R20UT4672JJ0100
Rev.1.00
2019.11.26

RH850 ファミリ用 C コンパイラパッケージ(CC-RH)
ブート領域、フラッシュ領域の分割方法

要旨

RH850 ファミリ用 C コンパイラ CC-RH を使用して、プログラムをブート領域、フラッシュ領域に分割する際に、必要な処理について説明します。

動作確認バージョン

本資料は、次のツール、バージョンで説明をします。

- RH850 ファミリ用 C コンパイラ CC-RH V2.01.00
- 統合開発環境 CS+ for CC V8.02.00

目次

1. 概要

1.1 ブート領域、フラッシュ領域の分割について

ブート領域とフラッシュ領域の分割の目的は、ブート領域上のプログラムの再構築を行わず、フラッシュ領域上のプログラムのみを変更することです。

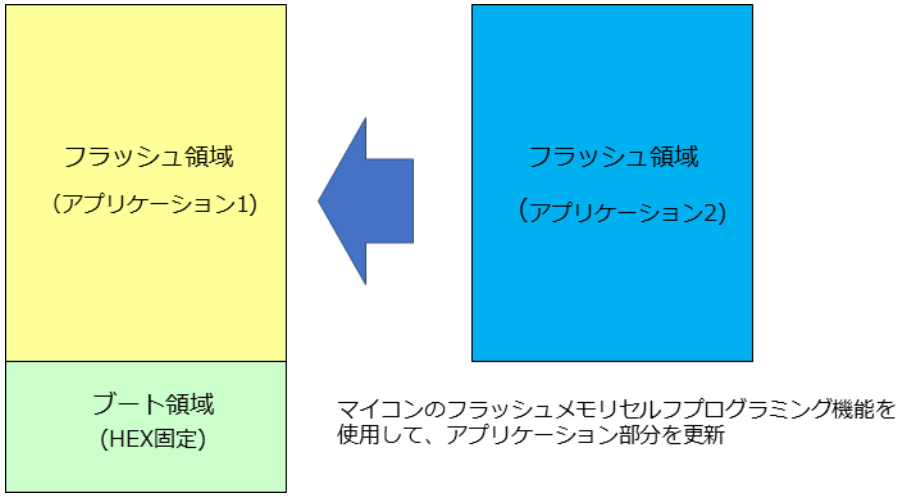

図 1 システム上のイメージ

※：本資料では、ブート領域はシステム設計上、書き換えを禁止する領域、フラッシュ領域はシステム設計上、上書き換え／取り換えが可能な領域とします。

ブート領域とフラッシュ領域の分割をするためには、ブート領域とフラッシュ領域の 2 つのプロジェクトを作成します。それぞれのプロジェクト間で下記を実現する必要があります。

- フラッシュ領域からブート領域の変数、関数をアクセスできるようにします。
  - ブート領域プロジェクトでリンカの-FSymbol オプションを利用して外部定義シンボルをファイルに出力します。
  - フラッシュ領域プロジェクトでそのファイルをビルド対象にします。

- ブート領域から、フラッシュ領域の関数を分岐テーブル経由で呼び出せるようにします。
  - ブート領域プロジェクトにて、フラッシュ領域の関数へ分岐する分岐テーブルのアドレスを呼び出すようにします。
  - フラッシュ領域プロジェクトにて、ブート領域プロジェクトから　呼び出される分岐テーブルを作成し、各関数への分岐命令を記述します。

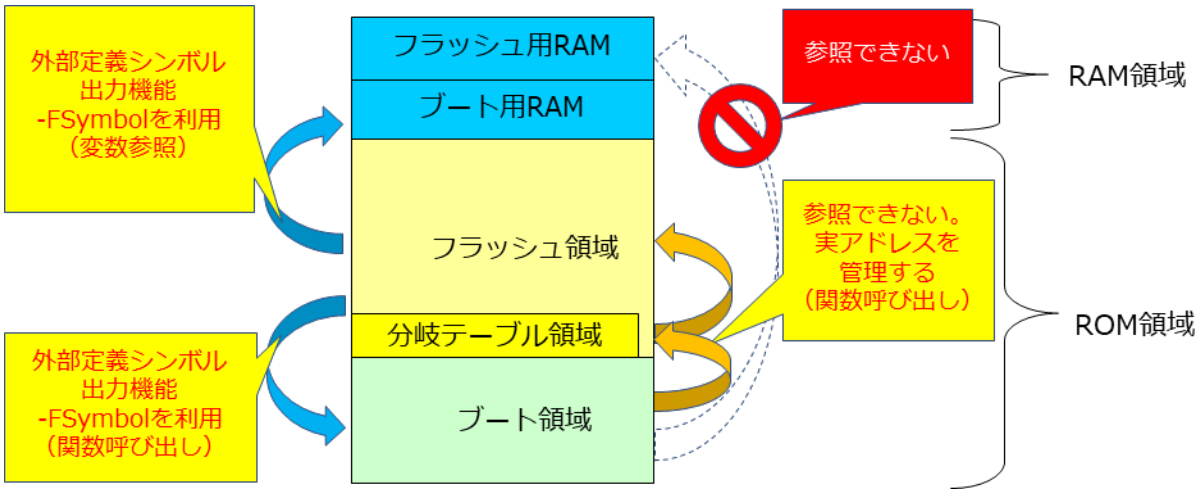

図 2 ブート領域、フラッシュ領域間の変数、関数の参照

1.2 ブート領域、フラッシュ領域の配置

次のようにブート領域とフラッシュ領域に配置します。

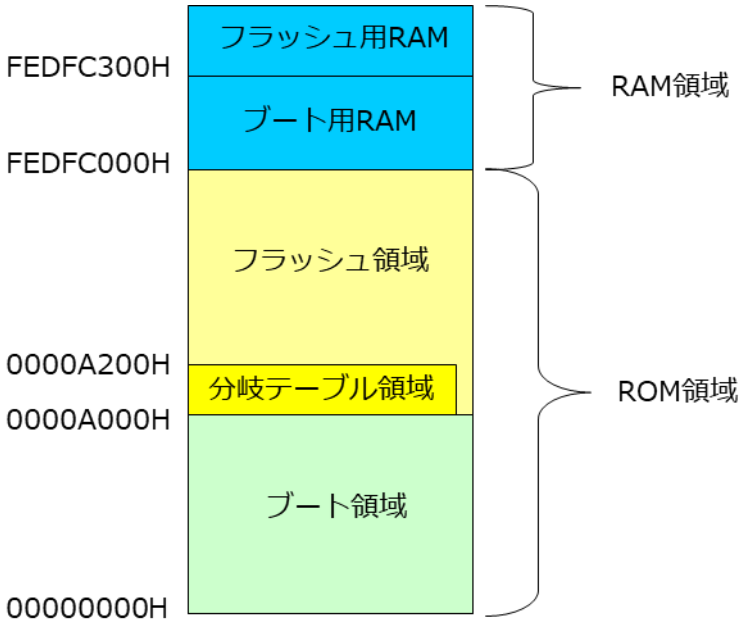

図 3 ブート領域とフラッシュ領域の配置例

1.3 ブート領域、フラッシュ領域のビルドイメージ

ブート領域、フラッシュ領域のビルドイメージを図 4 に示します。
フラッシュ領域のプロジェクトをビルドするためには、ブート領域のプロジェクトをビルドして生成される外部変数・関数のシンボル情報が必要です。そのため、ブート領域のプロジェクトをフラッシュ領域より先にビルドしてください。

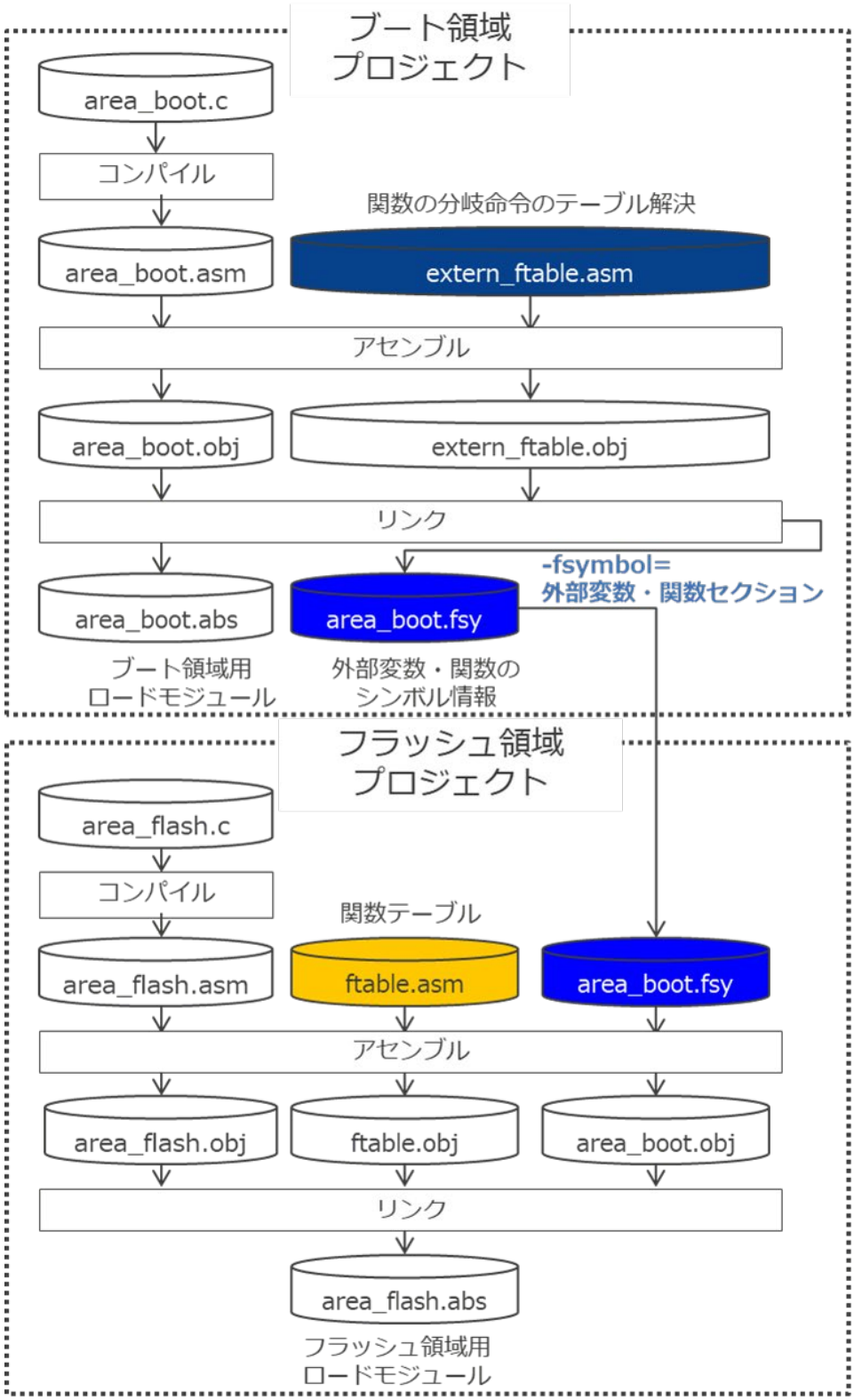

図 4 ブート領域、フラッシュ領域のビルドイメージ

2. ブート領域、フラッシュ領域共通

2.1 プロジェクトの作成

2.1.1 メインプロジェクト・サブプロジェクトの作成

CS+でメインプロジェクトとしてフラッシュ領域のプロジェクトを作成し、サブプロジェクトとしてブート領域のプロジェクトを作成します。【注】

【注】 CS+のビルド順は、「サブプロジェクト」→「メインプロジェクト」です。
ブート領域のプログラムは 1 度作成したら変更しないため、フラッシュ領域の 2 世代目以降の作成時には、サブプロジェクトを削除することが可能です。

2.1.2 自動生成ファイルの削除

(1) ブート領域のプロジェクトから以下ビルド対象ファイルを削除します。
- main.c

(2) フラッシュ領域のプロジェクトから以下ビルド対象ファイルを削除します。
- main.c
- boot.asm

2.1.3 ビルド対象ファイルの追加

(3) ブート領域のプロジェクトに以下ビルド対象ファイルを追加します。
- area_boot.c
- extern_ftable.asm
- ftable.inc 注1

(4) フラッシュ領域のプロジェクトに以下ビルド対象ファイルを追加します。
- area_flash.c
- ftable.asm
- ftable.inc
- area_boot.fsy 注2

注1. フラッシュ領域のプロジェクトの ftable.inc を共有します。
注2. ブート領域のプロジェクトをビルド後に追加します。

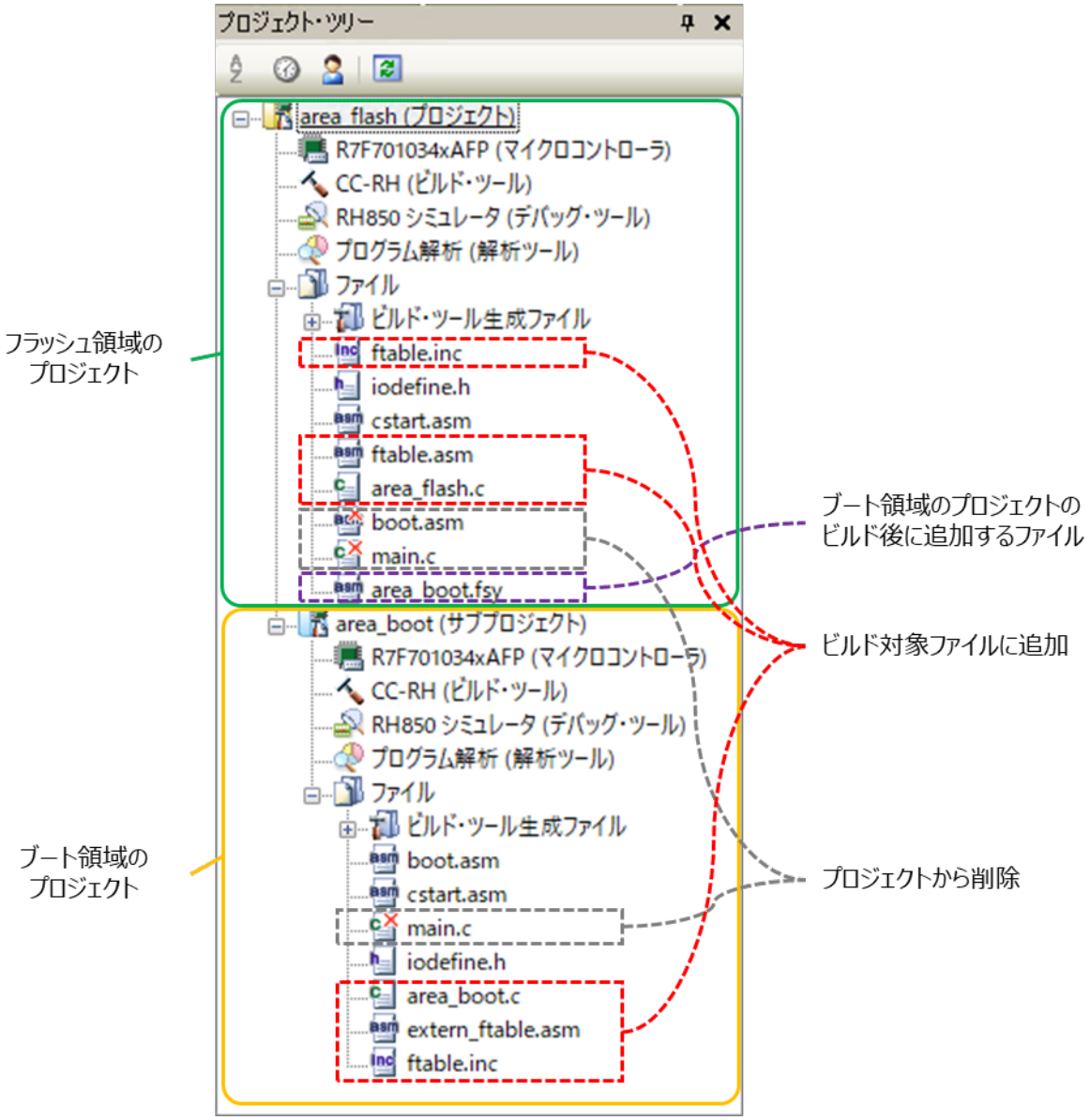

図 5 プロジェクト作成例

2.2 ブート領域、フラッシュ領域共通プログラムの作成

2.2.1 分岐テーブルのアドレス定義ファイル（アセンブラ）

- ブート領域、フラッシュ領域共通で参照する分岐テーブルアドレス定義のインクルードファイル ftable.inc を作成します。
  - FLASH_TABLE ： 分岐テーブルの先頭アドレス
  - INTERRUPT_OFFSET ： 分岐テーブルの割り込み領域分のサイズ

ftable.inc の作成例

```
FLASH_TABLE      .EQU  0xA000
INTERRUPT_OFFSET .EQU  0x100
```

2.3 ブート領域用とフラッシュ領域用のヘキサファイル

本資料で設定するファイル名は以下になります。（出力方法は後述）

- ブート領域とフラッシュ領域を結合したヘキサファイル：boot_flash.mot
- フラッシュ領域用ヘキサファイル：flashA000_ffff.mot
- ブート領域用ヘキサファイル：boot0000_9fff.mot

補足： ロードモジュールファイル（*.abs）は、ブート領域、フラッシュ領域のそれぞれに生成されます。

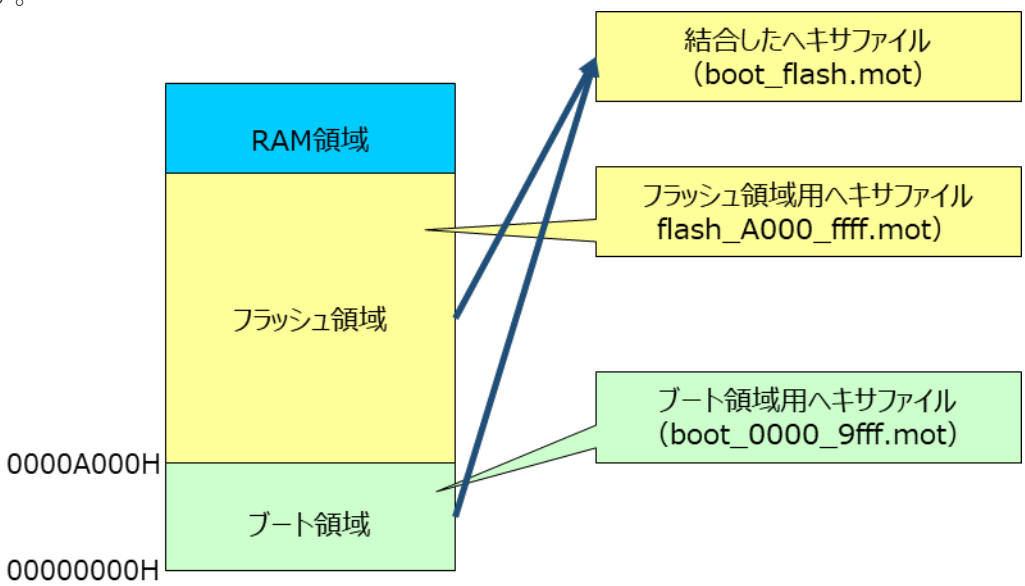

図 6 ブート領域用とフラッシュ領域用のヘキサファイル

2.4 初期化フロー

初期化フローを図 7 に示します。

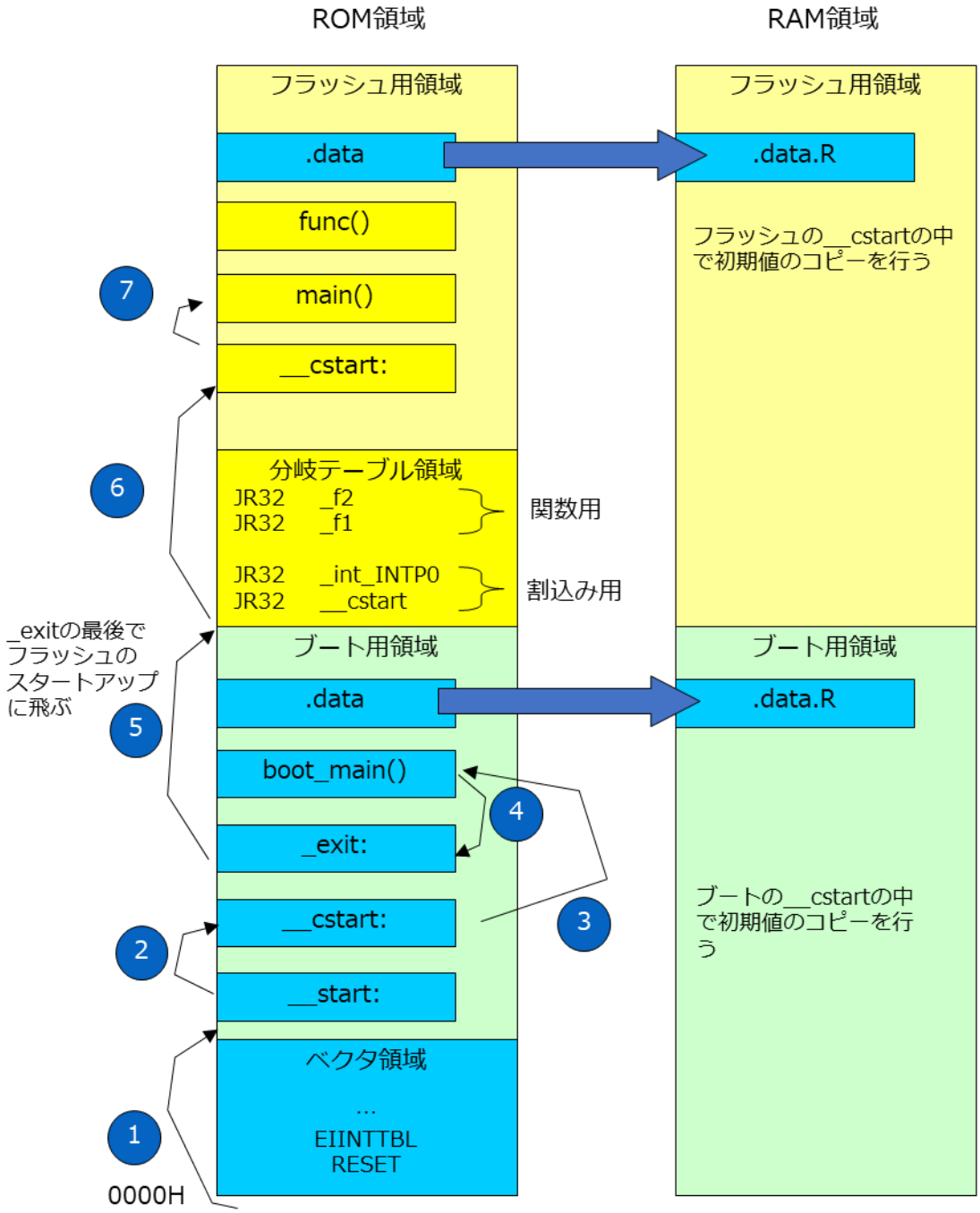

図 7 初期化フロー

3. ブート領域

3.1 ブート領域用プログラムの作成

ブート領域用に次のプログラムの変更・作成が必要です。

— スタートアップルーチンの変更
— 割り込み／例外ベクタのフォーマットの変更
— 分岐テーブルのアドレス解決用のファイル作成

3.1.1 スタートアップルーチン（cstart.asm）の変更

ここでは、スタートアップルーチン(cstart.asm)の変更手順を示します。

1. 分岐テーブルのアドレス定義のインクルードを追加します。

cstart.asm の変更例(1/5)

```
$INCLUDE "ftable.inc"

;-----------------------------------------------------------------------------
;       system stack
```

2. 外部定義シンボル出力機能-FSymbol の対象外にするために、セクション名を変更します。

cstart.asm の変更例(2/5)

```
;-----------------------------------------------------------------------------
;       startup
;-----------------------------------------------------------------------------
        .section    ".btext", text
        .public     __cstart
```

3. ブート側で FPU を使用しない場合は、FPU 設定部分をコメントアウトしてください。フラッシュ領域のスタートアップルーチンで FPU の初期設定を行います。

cstart.asm の変更例(3/5)

```
        ; enable FPU
$if 0 ; disable this block when not using FPU
        stsr   6, r10, 1    ; r10 <- PID
        shl    21, r10
        shr    30, r10
        bz     .L1                ; detecting FPU
        stsr   5, r10, 0          ; r10 <- PSW
        movhi  0x0001, r0, r11
        or     r11, r10
        ldsr   r10, 5, 0          ; enable FPU

        movhi  0x0002, r0, r11
        ldsr   r11, 6, 0          ; initialize FPSR
        ldsr   r0, 7, 0           ; initialize FPEPC
.L1:
$endif
```

4. 割り込み関数を使用する場合は、例外発生を許可してください。

cstart.asm の変更例(4/5)

```
      ; set various flags to PSW via FEPSW

      stsr  5, r10, 0          ; r10 <- PSW
      xori  0x0020, r10, r10   ; enable interrupt
      ;movhi      0x4000, r0, r11
      ;or   r11, r10           ; supervisor mode -> user mode
      ldsr  r10, 3, 0          ; FEPSW <- r10
```

5. ブート領域用のメイン関数の呼び出しに変更します。また、ブート領域用のメイン関数の実行後にフラッシュ領域のスタートアップルーチンへ分岐するよう、r31 レジスタ(lp)への設定を FLASH_TABLE に変更します。

cstart.asm の変更例(5/5)

```
      mov   FLASH_TABLE, lp          ; lp <- FLASH_TABLE
      mov   #_boot_main, r10
      ldsr  r10, 2, 0                ; FEPC <- #_boot_main
      ; apply PSW and PC to start user mode
      feret
_exit:
      br    _exit                    ;  end of program
```

3.1.2 割り込み／例外ベクタのフォーマット（boot.asm） の変更

ここでは、割り込み／例外ベクタのフォーマット(boot.asm)の変更手順を示します。

1. 分岐テーブルのアドレス定義のインクルードを追加します。

boot.asm の変更例(1/4)

```
$INCLUDE "ftable.inc"

      ; if using eiint as table reference method,
```

2. 割り込み方式をテーブル参照方式に変更します。

boot.asm の変更例(2/4)

```
      ; if using eiint as table reference method,
      ; enable next line's macro.

      USE_TABLE_REFERENCE_METHOD .set 1
```

3. フラッシュ領域およびブート領域の割り込み関数に分岐させるためのテーブル配置アドレスを設定します。以下はチャネル 0 "EIINT0"発生時にフラッシュ領域の割り込み関数 int_INTP0 を実行、チャネル 2 "EIINT2"発生時にブート領域の割り込み関数 int_INTP2 を実行する場合の記述例です。

boot.asm の変更例(3/4)

```
        .section "EIINTTBL", const
        .align      512
        .dw         FLASH_TABLE + 0x10      ; INT0
        .dw         #_Dummy_EI              ; INT1
        .dw         #_int_INTP2             ; INT2
        .rept       512 - 3
        .dw         #_Dummy_EI              ; INTn
        .endm
```

4. 外部定義シンボル出力機能-FSymbol の対象外にするために、セクション名を変更します。

boot.asm の変更例(4/4)

```
;-------------------------------------------------------------------------------
;     startup
;-------------------------------------------------------------------------------
        .section    ".btext", text
        .align      2
__start:
```

3.1.3 分岐テーブルのアドレス解決用のファイル（extern_ftable.asm）の作成

- 分岐テーブルのアドレス解決用のファイルの作成（アセンブラ）
  - ブート領域からフラッシュ領域の関数を呼び出す際に用いるフラッシュ領域の分岐テーブルのアドレスを解決するために、シンボルを定義してください。
  - このファイルを、プロジェクトに登録してください。

extern_ftable.asm の作成例

```
$INCLUDE "ftable.inc"
        .public     _f1
_f1     .equ  (FLASH_TABLE + INTERRUPT_OFFSET + (0 * 0x10))
        .public     _f2
_f2     .equ  (FLASH_TABLE + INTERRUPT_OFFSET + (1 * 0x10))
```

3.2 ブート領域のオプションの設定

ブート領域用に次のオプション設定が必要です。

— 外部定義シンボルのファイルの出力
— セクションの配置指定
— ブート領域のアドレスのみのヘキサファイルを出力する設定

3.2.1 外部定義シンボルのファイルの出力

フラッシュ領域のプロジェクトでブート領域の変数、関数をアクセスできるように外部定義シンボルのファイルを出力します。
対象とするセクションを-FSymbol オプションにすべて登録してください。

例
[CC-RH（ビルド・ツール）]→[リンク・オプション]タブ
→[セクション]→[外部定義シンボルをファイル出力するセクション]

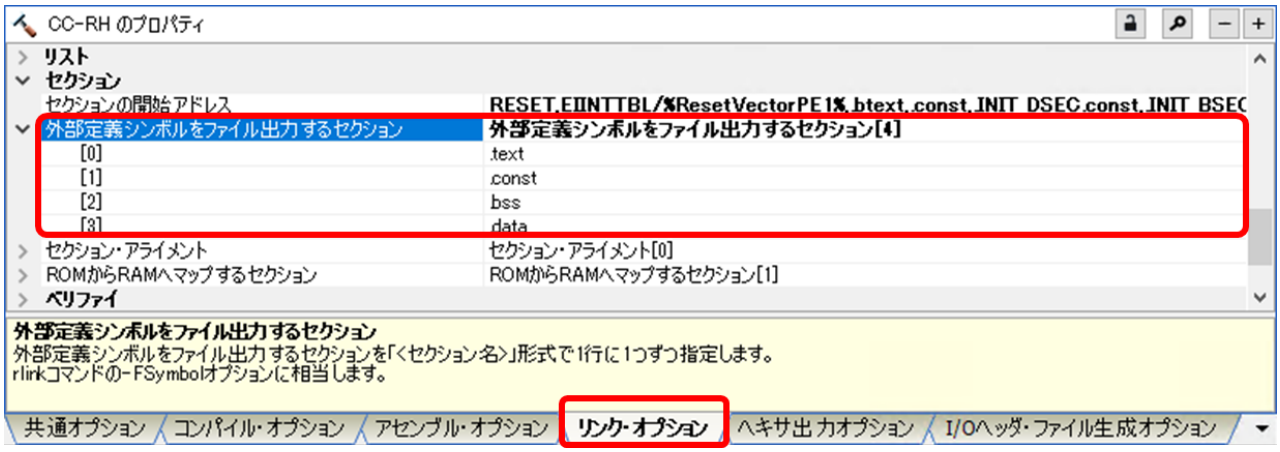

図 8 設定例

3.2.2 セクションの配置指定

リンカの-STARt オプションで、ブート領域のセクションの配置を指定してください。フラッシュ領域の配置と重ならない様に考慮してください。
スタック領域のセクションの指定も追加してください。

例
[CC-RH（ビルド・ツール）]→[リンク・オプション]タブ
→[セクション]→[セクションの開始アドレス]

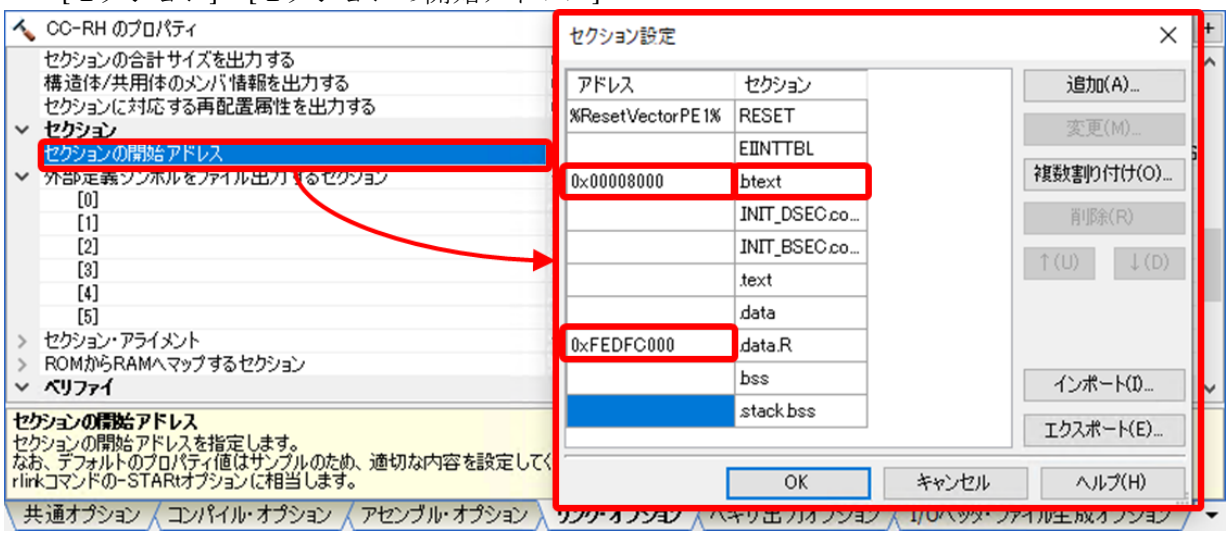

図 9 設定例

3.2.3 ブート領域のアドレスのみのヘキサファイルを出力する設定

出力するファイル名と、出力するアドレスを設定してください。

例
[CC-RH（ビルド・ツール）]→[ヘキサ出力オプション]タブ
→[出力ファイル]→[分割出力ファイル]に出力するファイル名と出力するアドレスを設定

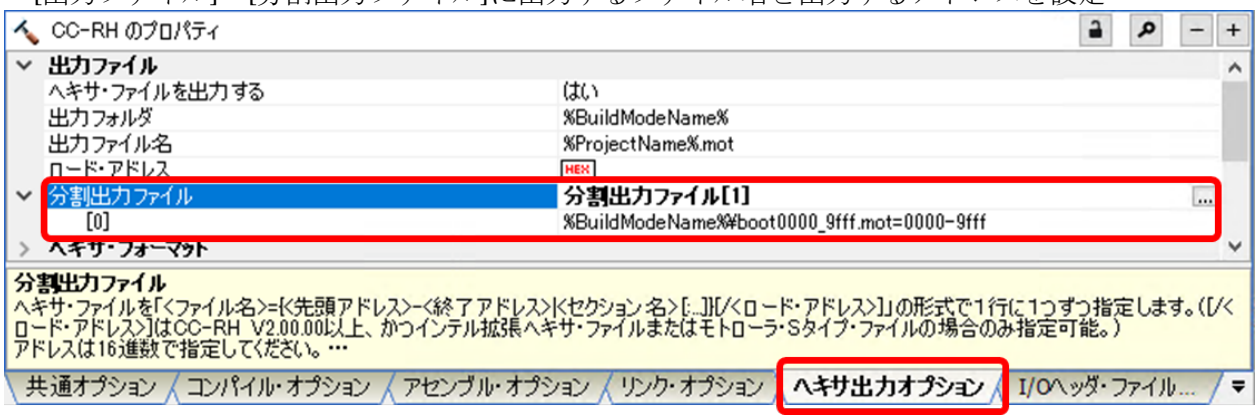

図 10 設定例

4. フラッシュ領域

4.1 フラッシュ領域用プログラムの作成

フラッシュ領域用に次のプログラムの変更・作成が必要です。

- スタートアップルーチンの変更
- 分岐テーブルの作成
- 割り込み関数の定義

4.1.1 スタートアップルーチン（cstart.asm）の変更

1. ベース・レジスタの設定部分をコメントアウトします。ブート領域のスタートアップルーチンで設定したベース・レジスタを使用するため、フラッシュ領域では再設定しないでください。【注】

   【注】 フラッシュ側で GP,EP レジスタを変更すると、ブート側での参照アドレスも変更されます。GP,EP レジスタはそれぞれについてフラッシュ側で使用するかブート側で使用するかを、プログラム全体で統一することを推奨します。

cstart.asm の変更例(1/2)

```
__cstart:
;       mov    #_stacktop, sp          ;  set sp register
;       mov    #__gp_data, gp          ;  set gp register
;       mov    #__ep_data, gp          ;  set ep register
```

2. フラッシュ領域のメイン関数へ分岐命令を追加してください。また、例外処理を定義する場合は、ユーザ・モードの設定をしてください。

cstart.asm の変更例(2/2)

```
        ; set various flags to PSW via FEPSW

        stsr   5, r10, 0          ; r10 <- PSW
        ;xori 0x0020, r10, r10   ; enable interrupt
        movhi 0x4000, r0, r11
        or     r11, r10           ; supervisor mode -> user mode
        ldsr   r10, 3, 0          ; FEPSW <- r10
        ;mov   #_exit, lp         ; lp <- #_exit
        ;mov   #_main, r10
        ;ldsr r10, 2, 0           ; FEPC <- #_main

        ; apply PSW and PC to start user mode
        ;feret
        jarl   _main, lp

_exit:
        br     _exit              ;  end of program
```

4.1.2 分岐テーブル（ftable.asm）の作成

ブート領域から呼び出されるアドレスに、ブート領域のプロジェクトに登録した extern_ftable.asm でシンボルを定義したフラッシュ領域の関数のアドレスへ分岐するための、分岐命令を記載してください。

例 ftable.asm

```
$INCLUDE "ftable.inc"

      .EXTERN     __cstart
      .EXTERN     _f1
      .EXTERN     _f2

.jtext      .CSEG text
      .ORG  FLASH_TABLE
      jr32        __cstart    ; RESET
      .align 16
      jr32        _int_INTP0  ; INTP0
.jtext2     .CSEG text
      .ORG  FLASH_TABLE+INTERRUPT_OFFSET
      jr32        _f1
      .align 16
      jr32        _f2
```

割り込み用

関数用

4.1.3 割り込み関数の定義

例 area_flash.c（抜粋）

```
#pragma interrupt int_INTP0(channel=0)

～省略～

volatile char f;

～省略～

void int_INTP0(void)
{
      f = 1;
}
```

4.2 フラッシュ領域のオプションの設定

フラッシュ領域用に次のオプション設定が必要です。

- 外部定義シンボルファイルのプロジェクト登録
- セクションの配置指定
- フラッシュ領域のアドレスのみのヘキサファイルを出力する設定
- ブート領域とフラッシュ領域のヘキサファイルの結合

4.2.1 外部定義シンボルファイルのプロジェクト登録

ブート領域の変数、関数をアクセスできるように、ブート領域で作成した外部定義シンボルファイルをプロジェクトに登録してください。

例

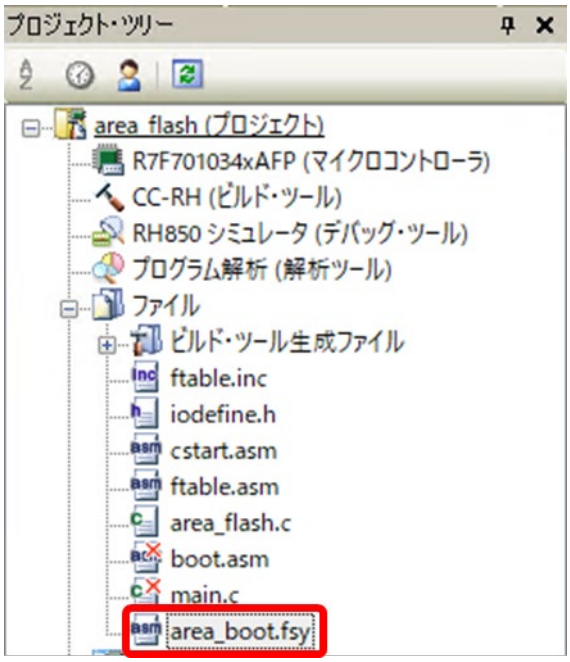

図 11 設定例

4.2.2 セクションの配置指定

リンカの-STARt オプションで、フラッシュ領域のセクションの配置を指定してください。

- ブート領域の配置と重ならない様に考慮してください。
- 分岐テーブルの領域を空けるようにしてください。
- RESET/EINTTBL のセクションは不要です。

例
[CC-RH（ビルド・ツール）]→[リンク・オプション]タブ
→[セクション]→[セクションの開始アドレス]

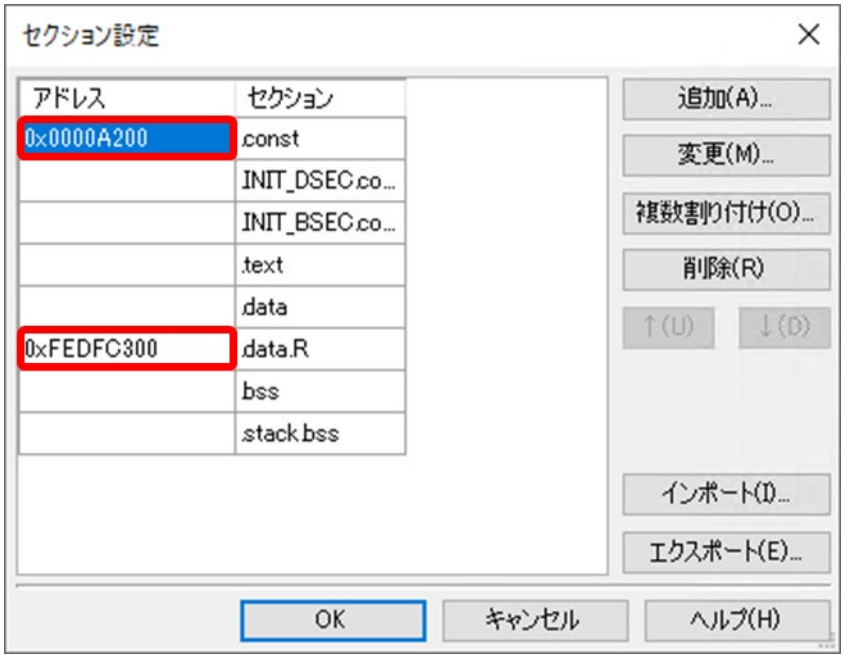

図 12 設定例

4.2.3 フラッシュ領域のアドレスのみのヘキサファイルを出力する設定

出力するファイル名と、出力するアドレスを設定してください。

例
[CC-RH（ビルド・ツール）]→[ヘキサ出力オプション]タブ
→[出力ファイル]→[分割出力ファイル]に出力するファイル名と出力するアドレスを設定

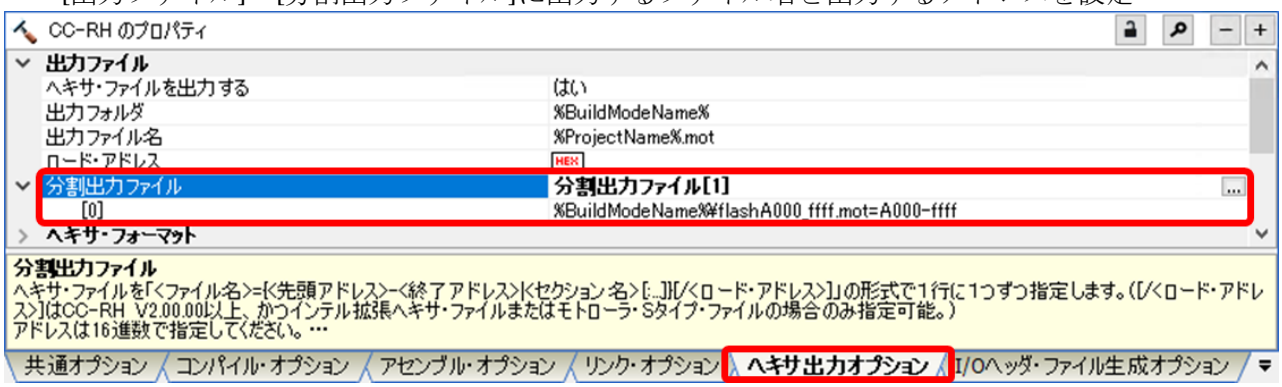

図 13 設定例

4.2.4 ブート領域とフラッシュ領域のヘキサファイルの結合

ブート領域とフラッシュ領域のヘキサファイルを結合して 1 つにする場合には、ビルド後の処理にリンカの実行を追加してください。

例
[CC-RH（ビルド・ツール）]→[共通オプション]タブ
→[その他]→[ビルド後に実行するコマンド]
にリンカの実行
"%MicomToolPath%¥CC-RH¥V2.01.00¥bin¥rlink.exe" -subcommand=sub_mot.txt
を追加

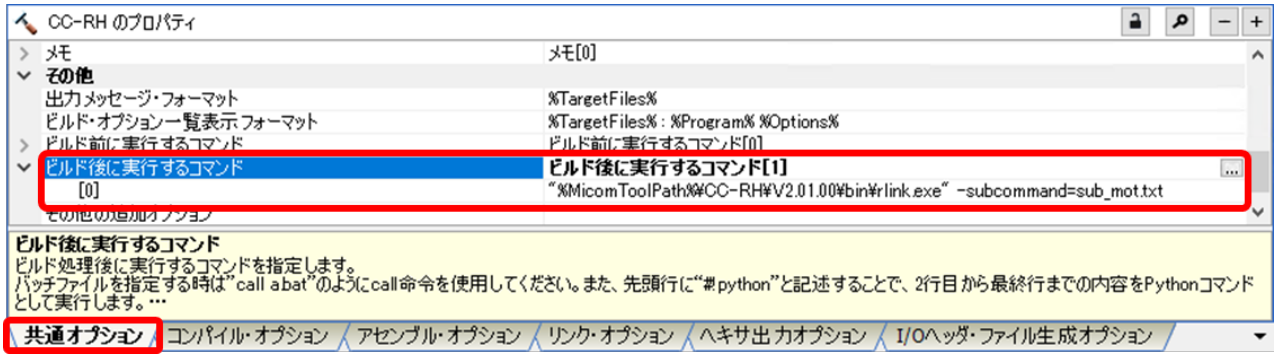

図 14 設定例

リンカに入力するサブコマンドファイルで、入力するヘキサファイルと、ヘキサファイルの形式、出力するファイル名を指定してください。

sub_mot.txt の作成例

```
-input=.¥area_boot¥DefaultBuild¥boot0000_9fff.mot
-input=.¥DefaultBuild¥flashA000_ffff.mot
-form=stype
-output=.¥DefaultBuild¥boot_flash.mot
```

5. デバッグツール

5.1 デバッグツールへのダウンロード

ロードモジュールファイル（*.abs）は、ブート領域用、フラッシュ領域用の 2 つのファイルが生成されるので、デバッグツールでは、2 つのロードモジュールファイルをダウンロードしてください。

例
[RH850 シミュレータ（デバッグ・ツール）]→[ダウンロード・ファイル設定]タブ
→[ダウンロード]→[ダウンロードするファイル]
にフラッシュ領域のプロジェクトにブート領域のロードモジュールファイルを追加

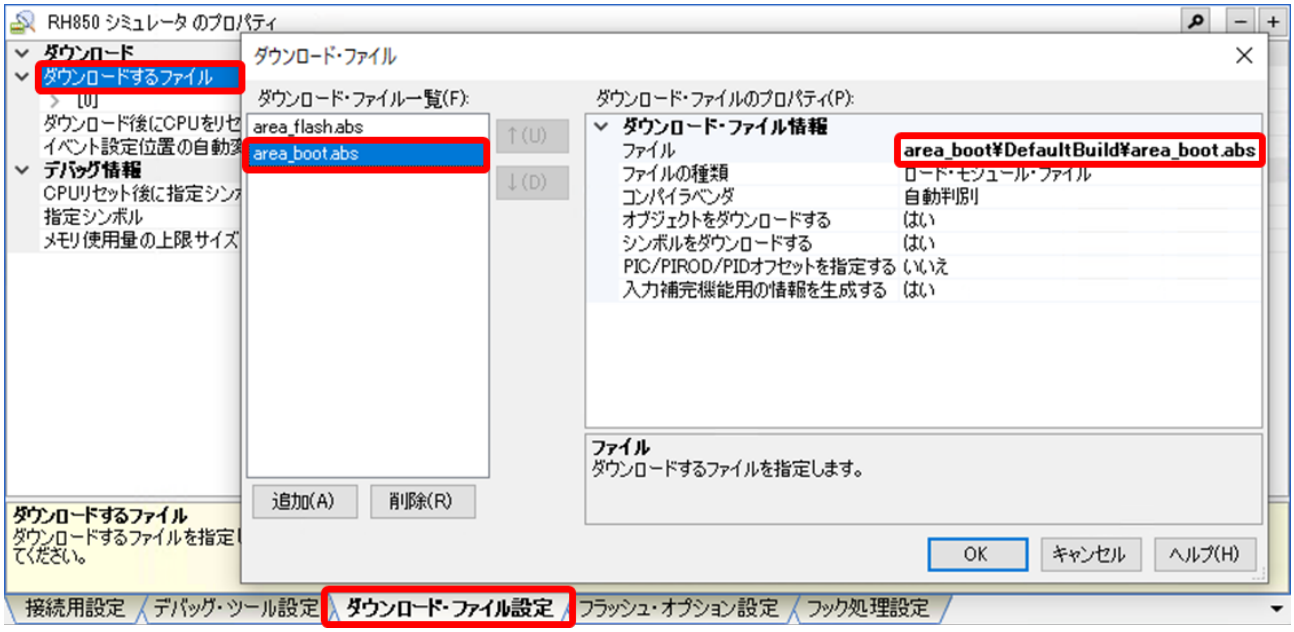

図 15 設定例

6. サンプルプログラム

今まで作成してきたプログラムを使用するための、ブート領域のプログラム、フラッシュ領域のプログラムの例を記載します。

6.1 ブート領域のサンプルプログラム（area_boot.c）

```
#include "iodefine.h"

#pragma interrupt int_INTP2 (channel=2)    /* ブート領域の割り込み定義 */

int boot_a = 0x12;
int boot_b = 0x34;
extern int f1(int);      /* フラッシュ領域の関数のプロトタイプ宣言 */
extern int f2(int);      /* フラッシュ領域の関数のプロトタイプ宣言 */
void boot_main(void)     /* ブート領域のメイン関数 */
{
      /*ブート領域のメインの処理 */
}
void boot_func(void)
{
  boot_a = f1(boot_a);   /* フラッシュ領域の関数呼び出し*/
  boot_b = f2(boot_b);   /* フラッシュ領域の関数呼び出し*/
}
void int_INTP2(void)     /*ブート領域の割り込み処理 */
{
  boot_a = 1;
}
```

6.2 フラッシュ領域のサンプルプログラム（area_flash.c）

```
#include "iodefine.h"
#pragma interrupt int_INTP0(channel=0)
volatile char f;
int flash_a, b;
extern int boot_a, boot_b;     /* ブート領域で定義された変数 */
extern void boot_func(void);   /* ブート領域で定義された関数 */
int f1(int a)
{
      return (++a);
}
int f2(int b)
{
      return (--b);
}
void main(void)    /* フラッシュ領域のメイン関数 */
{
      boot_a++;    /*  ブート領域の変数のアクセス*/
      boot_b++;    /*  ブート領域の変数のアクセス*/
      boot_func();/*  ブート領域の関数のアクセス*/
}
void int_INTP0(void)
{
      f = 1;
}
```

ホームページとサポート窓口

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

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

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2019/11/26	-	初版

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関し、当社は、一切その責任を負いません。
5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。

   標準水準：　コンピュータ、OA機器、通信機器、計測機器、AV機器、
   家電、工作機械、パーソナル機器、産業用ロボット等

   高品質水準：　輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、
   金融端末基幹システム、各種安全制御装置等

   当社製品は、データシート等により高信頼性、Harsh environment向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
6. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質および信頼性の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ジング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
8. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制するRoHS指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
9. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
10. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
12. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

注1.　本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社が直接的、間接的に支配する会社をいいます。

注2.　本資料において使用されている「当社製品」とは、注1において定義された当社の開発、製造製品をいいます。

(Rev.4.0-1 2017.11)

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/

Colophon 6.0