダをインクルード	—	対象ファイルは図 7-8 の②を参照
周辺機能ドライバのインスタンス宣言	—	—
—	レジスタ定義ヘッダをインクルード	対象ファイルは図 7-8 の③を参照
—	モジュールストップ制御	R_LPM_ModuleStart 関数 R_LPM_ModuleStop 関数
—	レジスタライトプロテクト制御	R_SYS_RegisterProtectDisable 関数 R_SYS_RegisterProtectEnable 関数
—	リソースロック制御	R_SYS_ResourceLock 関数 R_SYS_ResourceUnlock 関数
端子を制御		
—	R_PIN ドライバヘッダをインクルード	対象ファイルは図 7-8 の④を参照
—	端子制御	R_***_Pinset_*** 関数 R_***_Pinclr_*** 関数
割り込みを制御		
—	割り込み制御	R_NVIC_EnableIRQ 関数 R_NVIC_SetPriority 関数 R_NVIC_ClearPendingIRQ 関数 R_SYS_IrqEventLinkSet 関数 R_SYS_IrqStatusClear 関数 他

※：「***」は、使用するドライバおよび機能によって異なる名称です



② 周辺機能ドライバヘッダをインクルード
(周辺機能ドライバを使用する場合のみ必要)

② 周辺機能ドライバヘッダをインクルード
(周辺機能ドライバを使用する場合のみ必要)

① 共通機能ドライバヘッダをインクルード

② 周辺機能ドライバヘッダをインクルード
(周辺機能ドライバを使用する場合のみ必要)

④ R_PIN ドライバヘッダをインクルード

③ レジスタ定義ヘッダをインクルード
(周辺機能ドライバを使用しない場合のみ必要)

図 7-8 ユーザプログラム作成時のインクルード対象ファイル

7.2.1 初期設定

初期設定では、R_SYSTEM ドライバおよび R_LPM ドライバ関数を使用します。ユーザプログラムは、R_SYSTEM ドライバおよび R_LPM ドライバの共通機能ドライバヘッダをインクルードし、初期設定を行ってください。

7.2.1.1 共通機能ドライバヘッダをインクルード

R_SYSTEM ドライバを使用できるように、r_system_api.h をインクルードしてください。

R_LPM ドライバを使用できるように、r_lpm_api.h をインクルードしてください。

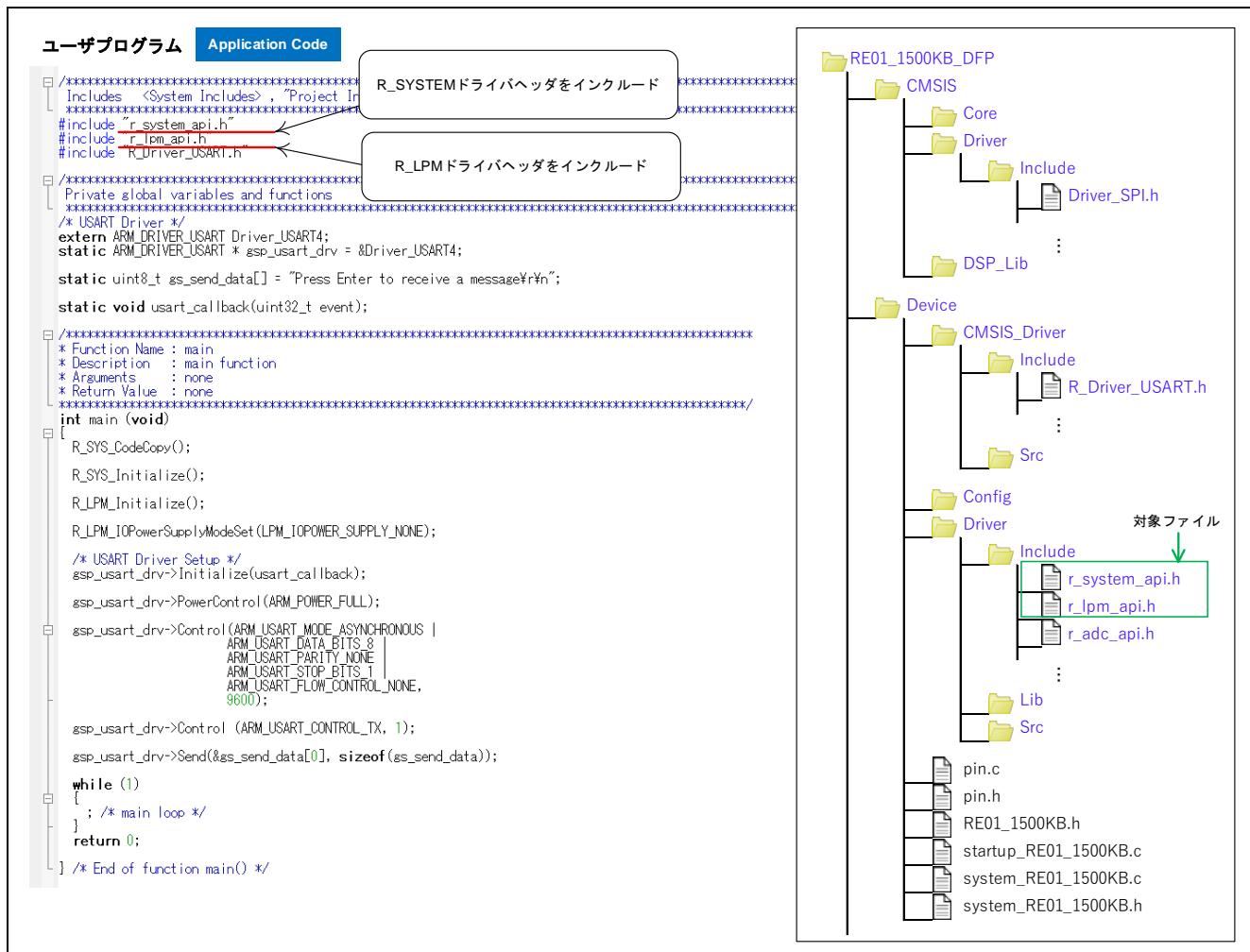


図 7-9 共通機能ドライバヘッダをインクルード

7.2.1.2 任意のプログラムを RAM に展開

本デバイスは、内蔵フラッシュメモリの電源を遮断することで消費電力を低減することができます。内蔵フラッシュメモリの電源を遮断した後に RAM に展開されたプログラムで動作させる場合は、ユーザプログラムの先頭で R_SYS_CodeCopy 関数を実行し、任意のプログラムを RAM に展開してください。

プログラムの RAM 配置は本処理だけでは実現できません。詳細は、6.6 プログラムの RAM 配置 を参照してください

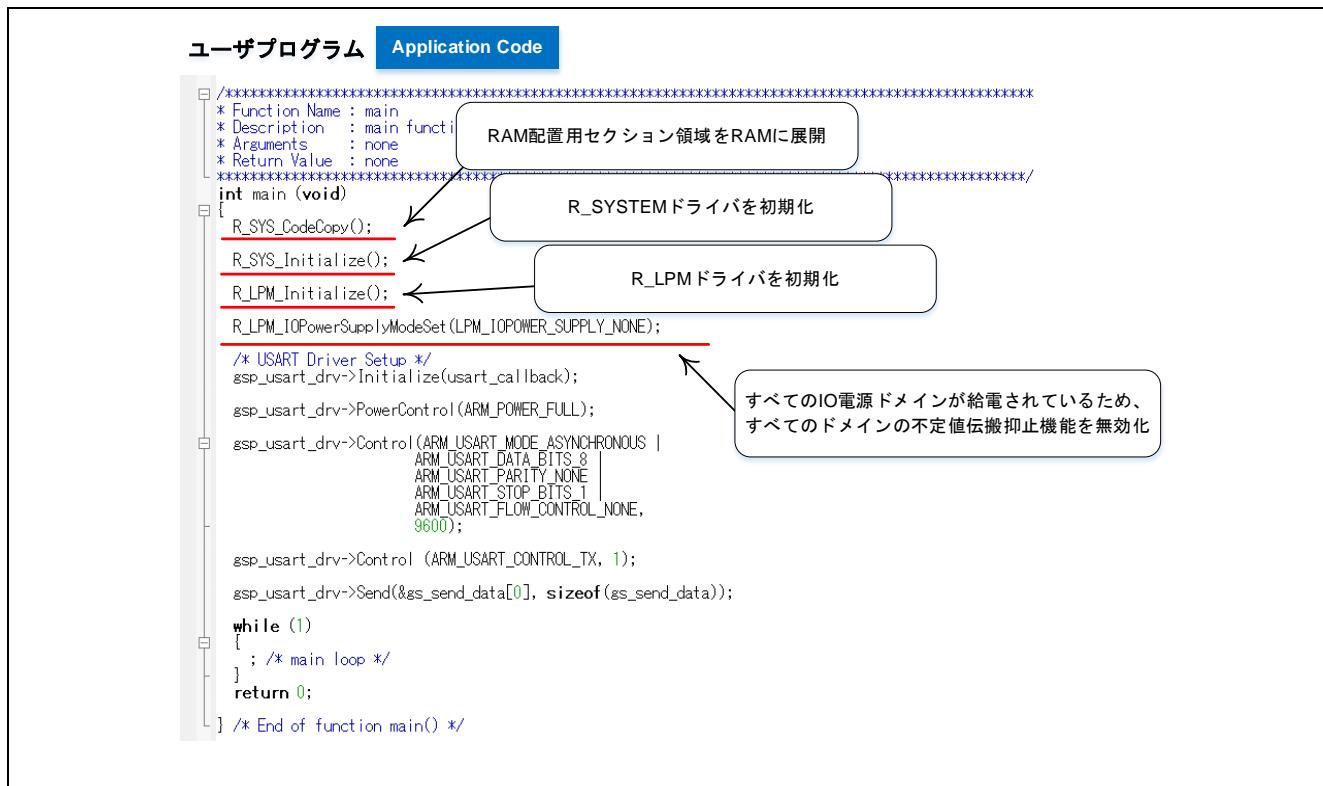


図 7-10 プログラムの RAM 展開、IO 電源ドメインに給電の設定例

7.2.1.3 IO 電源ドメインの不定値伝搬抑止機能を制御

リセット解除後、IOVCC を除くすべての IO 電源ドメインの不定値伝搬抑止機能が有効になっています。動作環境における電源ドメインへの給電状態にあわせ、R_LPM_IOPowerSupplyModeSet 関数を実行して、任意の IO 電源ドメインの不定値伝搬抑止機能を無効にしてください。

IO 電源ドメインについては、6.2 IO 電源ドメイン不定値伝搬抑止制御 を参照してください。

本設定は周辺機能ドライバでは行いません。ユーザプログラムで行ってください。

7.2.2 周辺機能を制御

7.2.2.1 周辺機能ドライバヘッダをインクルード

各ドライバには、ドライバを使用するために必要な定義を行うヘッダファイルがあります。使用する周辺機能ドライバのヘッダファイルをインクルードしてください。

CMSIS-Driver は一部ドライバでルネサス独自の拡張機能をサポートしています。以下の CMSIS-Driver を使用する場合は、拡張ヘッダをインクルードしてください。拡張ヘッダをインクルードした場合は、標準ヘッダのインクルードは不要です

表 7-3 拡張機能をサポートするドライバ

ドライバ名	インクルード対象の拡張ヘッダファイル
R_I2C	R_Driver_I2C.h
R_USART	R_Driver_USART.h

本処理は周辺機能ドライバを使用した場合のみ必要です。周辺機能ドライバを使用しない場合は必要ありません。

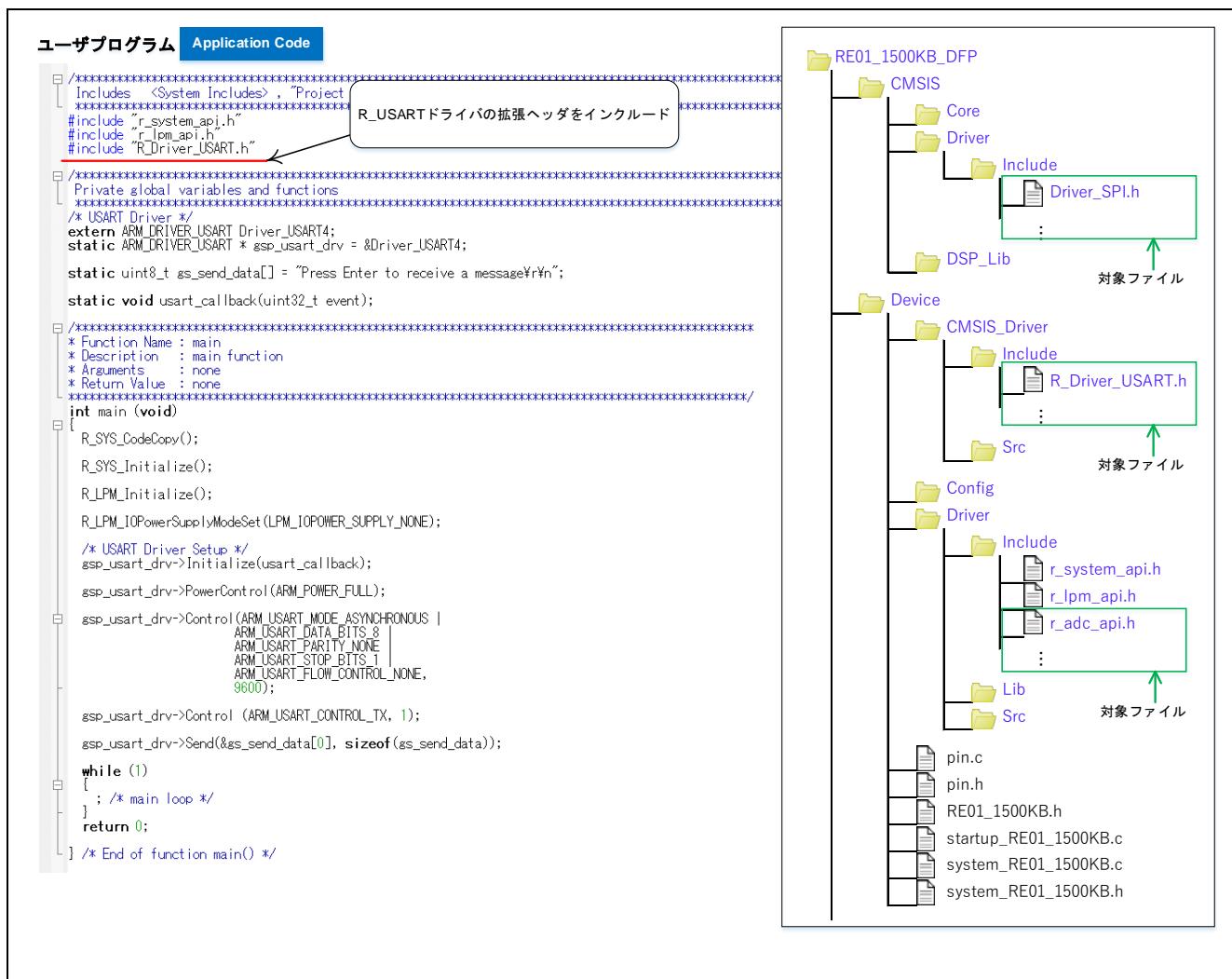


図 7-11 周辺機能ドライバヘッダのインクルード例

7.2.2.2 周辺機能ドライバのインスタンス宣言

一部ドライバでは、チャネルごとにインスタンスを用意しています。

これらのドライバを使用する場合は、インスタンス内の関数ポインタを使用して各ドライバ関数を実行することができます。各ドライバ関数の実行方法については、4章に示す各ドライバ仕様書 を参照してください。

本処理は周辺機能ドライバを使用した場合のみ必要です。周辺機能ドライバを使用しない場合は必要ありません。

```

ユーザプログラム Application Code

#include "r_system_api.h"
#include "r_lpm_api.h"
#include "R_Driver_USART.h"

Private global variables and functions
/* USART Driver */
extern ARM_DRIVER_USART Driver_USART4;
static ARM_DRIVER_USART *gsp_usart_drv = &Driver_USART4;

static uint8_t gs_send_data[] = "Press Enter to receive a message\r\n";
static void usart_callback(uint32_t event);

* Function Name : main
* Description   : main function
* Arguments     : none
* Return Value  : none

int main (void)
{
    R_SYS_CodeCopy();
    R_SYS_Initialize();
    R_LPM_Initialize();
    R_LPM_IOPowerSupplyModeSet(LPM_IOPOWER_SUPPLY_NONE);
    /* USART Driver Setup */
    gsp_usart_drv->Initialize(usart_callback);
    gsp_usart_drv->PowerControl(ARM_POWER_FULL);

    gsp_usart_drv->Control(ARM_USART_MODE_ASYNCHRONOUS |
                           ARM_USART_DATA_BITS_8 |
                           ARM_USART_PARITY_NONE |
                           ARM_USART_STOP_BITS_1 |
                           ARM_USART_FLOW_CONTROL_NONE,
                           9600);

    gsp_usart_drv->Control(ARM_USART_CONTROL_TX, 1);
    gsp_usart_drv->Send(&gs_send_data[0], sizeof(gs_send_data));

    while (1)
    {
        ; /* main loop */
    }
    return 0;
} /* End of function main() */

```

**R_USART ドライバ チャネル4のインスタンスを
本ファイルでも使用できるようextern宣言**

**チャネルを容易に変更できるよう
R_USART ドライバのインスタンスのアドレスを
ポインタ変数に設定**

**R_USART ドライバの初期化関数を実行
(ARM_USART_Initialize関数が実行される)**

図 7-12 周辺機能ドライバのインスタンス宣言使用例

7.2.2.3 レジスタ定義ヘッダをインクルード

レジスタ定義ヘッダを使用することで、ハードウェアレジスタへの書き込み/読み出しを、アドレスではなくレジスタ名で行うことができます。レジスタ定義ヘッダを使用する場合は、RE01_1500KB.h をインクルードしてください。

本処理は周辺機能ドライバを使用しない場合のみ必要です。周辺機能ドライバを使用する場合は必要ありません。

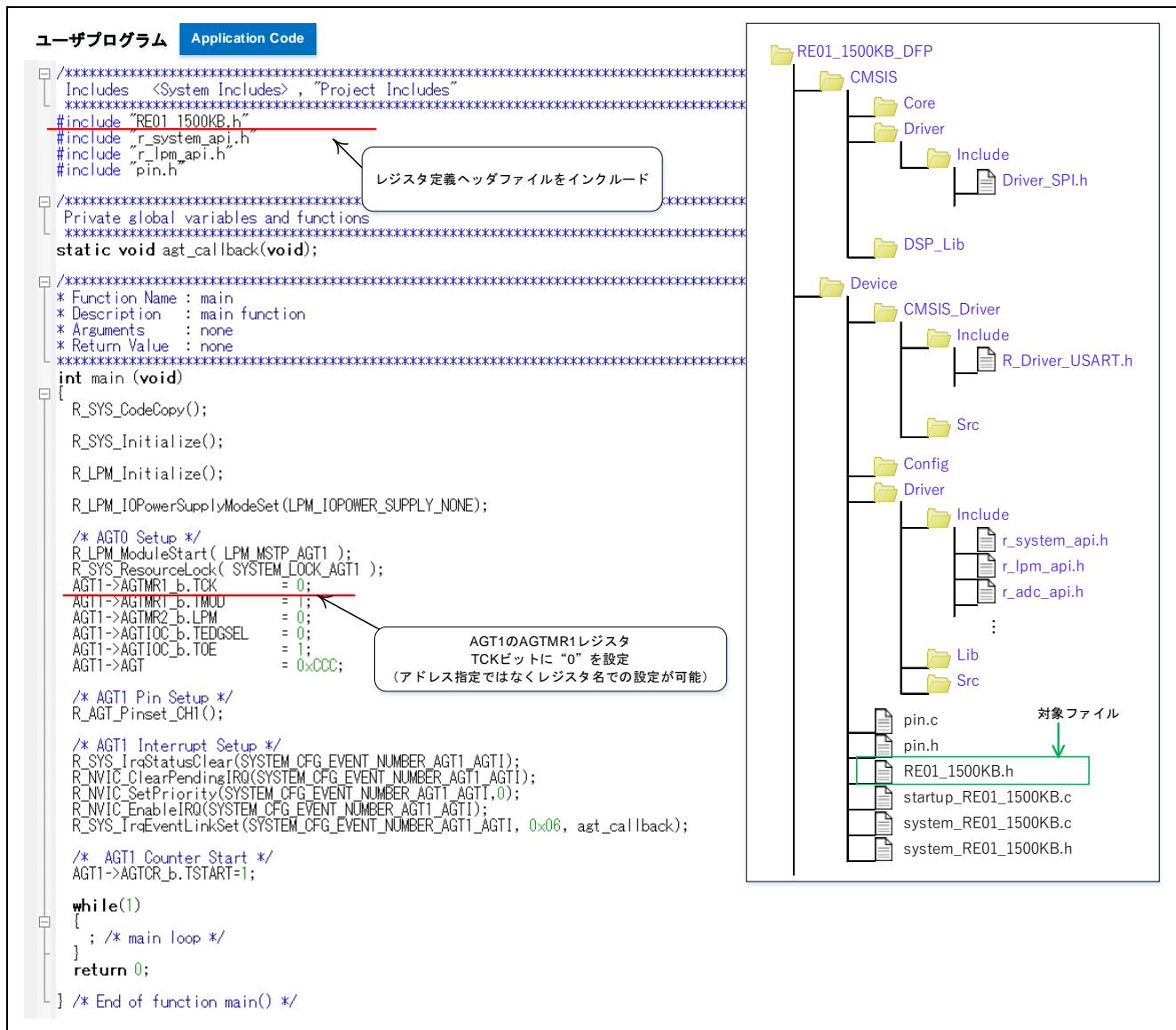


図 7-13 レジスタ定義ファイルの使用例

7.2.2.4 モジュールストップ制御

本デバイスは、一部周辺機能を除き、モジュールストップ機能を持っています。

リセット解除後は、DMA/DTC を除くすべてのモジュールストップ機能が有効になっています（周辺機能が停止）。ユーザプログラムで該当周辺機能を使用する場合は、周辺機能レジスタへアクセス前に、R_LPM_ModuleStart 関数を使用して、モジュールストップ機能を無効にしてください（周辺機能を動作）。

本処置は周辺機能ドライバを使用しない場合のみ必要です。周辺機能ドライバを使用する場合は必要ありません。

- R_LPM_ModuleStart 関数：モジュール動作
- R_LPM_ModuleStop 関数：モジュール停止 ←初期状態（DMA/DTC を除く）

```

ユーザプログラム Application Code

#include "RE01_1500KB.h"
#include "r_system_api.h"
#include "lpm_api.h"
#include "pin.h"

Private global variables and functions
static void agt_callback(void);

/* Function Name : main
 * Description  : main function
 * Arguments    : none
 * Return Value : none
 */
int main (void)
{
    R_SYS_CodeCopy();

    R_SYS_Initialize();
    R_LPM_Initialize();

    R_LPM_IOPowerSupplyModeSet (LPM_IOPOWER_SUPPLY_NONE);

    /* AGT0 Setup */
    R_LPM_ModuleStart( LPM_MSTP_AGT1 );
    R_SYS_ResourceLock( SYSTEM_LOCK_AGT1 );
    AGT1->AGTMR1_b.TOK      = 0;
    AGT1->AGTMR1_b.TMOD     = 1;
    AGT1->AGTMR2_b.LPM       = 0;
    AGT1->AGTI0C_b.TEDGSEL  = 0;
    AGT1->AGTI0C_b.TOE      = 1;
    AGT1->AGT      = 0xCCC;

    /* AGT1 Pin Setup */
    R_AGT_Pinset_CH1();

    /* AGT1 Interrupt Setup */
    R_SYS_IrqStatusClear(SYSTEM_CFG_EVENT_NUMBER_AGT1_AGT1);
    R_NVIC_ClearPendingIRQ(SYSTEM_CFG_EVENT_NUMBER_AGT1_AGT1);
    R_NVIC_SetPriority(SYSTEM_CFG_EVENT_NUMBER_AGT1_AGT1, 0);
    R_NVIC_EnableIRQ(SYSTEM_CFG_EVENT_NUMBER_AGT1_AGT1);
    R_SYS_IrqEventLinkSet(SYSTEM_CFG_EVENT_NUMBER_AGT1_AGT1, 0x06, agt_callback);

    /* AGT1 Counter Start */
    AGT1->AGTCR_b.TSTART=1;

    while(1)
    {
        ; /* main loop */
    }
    return 0;
} /* End of function main() */

```

AGT1のモジュールストップ機能を解除し
AG1を動作開始

図 7-14 モジュールストップ制御例

7.2.2.5 レジスタライトプロテクト制御

本デバイスの一部レジスタは、ライトプロテクション機能を持っています。リセット解除後はすべてのレジスタライトプロテクション機能が有効になっています（レジスタへの書き込みがプロテクトされている）ので、ユーザプログラムで該当レジスタへの書き込みを行う場合は、レジスタへ書き込みを行う前に、R_SYS_RegisterProtectDisable 関数を使用してプロテクトを解除してください。

- R_SYS_RegisterProtectDisable 関数：プロテクトを解除
- R_SYS_RegisterProtectEnable 関数：プロテクトを有効（初期状態）

本処理は周辺機能ドライバを使用しない場合のみ必要です。周辺機能ドライバを使用する場合は必要ありません。

周辺機能ドライバを使用した場合は、ドライバ内で本処理を行っています。ドライバのコンフィグレーション定義ヘッダでレジスタライトプロテクト制御を無効にした場合は、ドライバ内で本処理は行われませんので、ユーザプログラムで処理してください。

ユーザプログラム
Application Code

```

R_SYS_RegisterProtectDisable(SYSTEM_REG_PROTECT_0M_LPC_BATT);

/* Set the software standby mode */
SYSC->SBYCR = (uint16_t)SYSC_SBYCR_SSBY_Msk;
SYSC->DPSBYCR = R_LPM_PRV_REG_DPSBYCR_DSTBYOFF;

/* Disable the snooze mode */
SYSC->SNZCR = R_LPM_PRV_REG_SNZCR_SNZOFF;

/* Set the all power supply mode */
SYSC->PWSTCR = R_LPM_PRV_REG_PWSTCR_ALLPWON;

R_SYS_RegisterProtectEnable(SYSTEM_REG_PROTECT_0M_LPC_BATT);

```

レジスタのライトプロテクトを解除
引数のSYSTEM_REG_PROTECT_0M_LPC_BATTは、
対象が低消費電力モジュール／バッテリ関連レジスタであることを示す

レジスタのライトプロテクトを設定

図 7-15 レジスタライトプロテクト制御例

7.2.2.6 リソースロック制御

R_SYSTEM ドライバは、ハードウェア機能の競合を検出するためのリソースロック関数を用意しています。

周辺機能を使用する場合は、R_SYS_ResourceLock 関数を使用してリソースをロックしてください。

- R_SYS_ResourceLock 関数 : リソースをロック
- R_SYS_ResourceUnlock 関数 : リソースを開放（初期状態）

本処理は周辺機能ドライバを使用しない場合のみ必要です。周辺機能ドライバを使用する場合は必要ありません。周辺機能ドライバを使用した場合は、ドライバ内で本処理を行っています。

```

ユーザプログラム Application Code

/*
 * Includes <System Includes>, "Project Includes"
 */
#include "RE01_1500KB.h"
#include "r_system_api.h"
#include "r_lpm_api.h"
#include "pin.h"

/*
 * Private global variables and functions
 */
static void agt_callback(void);

/*
 * Function Name : main
 * Description   : main function
 * Arguments     : none
 * Return Value  : none
 */
int main (void)
{
    R_SYS_CodeCopy();
    R_SYS_Initialize();
    R_LPM_Initialize();
    R_LPM_IOPowerSupplyModeSet(LPM_IOPOWER_SUPPLY_NONE);

    /* AGT0 Setup */
    R_LPM_ModuleStart( LPM_MSTP_AGT1 );
    R_SYS_ResourceLock( SYSTEM_LOCK_AGT1 );
    AGT1->AGTMR1_b.TCK      = 0;
    AGT1->AGTMR1_b.TMOD     = 1;
    AGT1->AGTMR2_b.LPM      = 0;
    AGT1->AGT1OC_b.TEDGSEL  = 0;
    AGT1->AGT1OC_b.TOE      = 1;
    AGT1->AGT                = 0xCCC;

    /* AGT1 Pin Setup */
    R_AGT_PInset_CHT();

    /* AGT1 Interrupt Setup */
    R_SYS_IrqStatusClear(SYSTEM_CFG_EVENT_NUMBER_AGT1_AGT1);
    R_NVIC_ClearPendingIRQ(SYSTEM_CFG_EVENT_NUMBER_AGT1_AGT1);
    R_NVIC_SetPriority(SYSTEM_CFG_EVENT_NUMBER_AGT1_AGT1, 0);
    R_NVIC_EnableIRQ(SYSTEM_CFG_EVENT_NUMBER_AGT1_AGT1);
    R_SYS_IrqEventLinkSet(SYSTEM_CFG_EVENT_NUMBER_AGT1_AGT1, 0x06, agt_callback);

    /* AGT1 Counter Start */
    AGT1->AGTCR_b.TSTART=1;

    while(1)
    {
        ; /* main loop */
    }
    return 0;
} /* End of function main() */

```

図 7-16 ハードウェアのリソースロック制御例

7.2.3 端子を制御

R_PIN ドライバは、周辺機能で使用する端子設定関数を用意しています。ユーザプログラムは、R_PIN ドライバの端子設定関数を使用して端子を設定することができます。

7.2.3.1 R_PIN ドライバヘッダをインクルード

R_PIN ドライバには、ドライバを使用するために必要な定義を行っているヘッダファイルがあります。R_PIN ドライバを使用して端子設定を行う場合は、pin.h をインクルードしてください。

7.2.3.2 端子制御

R_PIN ドライバ関数を使用して端子設定を行うことができます。

周辺機能 ドライバを使用した場合は、ドライバ内で本制御を行っています。

```

ユーザプログラム Application Code

#include "RE01_1500KB.h"
#include "r_system_api.h"
#include "r_lpm_api.h"
#include "pin.h"

Private global variables and functions
***** R_PIN ドライバヘッダファイルをインクルード *****

static void agt_callback(void);

* Function Name : main
* Description   : main function
* Arguments     : none
* Return Value  : none
*****
int main (void)
{
    R_SYS_CodeCopy();
    R_SYS_Initialize();
    R_LPM_Initialize();
    R_LPM_IOPowerSupplyModeSet (LPM_IOPOWER_SUPPLY_NONE);

    /* AGT0 Setup */
    R_LPM_ModuleStart( LPM_MSTP_AGT0 );
    R_SYS_ResourceLock( SYSTEM_LOCK_AGT0 );
    AGT0->AGTMR1_b.TCK      = 0;
    AGT0->AGTMR1_b.TMOD     = 1;
    AGT0->AGTMR2_b.LPM      = 0;
    AGT0->AGTIOC_b.TEDGSEL  = 0;
    AGT0->AGTIOC_b.TOE      = 1;
    AGT0->AGT                = 0xCCC;

    /* AGT1 Pin Setup */
    R_AGT_Pinset_CHT();

    /* AGT1 Interrupt Setup */
    R_SYS_IrqStatusClear(SYSTEM_CFG_EVENT_NUMBER_AGT1_AGT1);
    R_NVIC_ClearPendingIRQ(SYSTEM_CFG_EVENT_NUMBER_AGT1_AGT1);
    R_NVIC_SetPriority(SYSTEM_CFG_EVENT_NUMBER_AGT1_AGT1, 0);
    R_NVIC_EnableIRQ(SYSTEM_CFG_EVENT_NUMBER_AGT1_AGT1);
    R_SYS_IrqEventLinkSet(SYSTEM_CFG_EVENT_NUMBER_AGT1_AGT1, 0x06, agt_callback);

    /* AGT1 Counter Start */
    AGT1->AGTCR_b.TSTART=1;

    while(1)
    {
        ; /* main loop */
    }
    return 0;
} /* End of function main() */

```

RE01_1500KB_DFP
CMSIS
Core
Driver
Include
Driver_SPI.h
DSP_Lib
Device
CMSIS_Driver
Include
R_Driver_USART.h
Src
Config
Driver
Include
r_system_api.h
r_lpm_api.h
r_adc_api.h
Lib
Src
pin.c
pin.h
RE01_1500KB.h
startup_RE01_1500KB.c
system_RE01_1500KB.c
system_RE01_1500KB.h

図 7-17 端子制御例

7.2.4 割り込みを制御

割り込みを使用する場合、周辺機能からの割り込みを IRQ 番号に割り当てる必要があります。IRQ 番号を定義する r_system_cfg.h は、r_system_api.h でインクルードされています。

7.2.4.1 割り込み制御

割り込みを使用する場合、周辺機能、ICU、NVIC それぞれに対して設定を行う必要があります。

本項では、ICU および NVIC への割り込み設定例を示します。周辺機能の割り込み設定は、各周辺機能のレジスタで設定してください。割り込み設定の詳細は、6.3 割り込み制御 を参照してください

周辺ドライバを使用した場合は、ドライバ内で本制御を行っています。

```

ユーザプログラム Application Code
└─ Includes <System Includes>, "Project Includes"
    └─ *****
        └─ #include "RE01_1500KB.h"
        └─ #include "r_system_api.h"
        └─ #include "r_lpm_api.h"
        └─ #include "pin.h"
    └─ *****
        └─ Private global variables and functions
            └─ *****
                └─ static void agt_callback(void);
    └─ *****
        └─ * Function Name : main
        └─ * Description  : main function
        └─ * Arguments   : none
        └─ * Return Value: none
            └─ *****
        └─ int main (void)
            └─ [
                └─ R_SYS_CodeCopy();
                └─ R_SYS_Initialize();
                └─ R_LPM_Initialize();
                └─ R_LPM_IOPowerSupplyModeSet (LPM_IOPOWER_SUPPLY_NONE);
                └─ /* AGT0 Setup */
                    └─ R_LPM_ModuleStart( LPM_MSTP_AGT1 );
                    └─ R_SYS_ResourceLock( SYSTEM_LOCK_AGT1 );
                    └─ AGT1->AGTMR1_b.TCK = 0;
                    └─ AGT1->AGTMR1_b.TMOD = 1;
                    └─ AGT1->AGTMR2_b.LPM = 0;
                    └─ AGT1->AGTI0C_b.TEDSEL = 0;
                    └─ AGT1->AGTI0C_b.TOE = 1;
                    └─ AGT1->AGT = 0xCCC;
                └─ /* AGT1 Pin Setup */
                    └─ R_AGT_Pinset_CHT();
                └─ /* AGT1 Interrupt Setup */
                    └─ R_SYS_IrqStatusClear(SYSTEM_CFG_EVENT_NUMBER_AGT1_AGT1);
                    └─ R_NVIC_ClearPendingIRQ(SYSTEM_CFG_EVENT_NUMBER_AGT1_AGT1);
                    └─ R_NVIC_SetPriority(SYSTEM_CFG_EVENT_NUMBER_AGT1_AGT1, 0);
                    └─ R_NVIC_EnableIRQ(SYSTEM_CFG_EVENT_NUMBER_AGT1_AGT1);
                    └─ R_SYS_IrqEventLinkSet(SYSTEM_CFG_EVENT_NUMBER_AGT1_AGT1, 0x06, agt_callback);
                └─ /* AGT1 Counter Start */
                    └─ AGT1->AGTCR_b.TSTART=1;
                └─ while(1)
                    └─ [ ; /* main loop */ ]
                └─ return 0;
            └─ ] /* End of function main() */
    └─ *****

```

r_system_cfg.h
SYSTEM_CFG_EVENT_NUMBER_AGT1_AGT1 = IRQ4
を設定している場合の例を示します

/* AGT1 Interrupt Setup */
R_SYS_IrqStatusClear(SYSTEM_CFG_EVENT_NUMBER_AGT1_AGT1);
R_NVIC_ClearPendingIRQ(SYSTEM_CFG_EVENT_NUMBER_AGT1_AGT1);
R_NVIC_SetPriority(SYSTEM_CFG_EVENT_NUMBER_AGT1_AGT1, 0);
R_NVIC_EnableIRQ(SYSTEM_CFG_EVENT_NUMBER_AGT1_AGT1);
R_SYS_IrqEventLinkSet(SYSTEM_CFG_EVENT_NUMBER_AGT1_AGT1, 0x06, agt_callback);

IRQ4の割り込みステータスをクリア

IRQ4に保留されている割り込みをクリア

IRQ4割り込みの優先度を設定

IRQ4割り込みを許可

AGT1のAGT1_AGT1割り込みをIRQ4に割り当てる
agt_callback関数をコールバック関数に登録
引数の0x06は、IRQ4の割り込みに対する
AGT1_AGT1のイベント番号を示します

図 7-18 割り込み制御例

7.3 ユーザプログラム作成例

7.3.1 周辺機能ドライバを使用する場合の例（UART 通信）

main.c ファイル

```
#include "r_system_api.h"
#include "r_lpm_api.h"
#include "R_Driver_USART.h"

/* USART Driver */
extern ARM_DRIVER_USART Driver_USART4;
static ARM_DRIVER_USART * gsp_usart_drv = &Driver_USART4;

static uint8_t gs_send_data[] = "Press Enter to receive a message\r\n";
static void usart_callback(uint32_t event);

/* main function */
int main (void)
{
    R_SYS_CodeCopy();
    R_SYS_Initialize();
    R_LPM_Initialize();
    R_LPM_IOPowerSupplyModeSet(LPM_IOPOWER_SUPPLY_NONE);

    /* USART Driver Setup */
    gsp_usart_drv->Initialize(usart_callback);
    gsp_usart_drv->PowerControl(ARM_POWER_FULL);
    gsp_usart_drv->Control(ARM_USART_MODE_ASYNCHRONOUS |
        ARM_USART_DATA_BITS_8 |
        ARM_USART_PARITY_NONE |
        ARM_USART_STOP_BITS_1 |
        ARM_USART_FLOW_CONTROL_NONE,
        9600);
    gsp_usart_drv->Control(ARM_USART_CONTROL_TX, 1);
    gsp_usart_drv->Send(&gs_send_data[0], sizeof(gs_send_data));

    while (1); /* main loop */
    return 0;
} /* End of function main() */

/* callback function */
static void usart_callback (uint32_t event)
{
    switch (event)
    {
        case ARM_USART_EVENT_SEND_COMPLETE:
            ; /* Success */
            break;
        default:
            ; /* */
            break;
    }
    return ;
} /* End of function usart_callback() */
```

pin.c ファイル

```
/************************************************************************** */
* @brief This function sets Pin of SCI4.
*****
/* Function Name : R_SCI_Pinset_CH4 */
void R_SCI_Pinset_CH4(void)
{
    R_SYS_RegisterProtectDisable(SYSTEM_REG_PROTECT_MPC);

    /* TxD4 : P812 */
    PFS->P812PFS_b.ASEL = 0U;
    PFS->P812PFS_b.ISEL = 0U;
    PFS->P812PFS_b.PSEL = R_PIN_PRV_SCI_PSEL_04;
    PFS->P812PFS_b.PMR = 1U;

    /* RXD4 : P813 */
    PFS->P813PFS_b.ASEL = 0U;
    PFS->P813PFS_b.ISEL = 0U;
    PFS->P813PFS_b.PSEL = R_PIN_PRV_SCI_PSEL_04;
    PFS->P813PFS_b.PMR = 1U;

    R_SYS_RegisterProtectEnable(SYSTEM_REG_PROTECT_MPC);

}/* End of function R_SCI_Pinset_CH4() */

```

r_system_api_cfg.h ファイル

```
/************************************************************************** */
* @name IRQ_EVENT_LINK_NUMBER_CONFIG_PART_2
*      Definition of IRQ event link number configuration part 2
*      Please select one of the IRQ event numbers SYSTEM_IRQ_EVENT_NUMBER1/9/17/25 or
*      SYSTEM_IRQ_EVENT_NUMBER5/13/21/29 for the following interrupt events.@n
*      Do not duplicate the IRQ event link number.
*      And, the Error Handlers had better be set smaller event link number
*      than those peripheral Transmit and Receive Handlers.
*****
/* @{ */

// 省略
#define SYSTEM_EVENT_NUMBER_GPT1_UDF (SYSTEM_EVENT_NUMBER_NOT_USED)
#define SYSTEM_EVENT_NUMBER_GPT3_CCMPP (SYSTEM_EVENT_NUMBER_NOT_USED)
#define SYSTEM_EVENT_NUMBER_GPT5_CCMPP (SYSTEM_EVENT_NUMBER_NOT_USED)
#define SYSTEM_EVENT_NUMBER_SCI1_AM (SYSTEM_EVENT_NUMBER_NOT_USED)
#define SYSTEM_EVENT_NUMBER_SCI2_TXI (SYSTEM_EVENT_NUMBER_NOT_USED)
#define SYSTEM_EVENT_NUMBER_SCI4_TXI (SYSTEM_EVENT_NUMBER5)
#define SYSTEM_EVENT_NUMBER_SCI9_TXI (SYSTEM_EVENT_NUMBER_NOT_USED)
#define SYSTEM_EVENT_NUMBER_SPI1_SPTI (SYSTEM_EVENT_NUMBER_NOT_USED)
#define SYSTEM_EVENT_NUMBER_GDT_FDCENDI (SYSTEM_EVENT_NUMBER_NOT_USED)
/* @} */

```

7.3.2 周辺機能ドライバを使用しない場合の例（パルス出力）

main.c ファイル

```

#include "RE01_1500KB.h"
#include "r_system_api.h"
#include "r_lpm_api.h"
#include "pin.h"

static void agt_callback(void);

/* main function */
int main (void)
{
    R_SYS_CodeCopy();
    R_SYS_Initialize();
    R_LPM_Initialize();
    R_LPM_IOPowerSupplyModeSet(LPM_IOPOWER_SUPPLY_NONE);

    /* AGT1 Setup */
    R_LPM_ModuleStart( LPM_MSTP_AGT1 );
    R_SYS_ResourceLock( SYSTEM_LOCK_AGT1 );
    AGT1->AGTMR1_b.TCK      = 0;
    AGT1->AGTMR1_b.TMOD     = 1;
    AGT1->AGTMR2_b.LPM      = 0;
    AGT1->AGTIOC_b.TEDGSEL  = 0;
    AGT1->AGTIOC_b.TOE      = 1;
    AGT1->AGT          = 0xCCC;

    /* AGT1 Pin Setup */
    R_AGT_Pinset_CH1();

    /* AGT1 Interrupt Setup */
    R_SYS_IrqStatusClear(SYSTEM_CFG_EVENT_NUMBER_AGT1_AGTI);
    R_NVIC_ClearPendingIRQ(SYSTEM_CFG_EVENT_NUMBER_AGT1_AGTI);
    R_NVIC_SetPriority(SYSTEM_CFG_EVENT_NUMBER_AGT1_AGTI, 0);
    R_NVIC_EnableIRQ(SYSTEM_CFG_EVENT_NUMBER_AGT1_AGTI);
    R_SYS_IrqEventLinkSet(SYSTEM_CFG_EVENT_NUMBER_AGT1_AGTI, 0x06, agt_callback);

    AGT1->AGTCR_b.TSTART=1; // AGT1 Counter Start

    while (1); /* main loop */
    return 0;
} /* End of function main() */

/* callback function */
static void agt_callback (void)
{
    /* Clear underflow flag */
    AGT1->AGTCR_b.TUNDF = 0;

    return ;
} /* End of function agt_callback() */

```

レジスタ定義ヘッダをインクルード
共通機能ドライバのヘッダをインクルード
R_PIN ドライバヘッダをインクルード

初期設定

AGT1 モジュールストップ機能解除
AGT1 ハードウェアリソースロック

レジスタ定義ヘッダを使用してレジスタに書き込み

AGT1 で使用する端子の設定

AGT1_AGTI 割り込みの設定

コールバック関数

pin.c ファイル

```
/************************************************************************** */
* @brief This function sets Pin of AGT1.
*****
/* Function Name : R_AGT_Pinset_CH1 */
void R_AGT_Pinset_CH1(void)
{
    R_SYS_RegisterProtectDisable(SYSTEM_REG_PROTECT_MPC);

    /* AGTIO1 : P309 */
    PFS->P309PFS_b.ASEL = 0U;
    PFS->P309PFS_b.ISEL = 0U;
    PFS->P309PFS_b.PSEL = R_PIN_PRV_AGT_PSEL;
    PFS->P309PFS_b.PMR = 1U;

    /* AGTO1 : P308 */
    PFS->P308PFS_b.ASEL = 0U;
    PFS->P308PFS_b.ISEL = 0U;
    PFS->P308PFS_b.PSEL = R_PIN_PRV_AGT_PSEL;
    PFS->P308PFS_b.PMR = 1U;

    R_SYS_RegisterProtectEnable(SYSTEM_REG_PROTECT_MPC);

}/* End of function R_AGT_Pinset_CH1 () */

```

P309 を AGTIO1 に設定

P308 を AGTO1 に設定

r_system_api_cfg.h ファイル

```
/************************************************************************** */
* @name IRQ_EVENT_LINK_NUMBER_CONFIG_PART_1
*      Definition of IRQ event link number configuration part 1
*      Please select one of the IRQ event numbers SYSTEM_IRQ_EVENT_NUMBER0/8/16/24 or
*      SYSTEM_IRQ_EVENT_NUMBER4/12/20/28 for the following interrupt events.@n
*      Do not duplicate the IRQ event link number.
*      And, the Error Handlers had better be set smaller event link number
*      than those peripheral Transmit and Receive Handlers.
*****
/* @{ */
#define SYSTEM_EVENT_NUMBER_PORT_IRQ0      (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED)
#define SYSTEM_EVENT_NUMBER_DMAC0_INT      (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED)
#define SYSTEM_EVENT_NUMBER_DTC_COMPLETE   (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED)
#define SYSTEM_EVENT_NUMBER_ICU_SNZCANCEL (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED)
#define SYSTEM_EVENT_NUMBER_LVD_LVD1       (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED)
#define SYSTEM_EVENT_NUMBER_AGT1_AGTI     (SYSTEM_IRQ_EVENT_NUMBER0)
#define SYSTEM_EVENT_NUMBER_WDT_NMIUNDF   (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED)
#define SYSTEM_EVENT_NUMBER_ADC140ADI     (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED)
#define SYSTEM_EVENT_NUMBER_ADC140WCMPM   (SYSTEM_IRQ_EVENT_NUMBER_NOT_USED)

// 省略
/* @} */

```

AGT1_AGTI 割り込みを IRQ0 に割り当てる

8. プロジェクトを作成してみよう

本章は RE01 1500KB グループを例に説明します。以降、特に記述がない限り、その手順は RE01 256KB グループも同じです。

本パッケージに同梱されているプロジェクトを立ち上げた後、動作環境にあわせて各設定を変更する方法を説明します。

8.1 EWARM 編

本パッケージに同梱されている実行ファイルをダブルクリックし、プロジェクトを EWARM で起動してください。

実行ファイル : project.eww

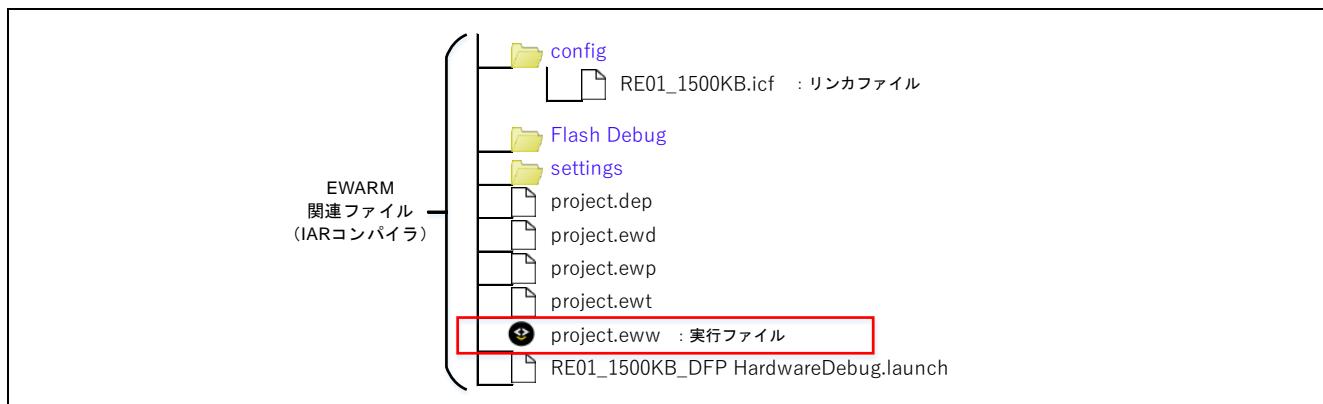


図 8-1 プロジェクト実行ファイル (EWARM)

プロジェクトの各設定は、プロジェクトのオプション画面で行います。図 8-2 を参考にオプション画面を表示してください。

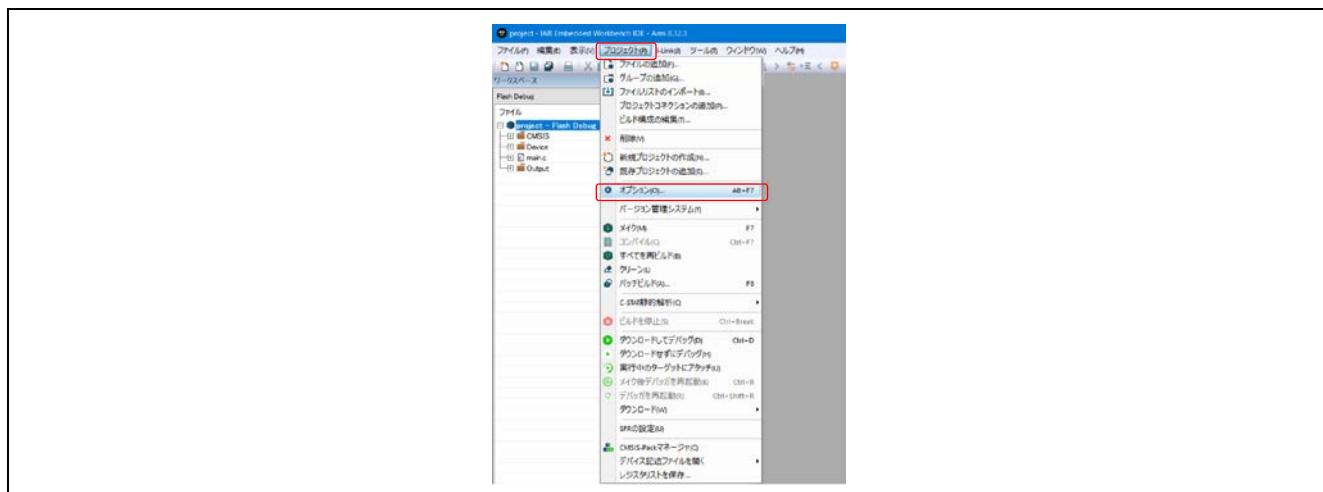


図 8-2 オプション画面表示方法 (EWARM)

8.1.1 ターゲットプロセッサの設定

RE01 1500KB グループ CMSIS パッケージは、ターゲットプロセッサに「RE01 1500KB R7F0E015D2CFB」を、RE01 256KB グループ CMSIS パッケージは、「RE01 256KB R7F0E01182CFP」を設定しています。ユーザが設定する場合は、図 8-3 を参考に使用するデバイスを選択してください。

デバイス : Renesas R7F0E015D2CFB

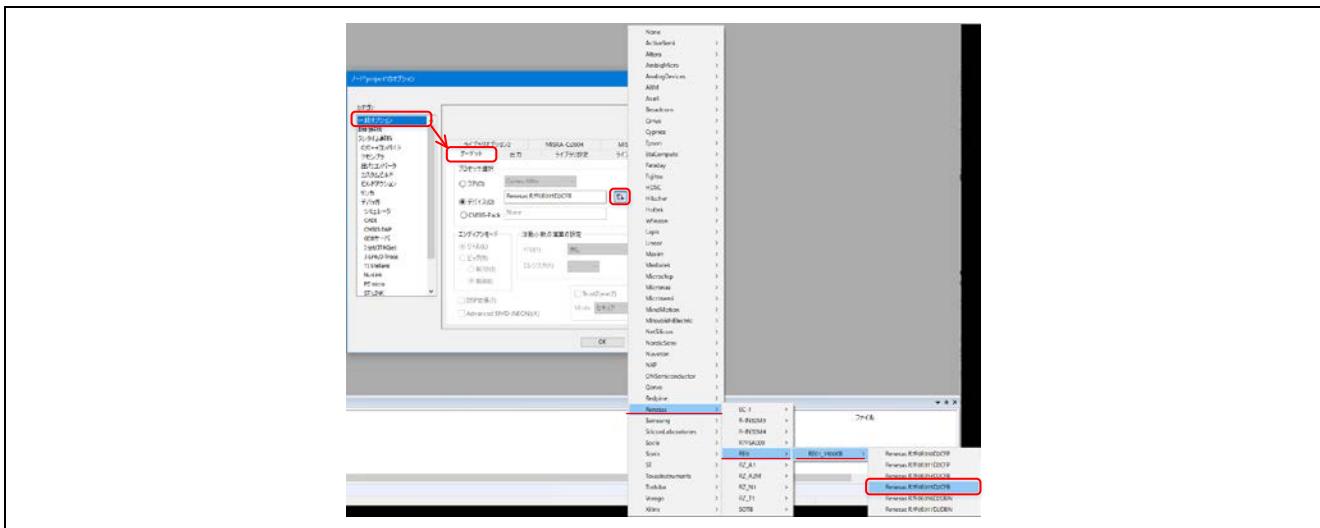


図 8-3 ターゲットプロセッサの設定例

8.1.2 リンカファイルの設定

本パッケージでは、RAM 配置用セクション定義および MTB 用領域を追加したリンクファイル「RE01_1500KB.icf」を設定しています。ユーザがリンクファイルを設定する場合は、図 8-4 を参考に使用するリンクファイルを設定してください。

リンク設定ファイル : \$PROJ_DIR\$\config\RE01_1500KB.icf

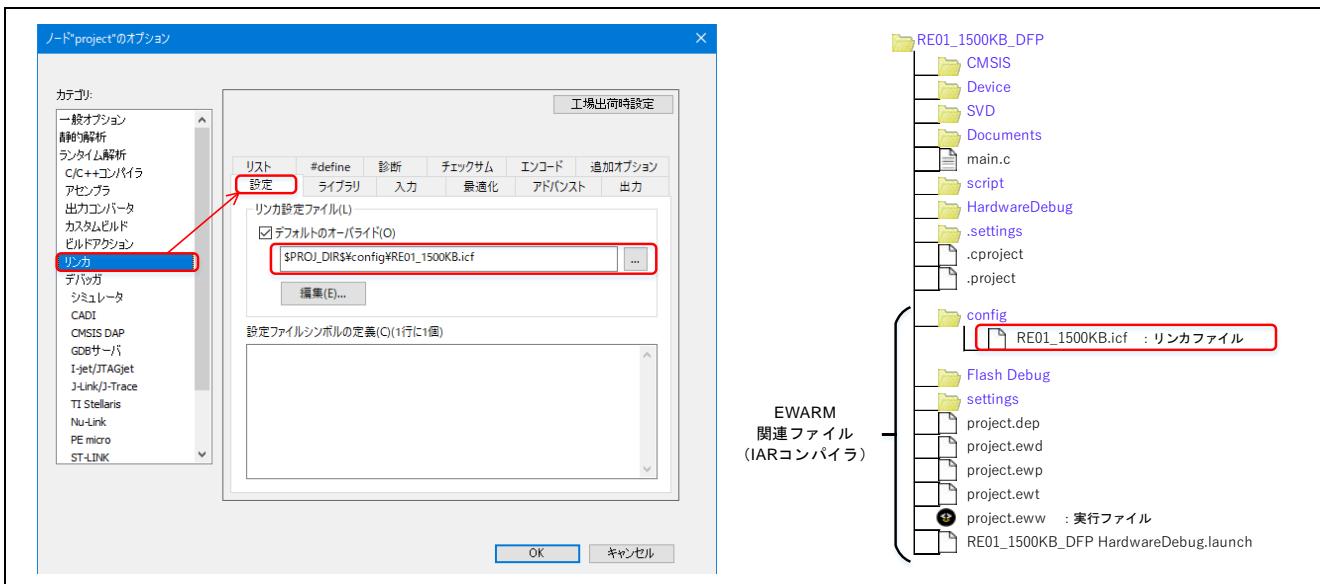


図 8-4 リンカファイルの設定例（EWARM）

RAM 配置用セクションについては、6.6.1 RAM 配置セクションによる RAM 配置方法 を参照してください。

MTB 用領域については 9.1 デバッグ用 MTB 領域の確保 を参照してください。

8.1.3 インクルードディレクトリの設定

本パッケージのドライバは、ヘッダファイルのインクルードをファイル名のみで行っているため、インクルードファイルの所在をコンパイル時に指定する必要があります。本パッケージでは、ドライバを使用するために必要なインクルードディレクトリを設定しています。

ユーザが設定する場合は、図 8-5 を参考に必要なパスを設定してください。

インクルードパス設定例

- \$PROJ_DIR\$\CMSIS\
- \$PROJ_DIR\$\CMSIS\Core\Include
- \$PROJ_DIR\$\CMSIS\Driver\Include
- \$PROJ_DIR\$\Device
- \$PROJ_DIR\$\Device\CMSIS_Driver\Include
- \$PROJ_DIR\$\Device\Config
- \$PROJ_DIR\$\Device\Driver\Include
- \$PROJ_DIR\$\Device\DSP_Lib\Include

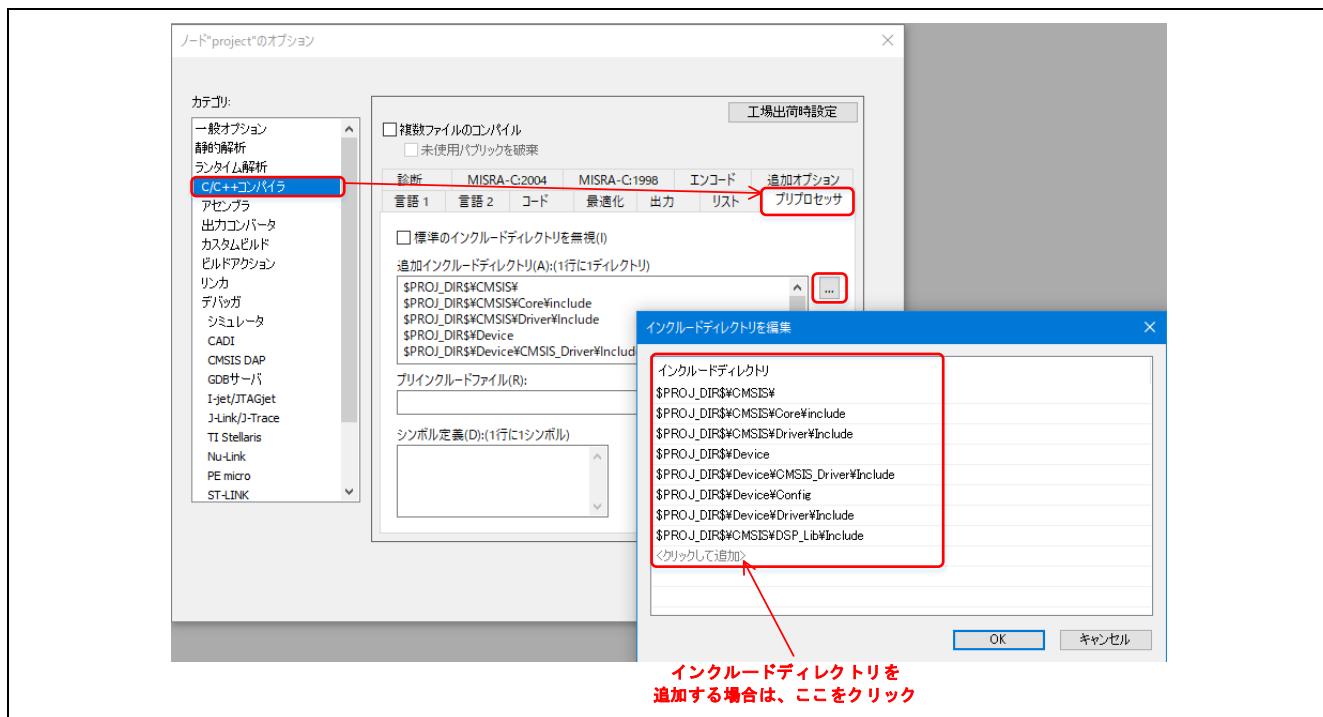


図 8-5 インクルードディレクトリ設定例 (EWARM)

8.2 e² studio 編

e² studio のインストールディレクトリに格納されている実行ファイルをダブルクリックし、e² studio を起動してください。e² studio の起動から本パッケージのインポートまでの手順は 2.2 e² studio 編を参照してください。

プロジェクトの各設定は、プロジェクトのプロパティ画面で行います。図 8-6 を参考にプロパティ画面を表示してください。

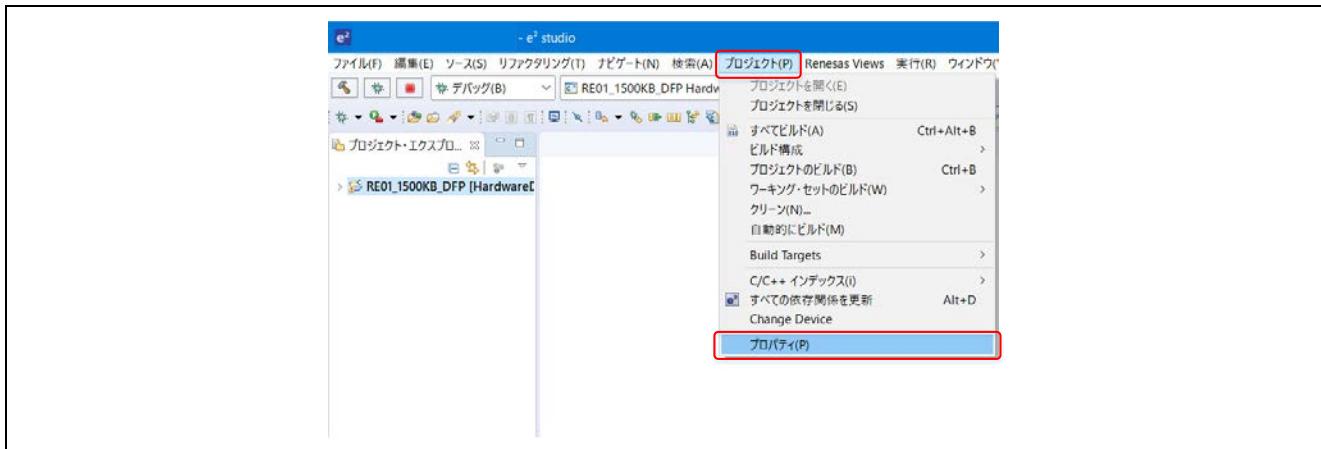


図 8-6 プロパティ画面表示方法 (e² studio)

8.2.1 ツールチェーンの設定

本パッケージでは、ツールチェーンに RE シリーズを設定しています。ユーザが設定する場合は、図 8-7 を参考に使用するツールチェーンを選択してください。

ツールチェーンに「RE」が表示されていない場合は、ツールチェーンのインストールおよび登録が必要です。e² studio の使い方については、開発環境のユーザーズマニュアルを参照してください。

現在の toolchain : RE

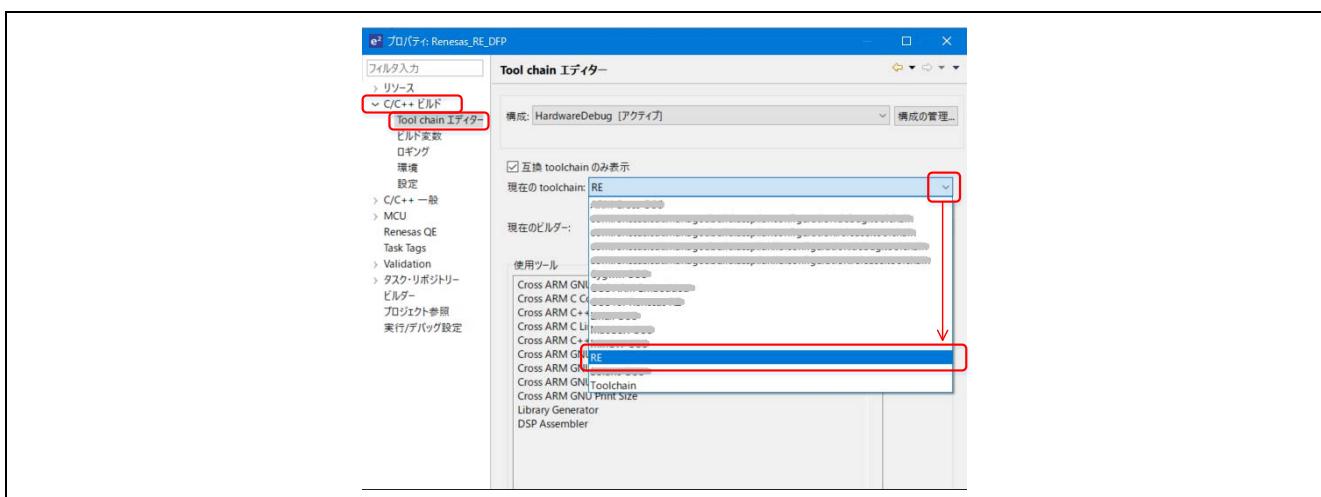


図 8-7 ツールチェーンの設定例 (e² studio)

8.2.2 リンカファイルの設定

本パッケージでは、RAM 配置用セクション定義および MTB 用領域を追加したリンカファイル「RE01_1500KB.ld」を設定しています。ユーザが設定する場合は、図 8-8 を参考に使用するリンカファイルを設定してください。

Script files : "\${workspace_loc:/\${ProjName}/script/RE01_1500KB.ld}"

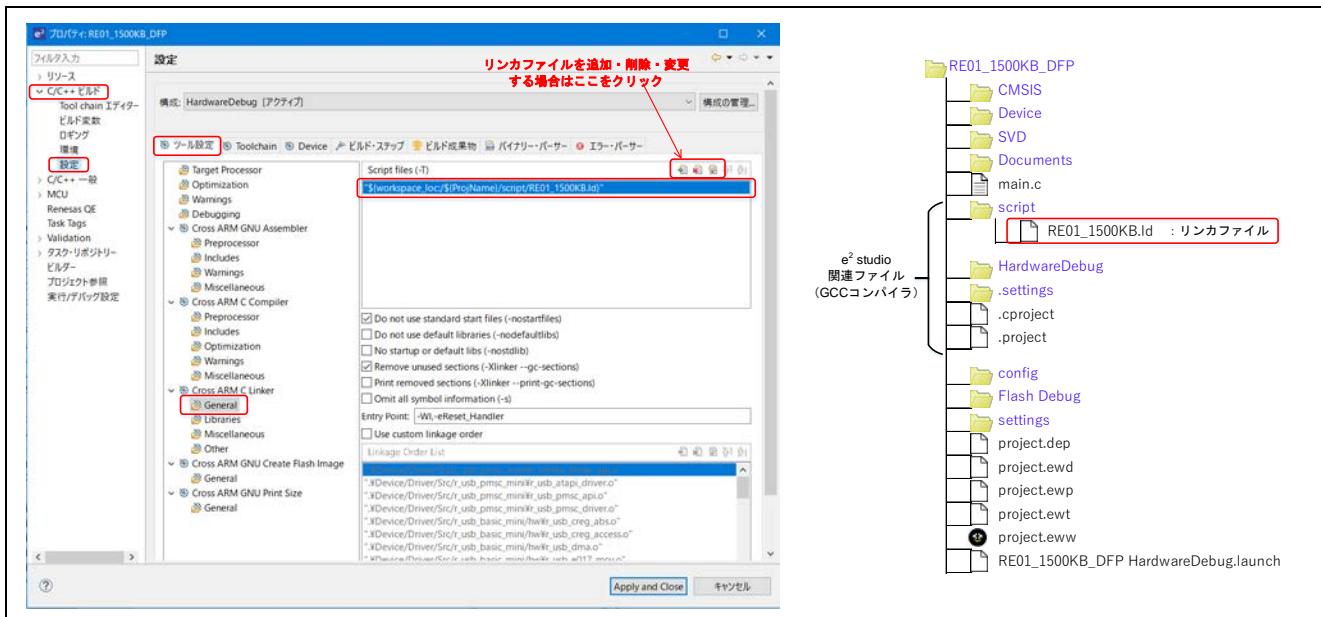


図 8-8 リンカファイル設定例 (e² studio)

RAM 配置用セクションについては、6.6.1 RAM 配置セクションによる RAM 配置 を参照してください。

MTB 用領域については 9.1 デバッグ用 MTB 領域 を参照してください。

8.2.3 インクルードパスの設定

本パッケージのドライバは、ヘッダファイルのインクルードをファイル名のみで行っているため、インクルードファイルの所在をコンパイル時に指定する必要があります。本パッケージでは、ドライバを使用するために必要なインクルードパスを設定しています。

ユーザが設定する場合は、図 8-9 を参考に必要なパスを設定してください。

インクルードパス設定例

- "\${workspace_loc:/\${ProjName}/Device}"
- "\${workspace_loc:/\${ProjName}/Device/Driver/Include}"
- "\${workspace_loc:/\${ProjName}/Device/CMSIS_Driver/Include}"
- "\${workspace_loc:/\${ProjName}/Device/Config}"
- "\${workspace_loc:/\${ProjName}/CMSIS/Core/Include}"
- "\${workspace_loc:/\${ProjName}/CMSIS/Driver/Include}"
- "\${workspace_loc:/\${ProjName}/CMSIS/DSP_Lib/Include}"
- \${ProjDirPath}/generate
- \${ProjDirPath}/src

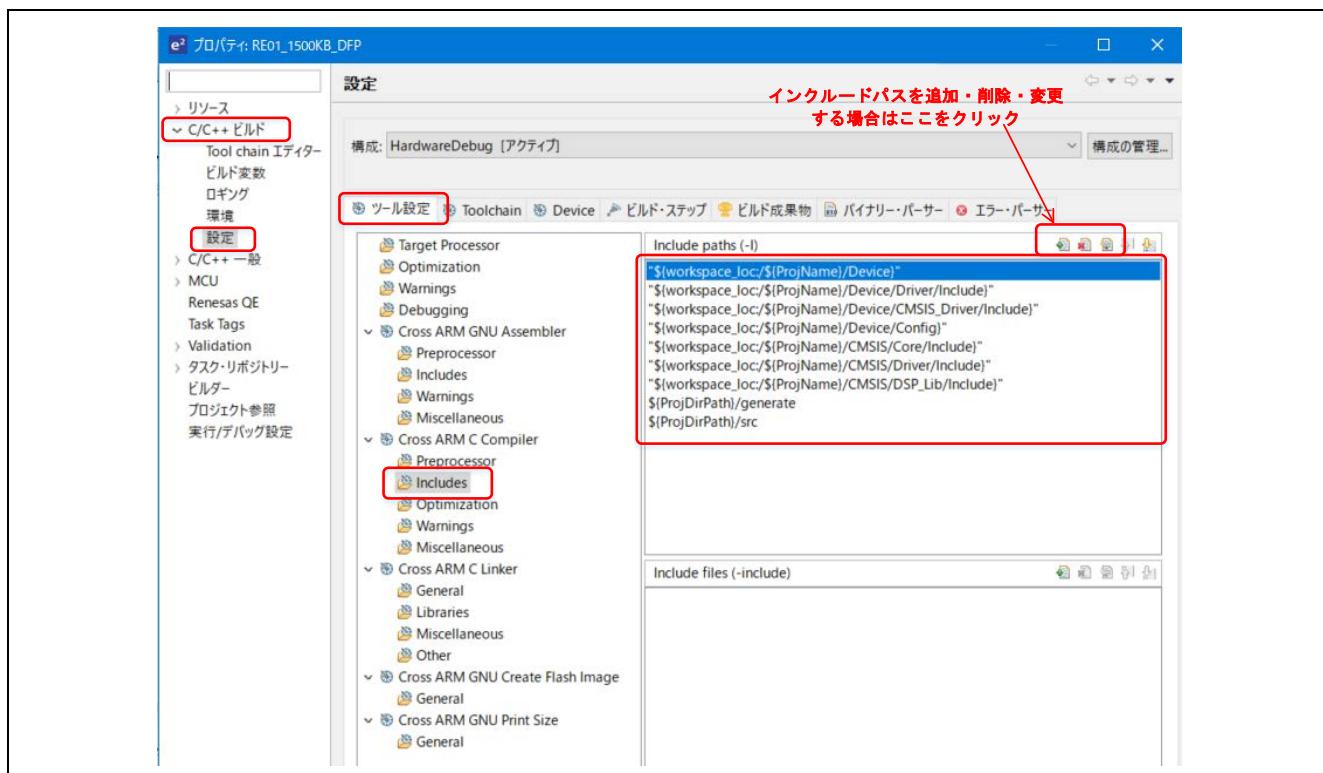


図 8-9 インクルードパス設定例（e² studio）

9. デバッガを使ってダウンロードしてみよう

本章は RE01 1500KB グループを例に説明します。以降、特に記述がない限り、その手順は RE01 256KB グループも同じです。

9.1 デバッグ用 MTB 領域の確保

本デバイスは、RAM の先頭 1KB を、デバッグ用の MTB 領域として空けておく必要があります。本パッケージで使用しているリンクファイルでは、RAM の先頭 1KB を MTB 用に空けています。本パッケージの RAM メモリマッピングを図 9-1 に示します。

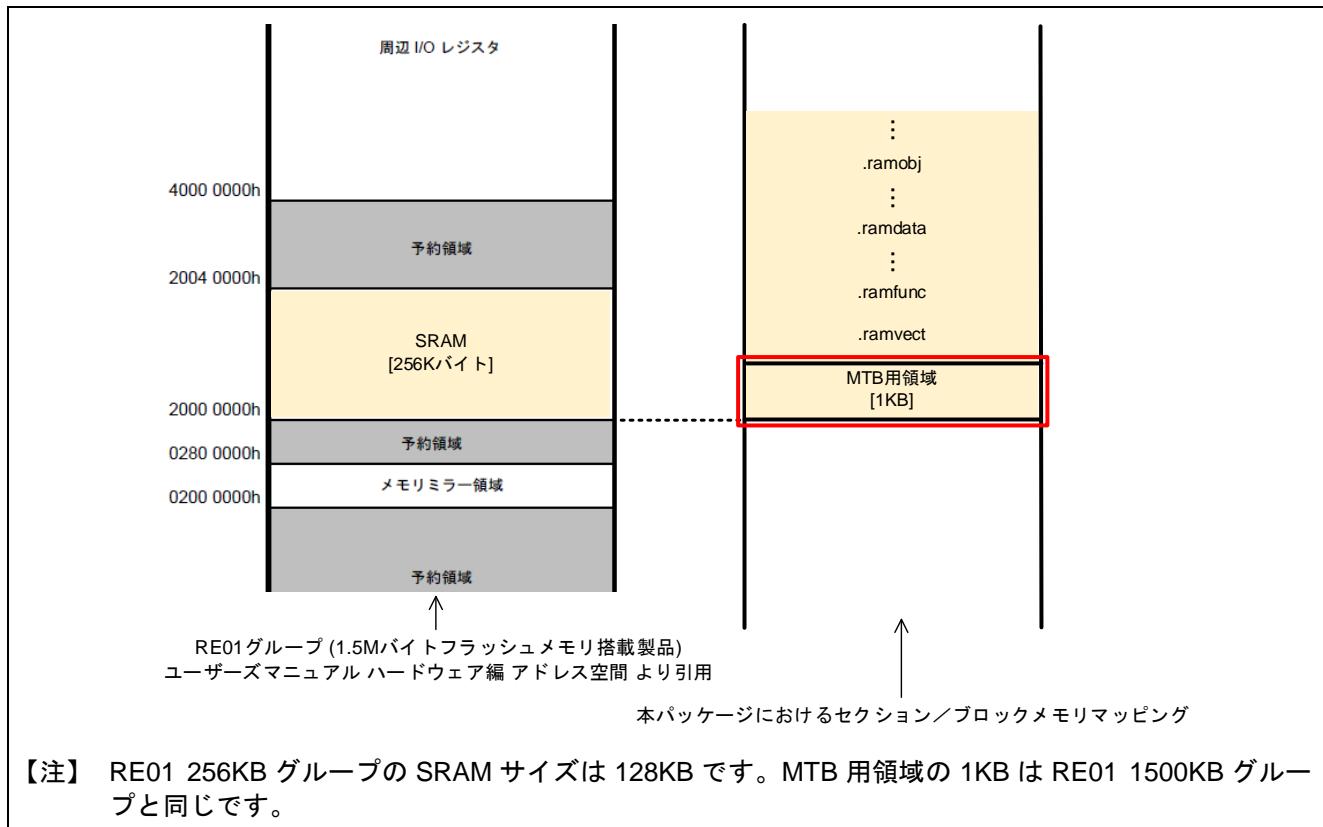


図 9-1 MTB 領域の確保例

9.2 低消費電力モード使用時の注意点

9.2.1 ソフトウェアブレークを設定できない条件

本デバイスは、ディープソフトウェアスタンバイモード、ソフトウェアスタンバイモードなどの低消費電力モードを持っています。低消費電力モード時は、オンチップデバッガ (OCD) から CPU レジスタなどの値を読み出しできなくなり、OCD が切断されます。

また本デバイスには、内蔵フラッシュメモリの書き換えができないモード（条件）があります。

本条件のいずれかに該当するアプリケーションのデバッグでは、内蔵フラッシュメモリ領域にソフトウェアブレークを設定しないでください。

内蔵フラッシュメモリにソフトウェアブレークを設定できない条件を以下に示します。

- デバイスの電力制御モードを High-Speed モード以外
- システムクロックが 1MHz 未満

低消費電力モード時のデバッグ接続方法、ブレークポイントの設定方法は、9.3 EWARM 編、9.4 e² studio 編で説明します。

9.3 EWARM 編

9.3.1 J-Link

本章では、ターゲットボード搭載の J-Link OB を使用する場合について説明します。

9.3.1.1 デバッガの選択

本パッケージでは、デバッガに「J-Link」を選択しています。ユーザが設定する場合は、図 9-2 を参考に使用するデバッガを選択してください。図 9-2 では、J-Link を選択した場合に確認が必要な項目を赤枠で示しています。

ドライバ : J-Link/J-Trac

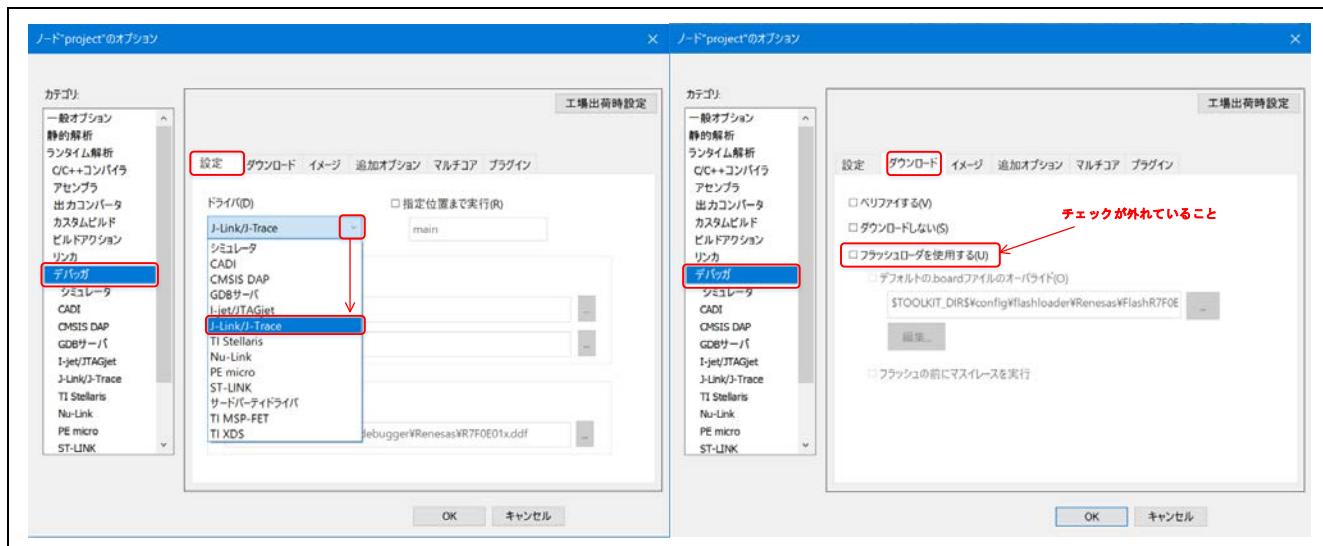


図 9-2 デバッガの J-Link 選択例 (EWARM)

9.3.1.2 J-Link の設定

本パッケージでは、デバッガに「J-Link」を選択した場合の初期設定を行っています。ユーザが設定する場合は、図 9-3 を参考に使用するデバッガの設定を行ってください。図 9-3 では、J-Link を選択した場合に確認が必要な項目を赤枠で示しています。

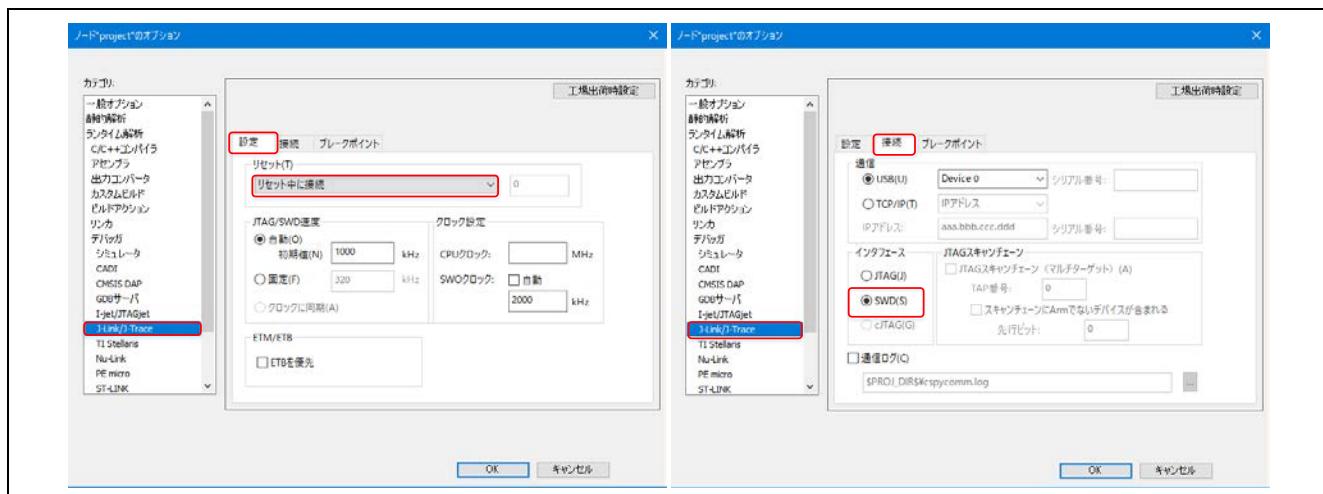


図 9-3 J-Link の設定例 (EWARM)

本パッケージでは、低消費電力モードを使用したプログラムをデバッグする場合の設定は行っていません。低消費電力モードを使用したプログラムをデバッグする場合の設定例を示します。

- EWARM でプロジェクトを開き、初めて J-Link OB を起動すると生成される[プロジェクト名]_Flash Debug.jlink ファイルを図 9-4 を参考に編集してください。

変更対象ファイル : [プロジェクト名]_Flash Debug.jlink

変更後 : LowPowerHandlingMode = 1

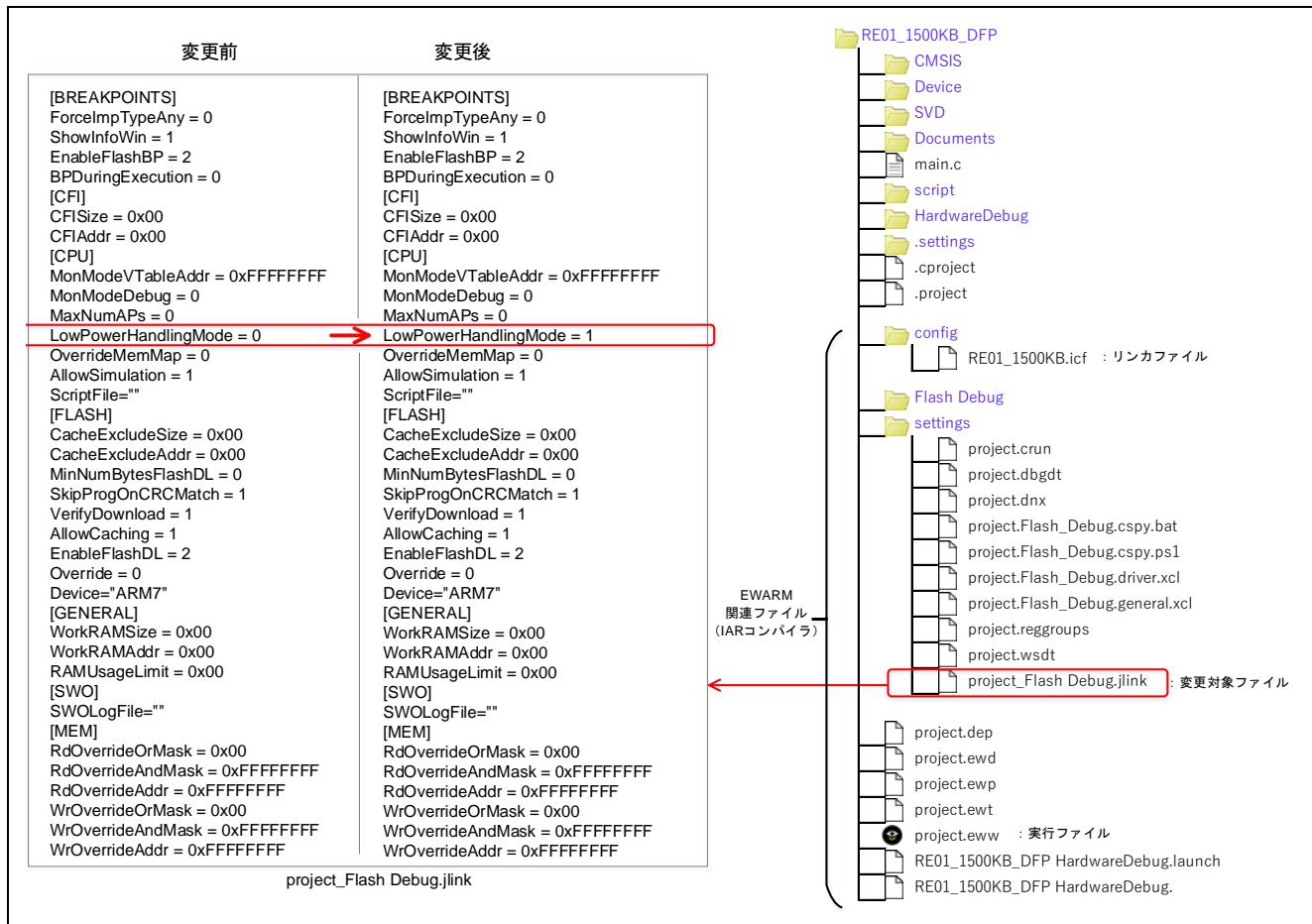


図 9-4 低消費電力モードを使用したプログラムのデバッグ設定例 (EWARM)

9.3.1.3 内蔵フラッシュメモリ領域へのソフトウェアブレーク設定禁止

本パッケージでは、内蔵フラッシュメモリ領域へのソフトウェアブレーク設定を禁止する設定は行っていません。ユーザが内蔵フラッシュメモリ領域へのソフトウェアブレーク設定を禁止する場合の設定例を示します。

- EWARM でプロジェクトを開き、初めて J-Link OB を起動すると生成される[プロジェクト名]_Flash Debug.jlink ファイルを図 9-5 を参考に編集し、任意の名前のスクリプトファイルを新規作成してください。

変更対象ファイル : [プロジェクト名]_Flash Debug.jlink

変更後 : ScriptFile="project.JLinkScript" (新規作成ファイルのパスと名称にあわせる)

新規作成ファイル : project.JLinkScript (ファイル名は任意)

新規作成ファイル内の記述 :

```
int ConfigTargetSettings(void) {
    JLINK_ExecCommand("SetWorkRAM 0x20008000-0x20017FFF");
    JLINK_ExecCommand("DisableFlashBPs");
    return 0;
}
```

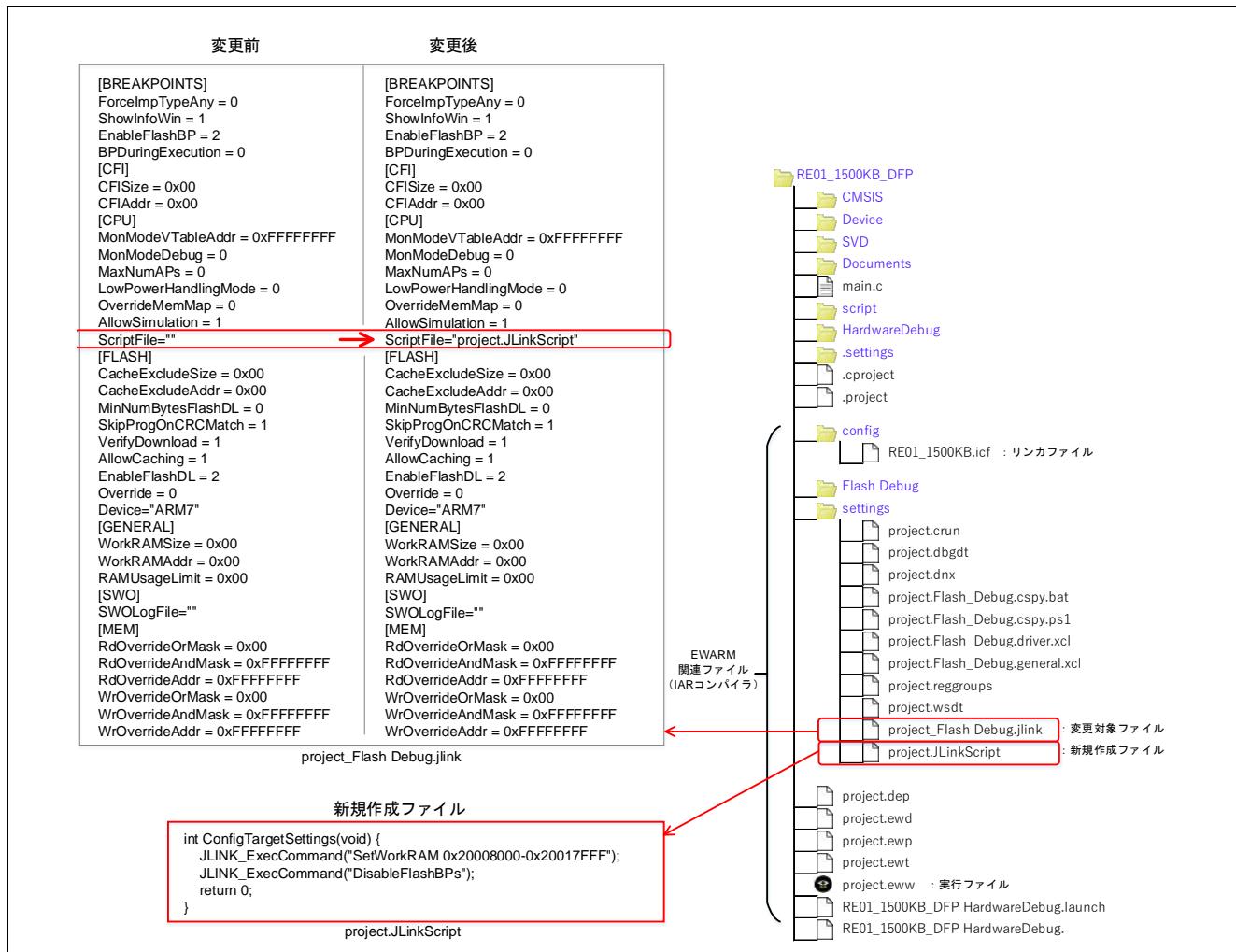


図 9-5 内蔵フラッシュメモリ領域へのソフトウェアブレークポイント設定禁止例 (EWARM)

9.3.2 I-Jet

9.3.2.1 デバッガの選択

本パッケージでは、デバッガに「J-Link」が設定されています。デバッガに I-Jet を使用する場合は、図 9-6 を参考に選択してください。図 9-6 では、I-Jet を選択した場合に確認が必要な項目を赤枠で示しています。

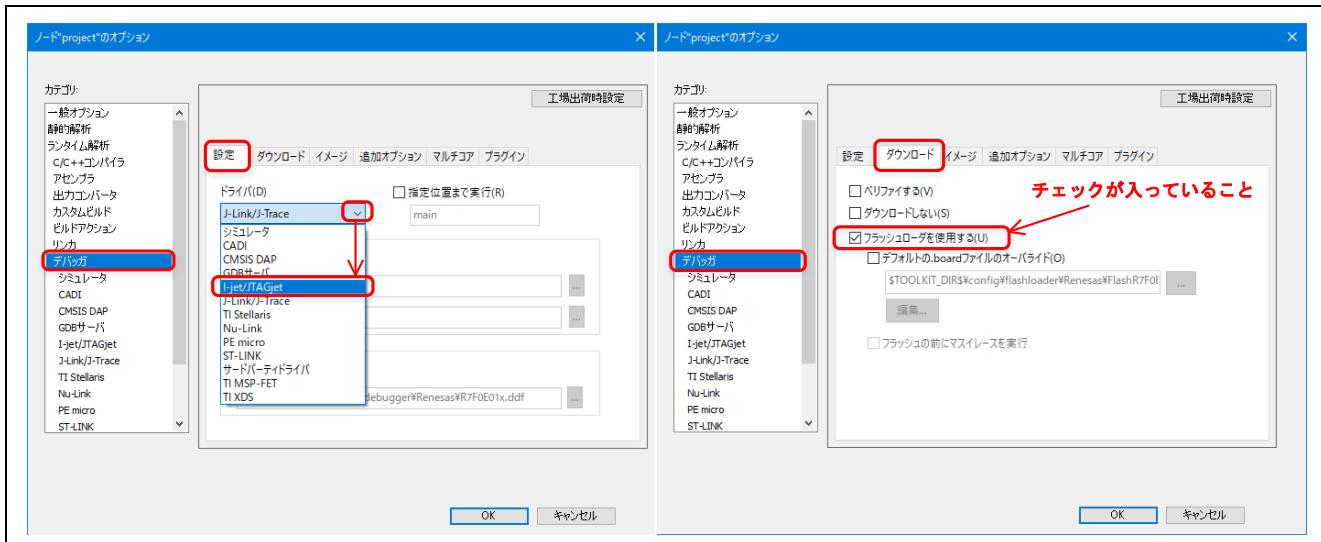
ドライバ : I-jet/JTAGjet

図 9-6 デバッガの I-Jet 選択例 (EWARM)

9.3.2.2 I-Jet の設定

デバッガに I-Jet を使用する場合は、図 9-7 を参考に設定してください。図 9-7 では、I-Jet を選択した場合に確認が必要な項目を赤枠で示しています。

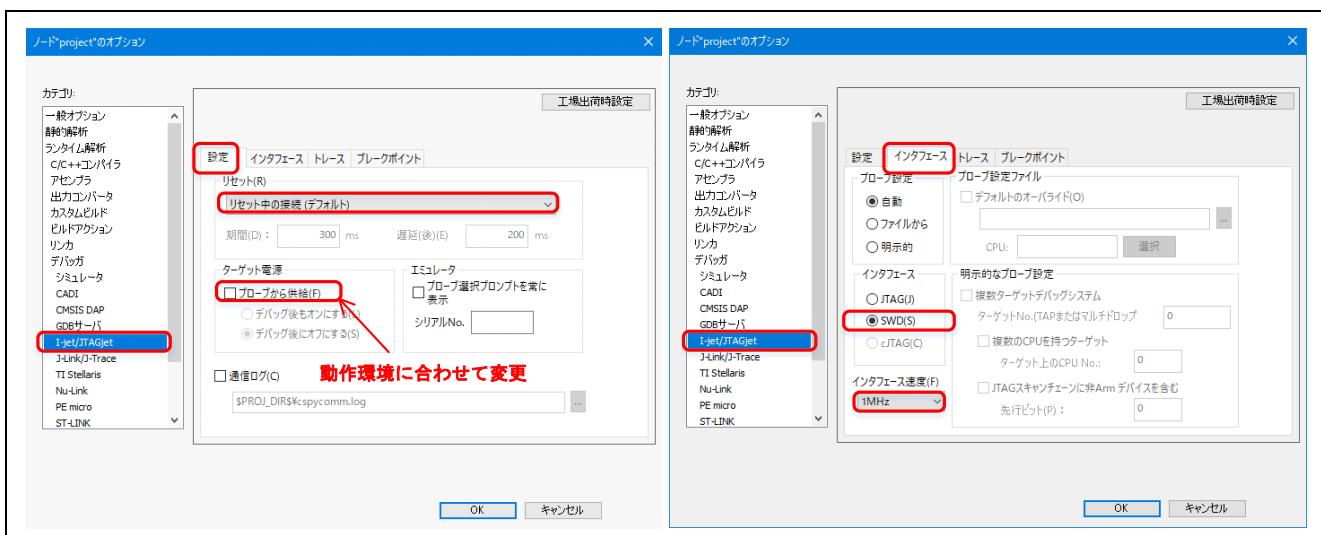


図 9-7 I-Jet の設定例 (EWARM)

9.4 e² studio 編

9.4.1 J-Link

本章では、ターゲットボード搭載の J-Link OB を使用する場合について説明します。

9.4.1.1 デバッガの選択

本パッケージでは、デバッガに「J-Link」を設定しています。ユーザが設定する場合は、図 9-8 を参考に使用するデバッガを選択してください。図 9-8 では、J-Link を選択した場合に確認が必要な項目を赤枠で示しています。

Debug hardware : J-Link ARM

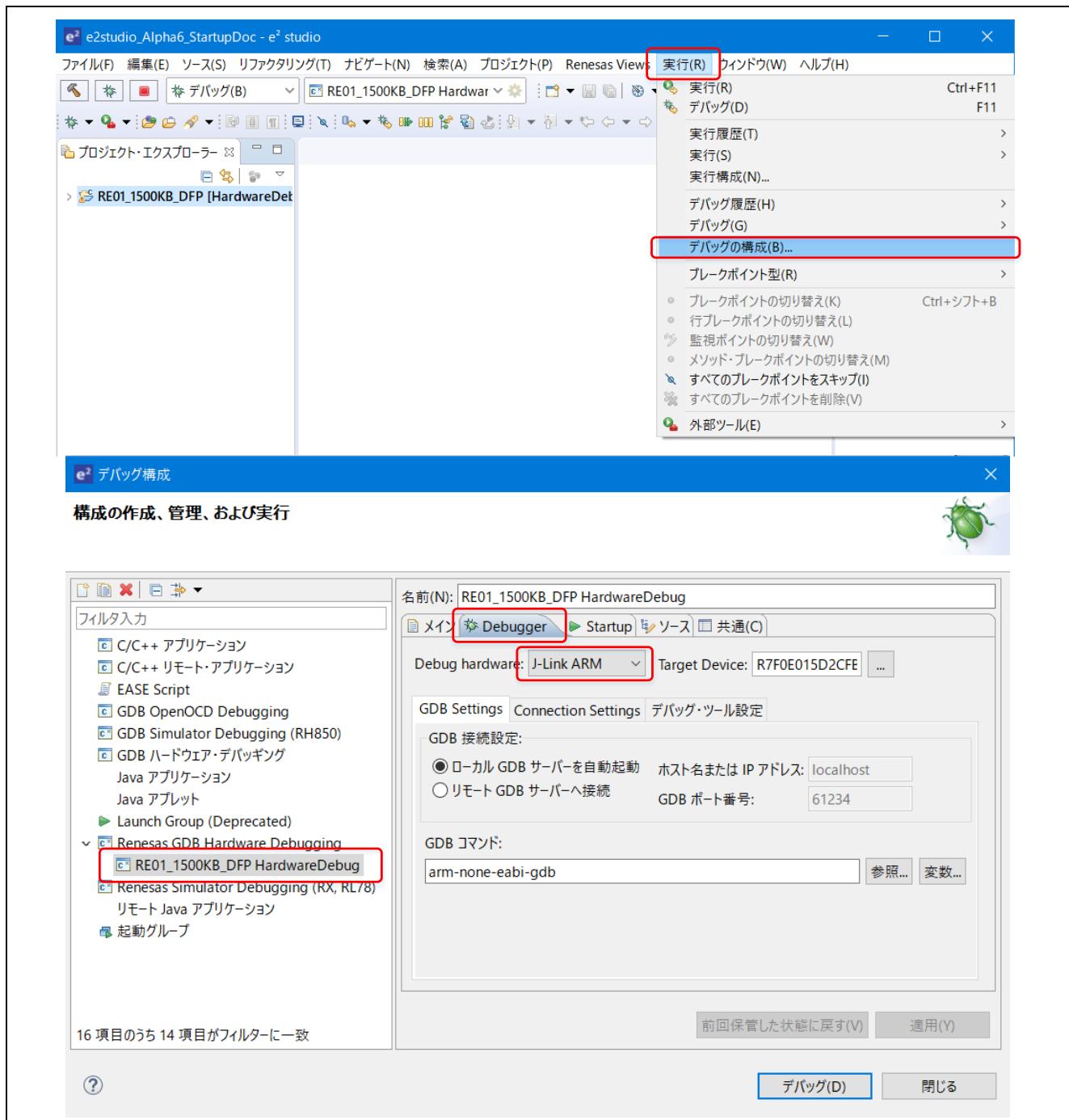


図 9-8 デバッガの J-Link 選択例 (e² studio)

9.4.1.2 J-Link の設定

本パッケージでは、デバッガに「J-Link」を選択した場合の初期設定を行っています。ユーザが設定する場合は、図 9-9 を参考に使用するデバッガの設定を行ってください。図 9-9 では、デバッガに J-Link を選択した場合に確認が必要な項目を赤枠で示しています。

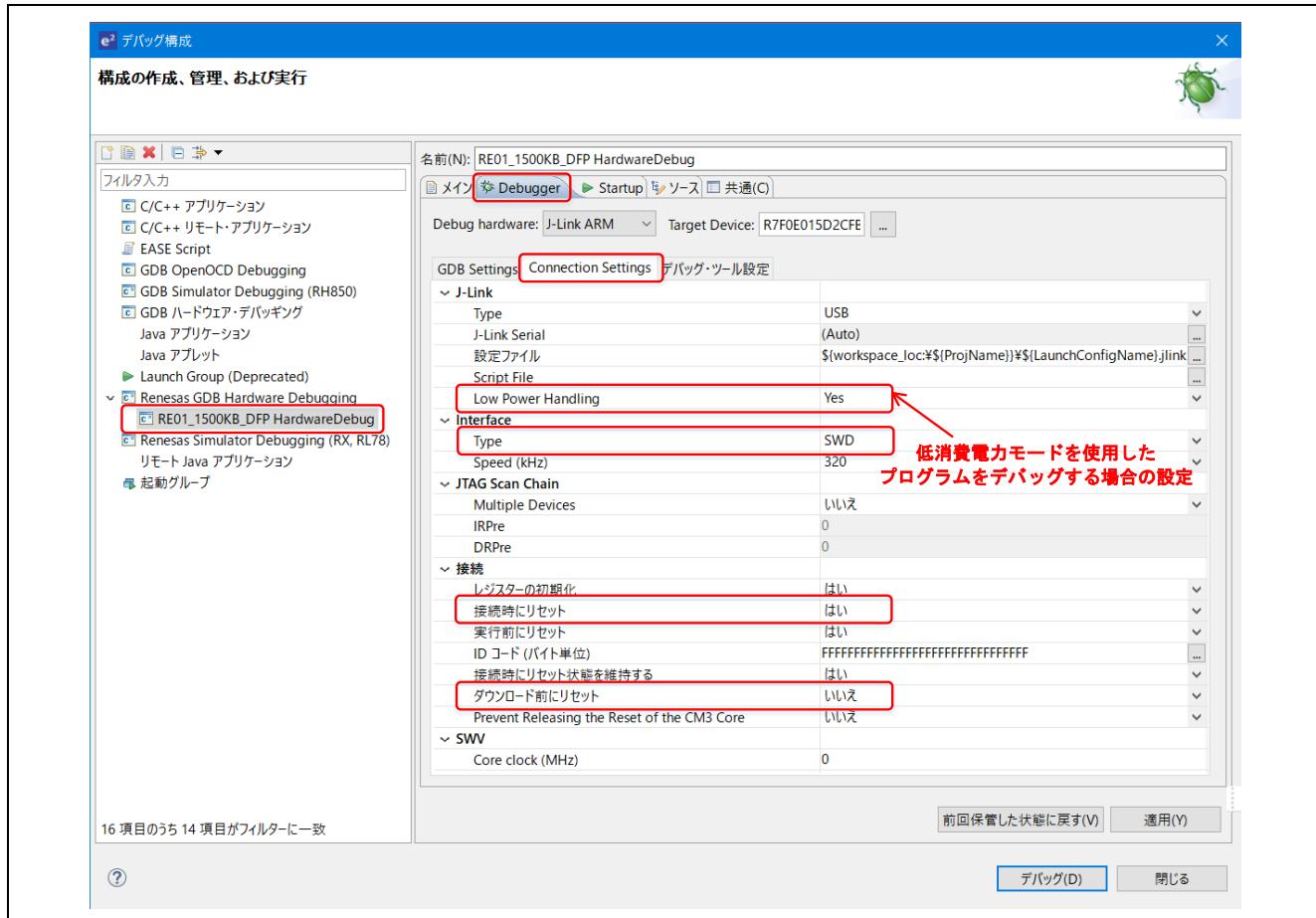


図 9-9 J-Link の設定例 (e² studio)

9.4.1.3 内蔵フラッシュメモリ領域へのソフトウェアブレーク設定禁止

本パッケージでは、デバッガに「J-Link」を選択した場合、内蔵フラッシュメモリ領域へのソフトウェアブレーク設定を禁止する設定を行っています。ユーザが設定する場合は、図 9-10 を参考に設定を行ってください。図 9-10 では、内蔵フラッシュメモリ領域へのソフトウェアブレーク設定を禁止する場合に確認が必要な項目を赤枠で示しています。

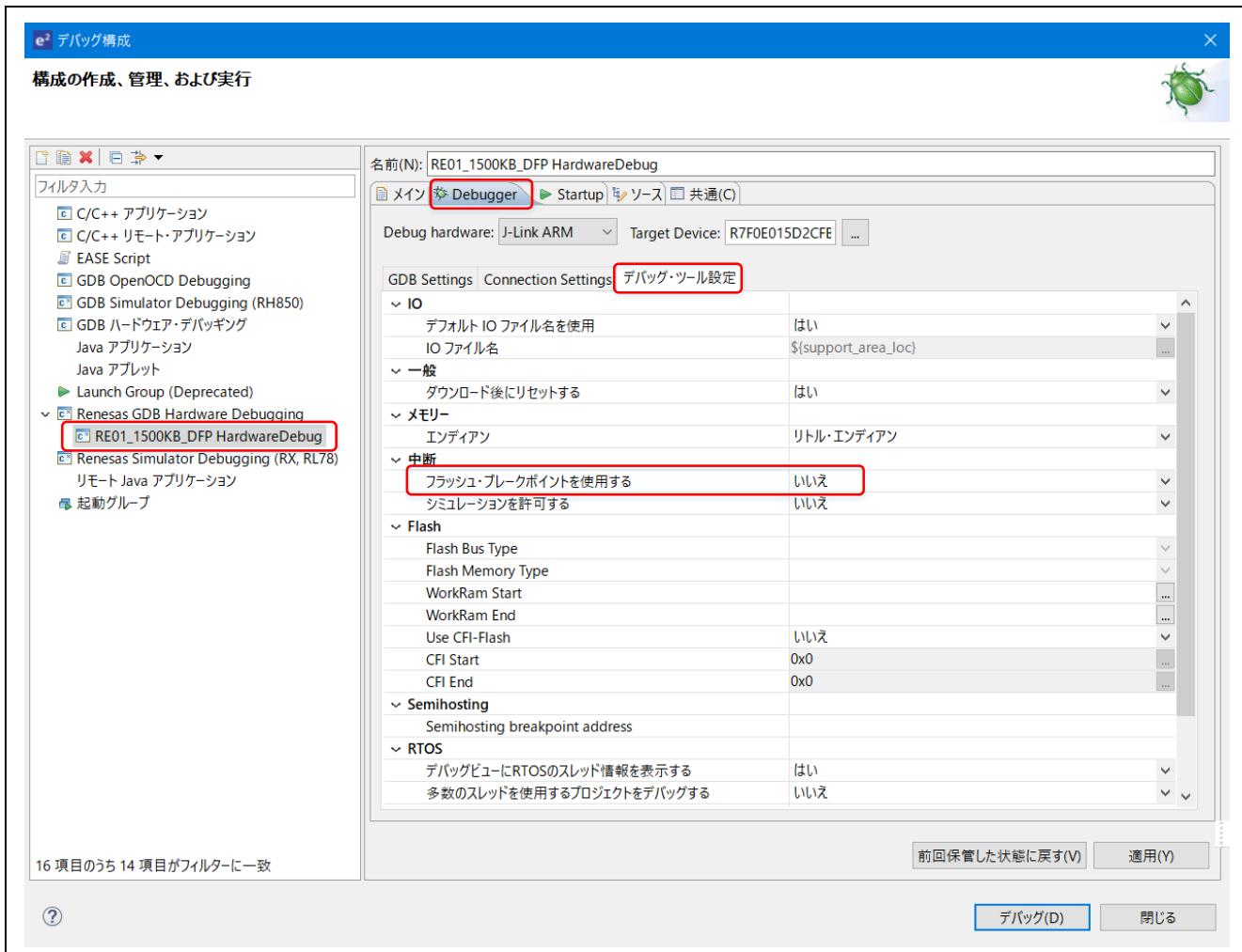


図 9-10 内蔵フラッシュメモリ領域へのソフトウェアブレークポイント設定禁止例

9.4.1.4 RAM 領域へのブレークポイント設定

RAM 領域にはハードウェアブレークポイントを設定できません。ハードウェアブレークポイントとソフトウェアブレークポイントを切り替える使用例を説明します。

1. ブレークポイントの初期設定をハードウェアブレークポイントに設定

特定の条件において、内蔵フラッシュメモリにソフトウェアブレークポイントを設定できないため、通常はハードウェアブレークポイントを設定します。

図 9-11 を参考に、初期設定をハードウェアブレークポイントに設定してください。



図 9-11 ブレークの初期設定確認例 (e² studio)

2. RAM 領域にはソフトウェアブレークポイントを設定

RAM 領域にはハードウェアブレークポイントを設定できないため、RAM 領域ではソフトウェアブレークポイントに切り替えて設定します。

図 9-12 を参考に、ブレークポイントの種類を切り替えて設定してください。

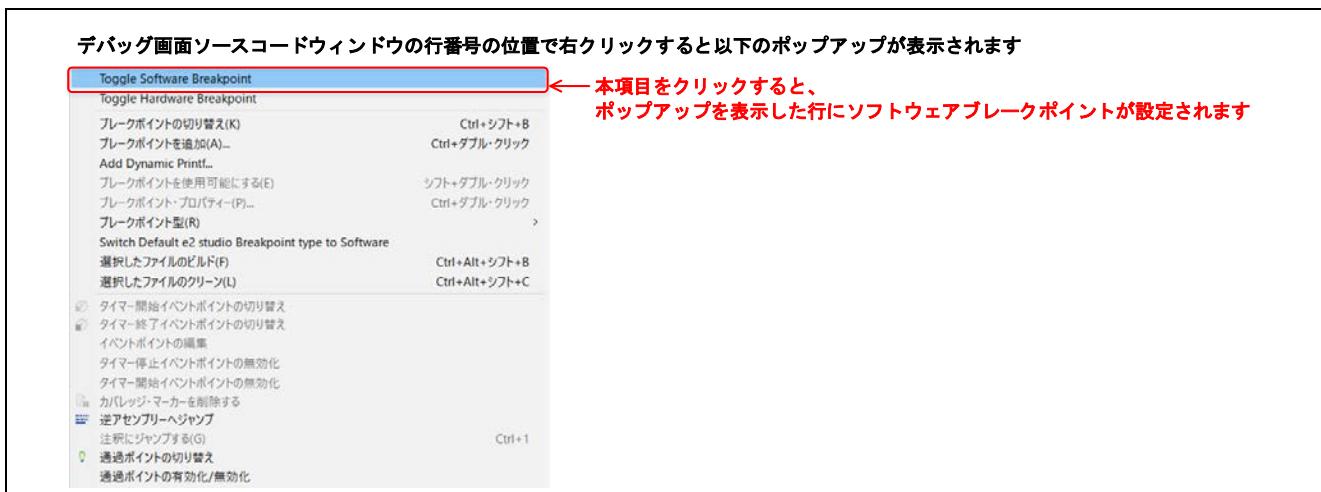


図 9-12 ソフトウェアブレークポイントの設定例 (e² studio)

10. 付録

モジュール/機能	対応 ドライバ	周辺機能名	割り込み名	調歩同期式			クロック同期式			スマートカード							
				送信	受信	送受信	送信	受信	送受信	送信	受信	送受信					
シリアルコミュニケーション インターフェース	R_USART	SCI	SCIn_TXI	✓		✓	✓	✓	✓	✓	✓	✓					
			SCIn_RXI		✓	✓		✓	✓		✓	✓					
			SCIn_ERI		✓	✓		✓	✓		✓	✓					
モジュール/機能	対応 ドライバ	周辺機能名	割り込み名	マスター		スレーブ											
				送信	送受信	送信	送受信										
シリアルペリフェラル インターフェース	R_SPI	SPI	SPIn_SPTI	✓	✓	✓	✓										
			SPIn_SPR1		✓												
			SPIn_SPII	✓	✓												
			SPIn_SPEI			✓	✓										
I2Cバスインタフェース	R_I2C	RIIC	SPIn_SPTEND	✓		✓											
			IICn_RXI														
			IICn_TXI														
			IICn_TEI														
IICn_EEI			IICn_EEI														
モジュール/機能	対応 ドライバ	周辺機能名	割り込み名	DTC/DMACを使用しない場合		DMAC使用時		DTC使用時									
				ADC140_ADI	✓												
14ビットA/Dコンバータ	R_ADC	S14AD	ADC140_GBADI	✓													
			ADC140_GCADI	✓													
			ADC140_CMPAI	✓													
			ADC140_CMPBI	✓													
			ADC140_WCMPM	✓													
			ADC140_WCMPUM	✓													
			DMACn_INT				✓										
モジュール/機能	対応 ドライバ	周辺機能名	割り込み名														
DMAコントローラ	R_DMAC	DMAC	DMACn_INT														
モジュール/機能	対応 ドライバ	周辺機能名	割り込み名														
データトランസフアントローラ	R_DTC	DTC	DTC_COMPLETE														
モジュール/機能	対応 ドライバ	周辺機能名	割り込み名														
2Dグラフィックデータ変換回路	R_GDT	GDT	GDT_DATII GDT_DATOI DMACn_INT (n=0~3)														

↑
周辺機能の割り込み名

図 10-1 ドライバが使用する割り込み一覧

11. トラブルシューティング

11.1 Q : ビルドエラーが発生する

A-1 : インクルードディレクトリが設定されていますか？

ファイルをインクルードする場合、コンパイラはインクルードディレクトリに指定された場所を検索します。インクルードファイルの所在を、インクルードディレクトリで指定してください。

- EWARM をご使用の場合、8.1.3 インクルードディレクトリの設定を参照してください。
- e² studio をご使用の場合、8.2.3 インクルードパスの設定を参照してください。

A-2 : 必要なファイルがコンパイル対象になっていますか？

- EWARM をご使用の場合

EWARM のワークスペースウィンドウにあるファイルがコンパイル対象ファイルです。不足している場合は、EWARM のワークスペースウィンドウの任意の場所で右クリックし、追加を選択してください。フォルダは「グループの追加」を、ファイルは「ファイルの追加」を選択してください。

- e² studio をご使用の場合

e² studio のプロジェクト・エクスプローラにあるファイルに斜線がある場合は、コンパイル対象外になっています。任意のファイルで右クリックし「リソース構成」→「ビルドから除外」を選択し、チェックボックスを外してください。

A-3 : 対象デバイスのコンパイラが選択されていますか？

コンパイル時は、対象デバイスのコンパイラを選択してください。

- EWARM をご使用の場合、8.1.1 ターゲットプロセッサの設定 を参照してください。
- e² studio をご使用の場合、8.2.1 ツールチェーンの設定 を参照してください。

11.2 Q : ドライバ関数を実行すると HardFault Error が発生する

A-1 : RAM 配置用セクションに配置したプログラムを RAM に展開しましたか？

本パッケージのドライバ関数は、RAM 配置用セクションを使用して任意のプログラムを RAM に展開して使用します。ユーザプログラムの先頭で R_SYS_CodeCopy 関数を実行してプログラムを RAM に展開してください。

プログラムの RAM 配置については、6.6 プログラムの RAM 配置 を参照してください。

A-2 : 内蔵フラッシュメモリが遮断中に内蔵フラッシュメモリにアクセスしていませんか？

内蔵フラッシュメモリを遮断後も動作を継続する場合、以下のような原因が考えられます

- NVIC 関数、標準関数など、内蔵フラッシュメモリに配置されたプログラムを実行した場合
- RAM に配置したはずのプログラムが、RAM に期待どおり配置できていない場合
- 内蔵フラッシュメモリに配置されたプログラムから実行された関数がインライン展開され、期待どおりに RAM に配置できていない場合

プログラムの RAM 配置については、6.6 プログラムの RAM 配置 を参照してください。

11.3 Q : ドライバ関数を実行しているが周辺機能が動作しない

A-1 : ドライバ関数の設定が問題無くできていますか？

ドライバ関数でエラーを検出していることが考えられます。

ドライバ関数の戻り値を確認し、エラー値が返っていないかをご確認ください。

特に割り込み設定が不足していることでエラー値が返っている事例が多く発生しています。

周辺機能ドライバを使用している場合の割り込みの設定については、6.3.4 ドライバ関数 を参照してください。

A-2 : 周辺機能で使用するクロックが発振していますか？

周辺機能動作を開始する前に使用するクロックのクロックソースが発振している必要があります。

クロックソースの発振開始は、r_core_cfg.h を設定しデバイスの動作開始時に制御する方法と、デバイスの動作開始後にユーザプログラムで R_SYSTEM ドライバのクロック制御関数で制御する方法があります。

クロック設定については、6.4 クロック設定を参照してください。

11.4 Q : ドライバ関数の戻り値は正常だが、周辺機能の端子から期待した入出力が確認できない

A-1 : R_PIN ドライバの端子設定関数は編集しましたか？

周辺機能ドライバ関数は、R_PIN ドライバ関数を実行して周辺機能で使用する端子の設定を行います。任意の端子が周辺機能で使用されるよう、R_PIN ドライバ関数の処理を編集してください。

端子設定については、6.5.2 ドライバ関数の編集 を参照してください。

A-2 : 対応 IO 電源ドメインに給電していますか？

本デバイスは、複数の IO 電源ドメインを持ち、ドメインごとに電源供給／遮断を制御できます。周辺機能で使用する端子が配置されている IO 電源ドメインに電源供給してください。

対応 IO 電源ドメインについては、6.2.1 対応電源ドメイン を参照してください。

A-3 : IO 電源ドメインの不定値伝搬抑止機能が有効になっていますか？

対応 IO 電源ドメインに電源供給しても期待した入出力が確認できない場合は、不定値伝搬抑止機能が有効になっていることが考えられます。リセット解除後は、IOVCC を除く IO 電源ドメインの不定値伝搬抑止機能が有効なため、電源供給されるドメインの不定値伝搬抑止機能は無効にしてください。

IO 電源ドメインの不定値伝搬抑止機能を制御するドライバ関数については、6.2.2 ドライバ関数 を参照してください。

11.5 Q : クロックや消費電力低減機能のレジスタに書き込んだが反映されない

A : レジスタライトプロテクトが有効になっていますか？

リセット解除後は、レジスタライトプロテクション機能が有効なため、該当レジスタへの書き込みができません。クロックや消費電力低減機能のレジスタに書き込みを行う場合は、レジスタライトプロテクション機能を無効にしてください。

レジスタライトプロテクション機能の使い方については、7.2.2.5 レジスタライトプロテクト制御 を参照してください。

11.6 Q : 周辺機能のレジスタに書き込んだが反映されない

A : モジュールストップ機能が有効になっていませんか？

リセット解除後は、一部機能を除きモジュールストップ機能が有効です。周辺機能を使用する前に、モジュールストップ機能を無効してください。

モジュールストップ機能の使い方については、7.2.2.4 モジュールストップ制御 を参照してください。

11.7 Q : デバッガを使ってターゲットボードにダウンロードできない

A-1 : デバッガの設定を確認してください

- EWARM で J-Link をご使用の場合、9.3.1 J-Link を参照してください。
- EWARM で I-Jet をご使用の場合、9.3.2 I-Jet を参照してください。
- e² studio で J-Link をご使用の場合、9.4.1 J-Link を参照してください。

A-2 : 電源は正しく供給できていますか？

I-Jet をご使用の場合、ターゲットボードへの電源供給をデバッガもしくは外部からで選択することができます。デバッガの設定を動作環境に合わせて設定してください。

I-Jet の設定については、9.3.2.2 I-Jet の設定 を参照してください。

11.8 Q : デバッガを接続したが動作しない

A : デバッガの設定を確認してください

- EWARM で J-Link をご使用の場合、9.3.1 J-Link を参照してください。
- EWARM で I-Jet をご使用の場合、9.3.2 I-Jet を参照してください。
- e² studio で J-Link をご使用の場合、9.4.1 J-Link を参照してください。

11.9 Q : 割り込みが発生しない

A-1 : 割り込みの設定が不足していませんか？

周辺機能ドライバを使用して割り込みを制御する場合は、周辺機能ドライバ内で R_SYSTEM の割り込み制御関数を実行するため、ユーザプログラムで実行する必要はありません。

周辺機能ドライバを使用せずに割り込みを制御する場合は、ユーザプログラムで R_SYSTEM ドライバの割り込み制御関数を実行してください。また、動作環境にあわせて r_system_cfg.h の定義値を編集してください。

割り込み設定については、6.3 割り込み制御を参照してください。

A-2 : 一つの IRQ 番号を複数の割り込み要因に割り当てていませんか？

複数の割り込み要因に対して、同一の IRQ 番号を割り当てるることはできません。

割り込み要因と IRQ 割り込み番号の割り当てについては、6.3.2 IRQ 番号の割り当ておよび 6.3.3 r_system_cfg.h の編集 を参照し設定を確認してください。

A-3 : コールバック関数を登録していますか？

周辺機能ドライバを使用する場合は、各ドライバの仕様に従いコールバック関数を登録する必要があります。設定方法は、4 章に示すドライバ仕様書を参照してください。

周辺機能ドライバを使用しない場合は、ユーザプログラムで R_SYSTEM ドライバ関数を実行してコールバック関数を登録する必要があります。設定方法は、6.3.4 ドライバ関数を参照してください。

12. RE01 1500KB グループと RE01 256KB グループ間でのプログラム移行時の注意事項

RE01 グループ間でプログラムを移行する際、製品仕様やドライバ仕様に応じてプログラム変更が必要です。

- グループ間の製品仕様差異をご確認ください。
 - 端子割り当てをご確認ください。
 - メモリ領域にご注意ください。
 - 搭載機能の違いをご確認ください。
 - 周辺機能は同じでも搭載 ch 数が異なることがあります。
 - 周辺機能は同じでも電源ドメインが異なることがあります。
 - 動作モードと状態遷移の違いをご確認ください。
- 製品仕様は、各デバイスの「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。
- グループに対応した CMSIS Driver Package をお使いください。
ドライバ仕様は、各デバイスの CMSIS Driver Package に同梱された、4 章に示すドライバ仕様書を参照してください。

12.1 モード遷移例

RE01 グループ間でのプログラム移行の注意点の代表例として、消費電力低減機能におけるモード遷移を説明します。

RE01 グループ間の製品仕様は、動作モードおよび状態遷移に仕様差異があります。

ドライバ仕様は、R_SYSTEM ドライバ関数と R_LPM ドライバ関数に仕様差異があります。

12.1.1 遷移例 1：ブーストモードと低リーク電流モード間の遷移

BOOST_OPE ⇒ VBB_OPE 遷移、および VBB_OPE ⇒ BOOST_OPE 遷移時のドライバ関数使用例を図 12-2 に示します。製品移行時のプログラム変更箇所を赤文字で示します。

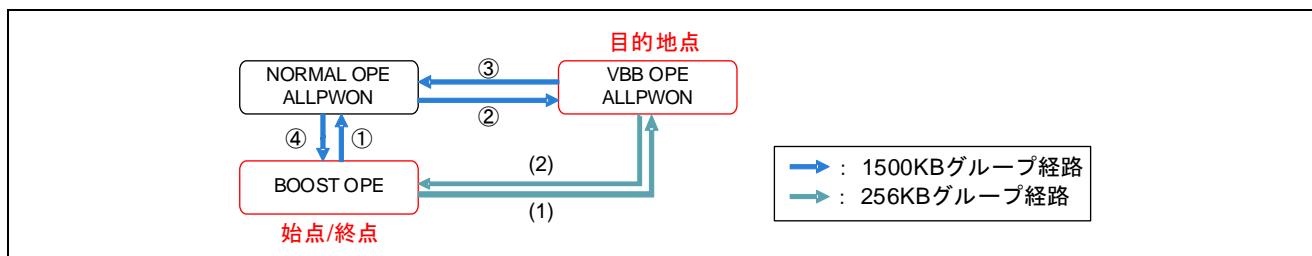


図 12-1 BOOST_OPE ⇒ VBB_OPE(ALLPWON) ⇒ BOOST_OPE 遷移図

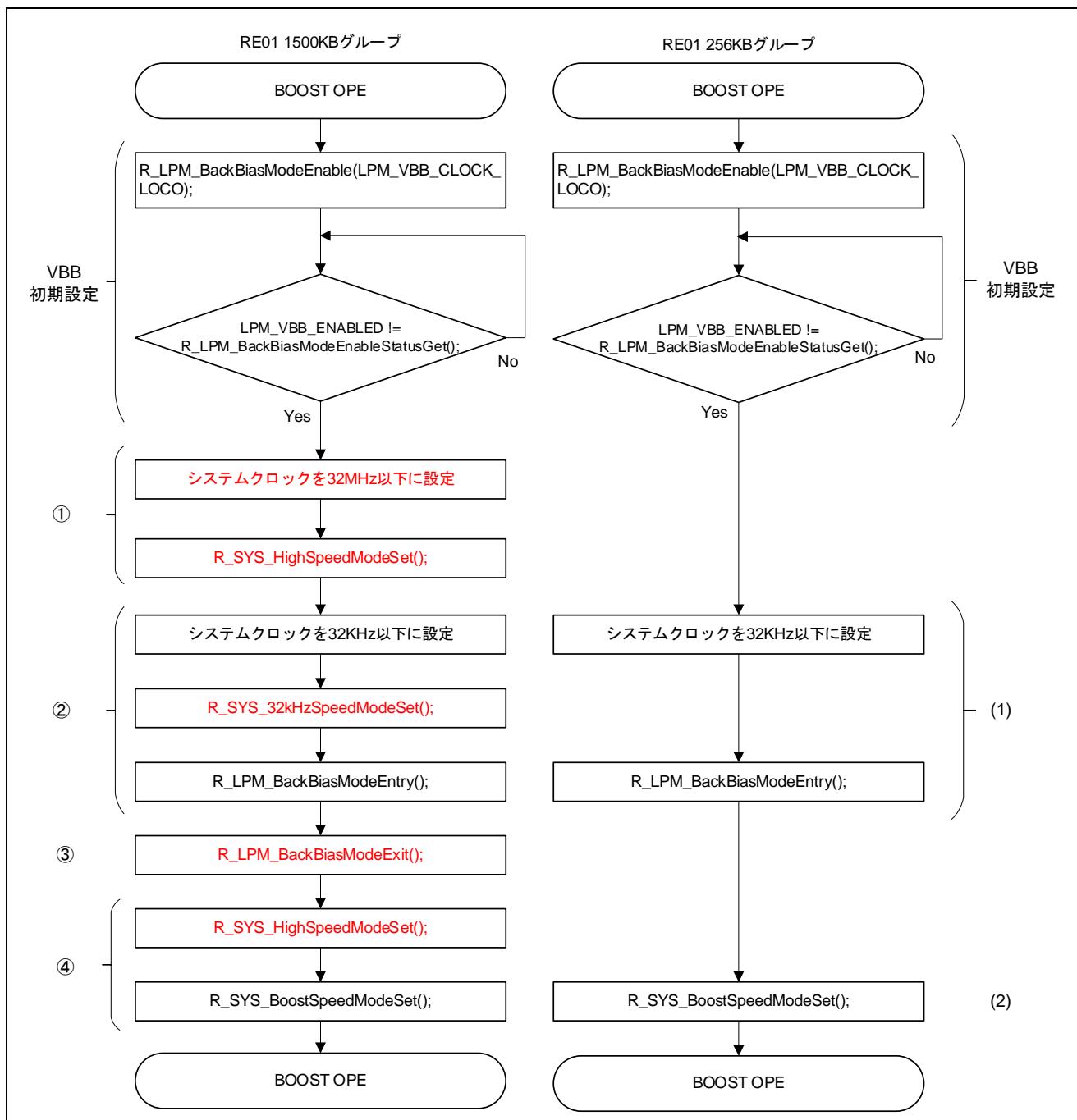


図 12-2 BOOST_OPE ⇒ VBB_OPE(ALLPWON) ⇒ BOOST_OPE 遷移のドライバ関数フロー

12.1.2 遷移例 2：ノーマルモードとソフトウェアスタンバイモード間の遷移

NORMAL_OPE ⇒ SSTBY 遷移、および SSTBY ⇒ NORMAL_OPE 復帰時のドライバ関数フローを図 12-4 に示します。製品移行時のプログラム変更箇所を赤文字で示します。

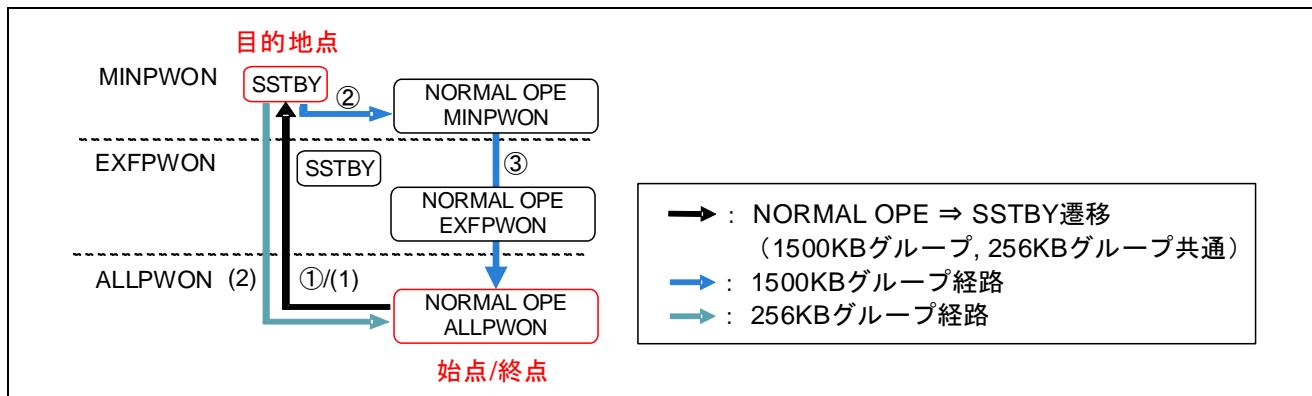


図 12-3 NORMAL OPE(ALLPWON) ⇒ SSTBY(MINPWON) ⇒ NORMAL OPE(ALLPWON)遷移図

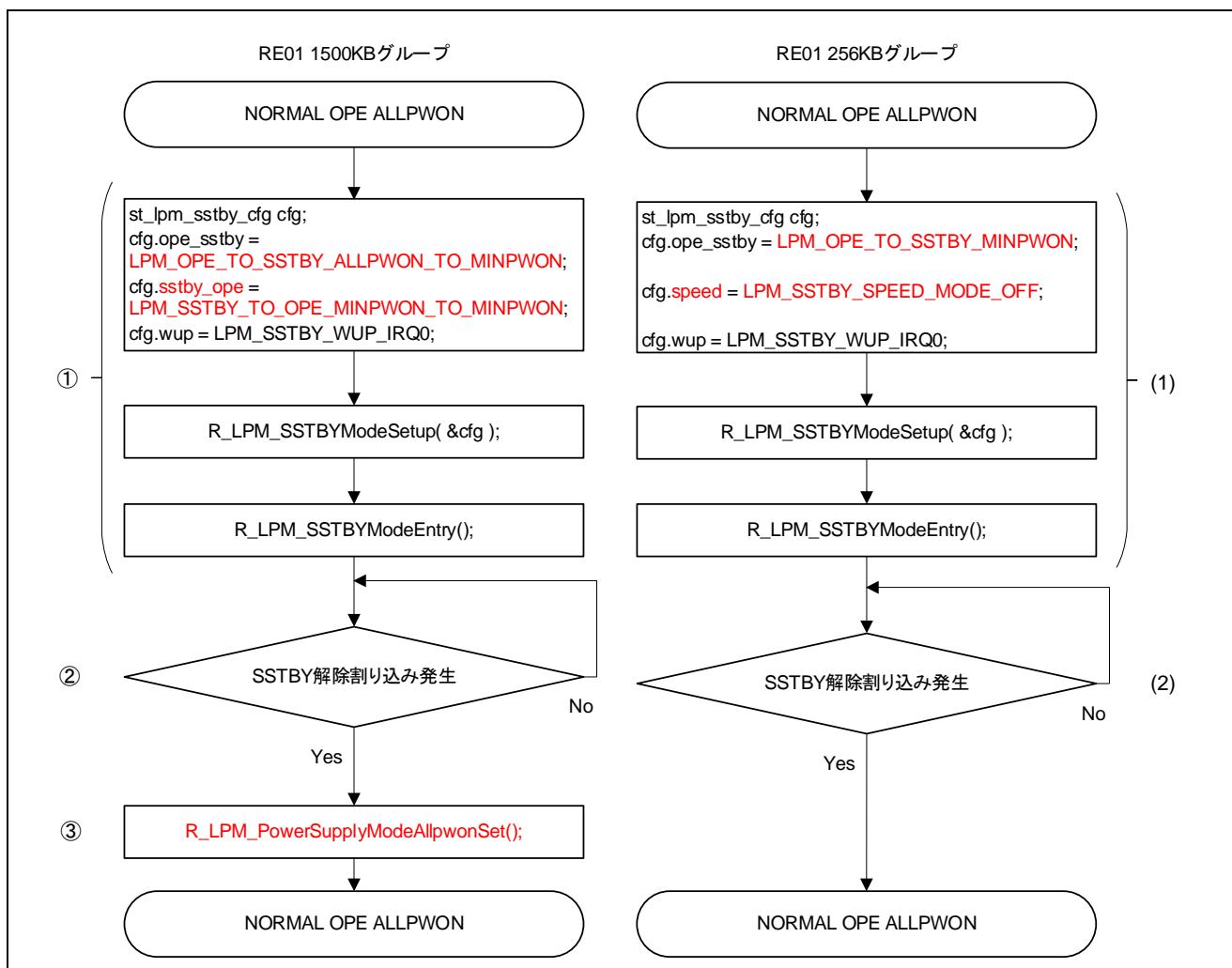


図 12-4 NORMAL OPE(ALLPWON) ⇒ SSTBY(MINPWON) ⇒ NORMAL OPE(ALLPWON)遷移のドライバ関数フロー

12.1.3 遷移例 3 : High-Speed モードと Low-Speed モード間の遷移

ALLPWON(High-Speed) ⇒ EXFPWON/MINPWON(Low-Speed)遷移、および
EXFPWON/MINPWON(Low-Speed) ⇒ ALLPWON(High-Speed)遷移時のドライバ関数フローを図 12-6 に示します。製品移行時のプログラム変更箇所を赤文字で示します。

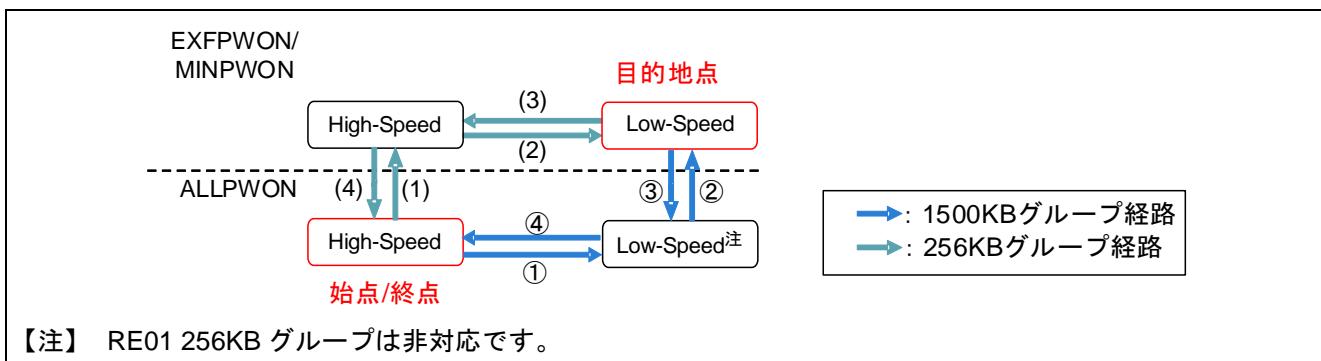


図 12-5 ALLPWON(High-Speed) ⇒ EXFPWON/MINPWON(Low-Speed) ⇒ ALLPWON(High-Speed)遷移図

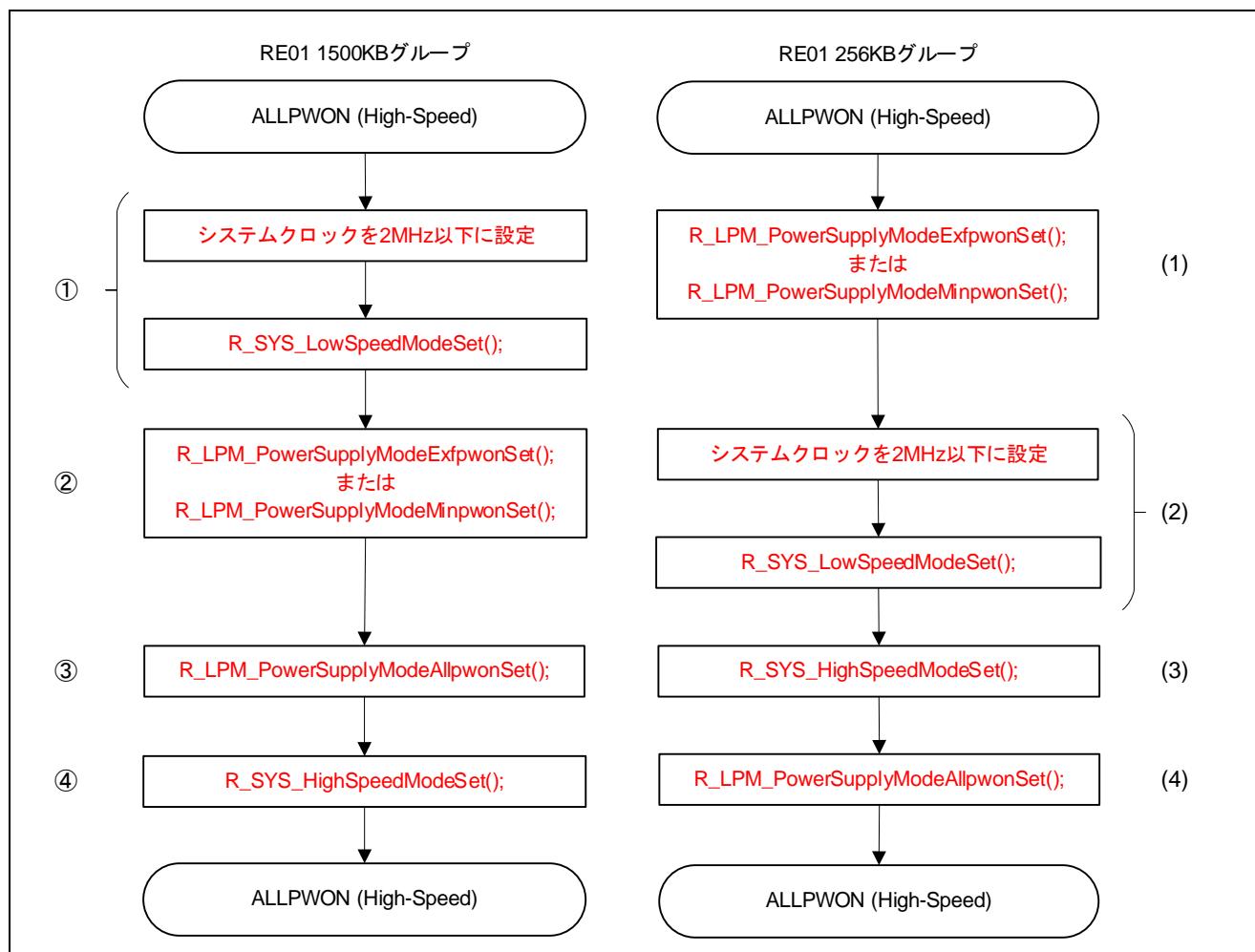


図 12-6 ALLPWON(High-Speed) ⇒ EXFPWON/MINPWON(Low-Speed) ⇒ ALLPWON(High-Speed)遷移のドライバ関数フロー

13. サンプルコード

サンプルコードは、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。

14. 参考ドキュメント

ユーザーズマニュアル：ハードウェア

R7F0E01 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。)

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

(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。)

J-Link/J-Trace User Guide : UM08001_JLink.pdf

<J-Link インストールフォルダ>/Doc/Manuals/ UM08001_JLink.pdf

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/>

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

Rev.	発行日	改訂内容	
		ページ	ポイント
0.10	2018/06/22	-	初版発行
0.70	2019/01/28	-	RE01 1.5MB WS2 に対応
0.71	2019/02/15	-	2.4.1 章の割り込みについて説明補足を追加
		-	2.4 章の説明を改善
0.72	2019/07/01	-	全章の構成を大幅見直し 1 章に CMSIS PACK の詳細を追加 5 章にユーザコードの組み込み方法を追加
1.00	2019/09/25	-	全編スタイルガイドに準じてスタイル適正化 用語統一（図内含む） 誤記修正（図内含む） SOTB などファミリ名、グループ名を修正：RE ファミリ、 RE01_1500KB フォルダ構成を最新版に更新 1.4.4 pin.c の関数とドライバの対応を追加 6.3 (a) RAM マッピング方法に EWARM でのリンクファイル のパス設定を追加 7. J-link 注意事項追加 9.6 トラブルシューティングの項目を追加 末尾「ご注意書き」を最新ファイルに差し替え
		-	全編にわたり章構成変更 「2 章 プロジェクトを動かしてみよう」追加
1.02a	2019/12/16	-	タイトルに RE01 256KB を追加
1.03	2020/6/19	-	RE01 256KB グループ情報追加 ドライバ名変更：R_SYS → R_SYSTEM 用語統一（図内含む） 誤記修正（図内含む）
		8, 27, 45	USB 機能の対応 ドライバ R_USB を追加 表 1-3, 表 3-3, 表 6-3
		16	Evaluation Kit RE01 1500KB ボード型名更新 表 1-4, 表 1-5
		45	6.5.1 (1) SCI チャネル修正, RTSn 端子追加
		98, 99	トラブルシューティングの項目を追加 11.3 A-2, 11.9
		100-103	「12 章 RE01 1500KB グループと RE01 256KB グループ間 でのプログラム移行時の注意事項」を追加

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. 静電気対策

CMOS 製品の取り扱いの際は静電気防止を心がけてください。CMOS 製品は強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレー やマガジンケース、導電性の緩衝材、金属ケースなどを利用し、組み立て工程にはアースを施してください。プラスチック板上に放置したり、端子を触ったりしないでください。また、CMOS 製品を実装したボードについても同様の扱いをしてください。

2. 電源投入時の処置

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. 電源オフ時における入力信号

当該製品の電源がオフ状態のときに、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源オフ時における入力信号」についての記載のある製品は、その内容を守ってください。

4. 未使用端子の処理

未使用端子は、「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。

5. クロックについて

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。

6. 入力端子の印加波形

入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がノイズなどに起因して、 V_{IL} (Max.) から V_{IH} (Min.) までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、 V_{IL} (Max.) から V_{IH} (Min.) までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。

7. リザーブアドレス（予約領域）のアクセス禁止

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、将来の拡張機能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。

8. 製品間の相違について

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、フラッシュメモリ、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、ノイズ幅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（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
 2. 当社製品、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またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
 3. 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、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に關し、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、データシート等において高信頼性、Harsh environment向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他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ティング処理等、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。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. オ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)



ルネサス エレクトロニクス株式会社

■営業お問合せ窓口

<http://www.renesas.com>

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

ルネサス エレクトロニクス株式会社 〒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/>