

## RL78 ファミリ用 C コンパイラパッケージ(CC-RL)

### ブート領域、フラッシュ領域の分割方法

#### 要旨

RL78 ファミリ用 C コンパイラ CC-RL 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。

- RL78 ファミリ用 C コンパイラ CC-RL V1.08.00
- 統合開発環境 e<sup>2</sup> studio V7.3.0
- 統合開発環境 CS+ for CC V8.01.00

#### 目次

|   |    |
|---|----|
| 1. 概要 .....   | 3  |
| 1.1 ブート領域、フラッシュ領域の分割について .....                          | 3  |
| 1.2 ブート領域、フラッシュ領域の配置 .....                              | 4  |
| 1.3 ブート領域、フラッシュ領域の分割をするために .....                        | 4  |
| 1.4 ブート領域、フラッシュ領域のビルドイメージ .....                         | 5  |
| 2. ブート領域、フラッシュ領域共通 .....                                | 6  |
| 2.1 プロジェクトの作成 .....                                     | 6  |
| 2.1.1 e <sup>2</sup> studio .....                       | 6  |
| 2.1.2 CS+ .....   | 8  |
| 2.2 ブート領域、フラッシュ領域共通プログラムの作成 .....                       | 10 |
| 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) .....                   | 10 |
| 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 .....                       | 10 |
| 2.4 初期化フロー .....  | 11 |
| 3. ブート領域 .....  | 12 |
| 3.1 ブート領域用プログラムの作成 .....                                | 12 |
| 3.1.1 スタートアップルーチン (cstart.asm) の変更 .....                | 12 |
| 3.1.2 hdwinit.asm と stkinit.asm の変更 .....               | 14 |
| 3.1.3 オンチップ・デバッグ用領域の確保プログラム (ocdrom_ce.asm) の作成 .....   | 14 |
| 3.1.4 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 ..... | 14 |
| 3.2 ブート領域のオプションの設定 .....                                | 14 |
| 3.2.1 外部定義シンボルのファイルの出力 .....                            | 14 |
| 3.2.2 セクションの配置指定 .....                                  | 16 |
| 3.2.3 フラッシュ領域の割り込み関数に分岐させるためのベクタの設定 .....               | 17 |
| 3.2.4 オンチップ・デバッグを使用するための設定 .....                        | 18 |
| 3.2.5 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 .....                 | 19 |

|       |                               |    |
|-------|-------------------------------|----|
| 4.    | フラッシュ領域                       | 21 |
| 4.1   | フラッシュ領域用プログラムの作成              | 21 |
| 4.1.1 | スタートアップルーチン (cstart.asm) の変更  | 21 |
| 4.1.2 | 分岐テーブル (ftable.asm) の作成       | 21 |
| 4.1.3 | 割り込み関数の定義                     | 22 |
| 4.2   | フラッシュ領域のオプションの設定              | 22 |
| 4.2.1 | 外部定義シンボルファイルのプロジェクト登録         | 22 |
| 4.2.2 | セクションの配置指定                    | 24 |
| 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 | 25 |
| 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      | 26 |
| 5.    | デバッグツール                       | 28 |
| 5.1   | デバッグツールへのダウンロード               | 28 |
| 6.    | サンプルプログラム                     | 30 |
| 6.1   | ブート領域のサンプルプログラム (boot.c)      | 30 |
| 6.2   | フラッシュ領域のサンプルプログラム (flash.c)   | 31 |

## 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クの-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。

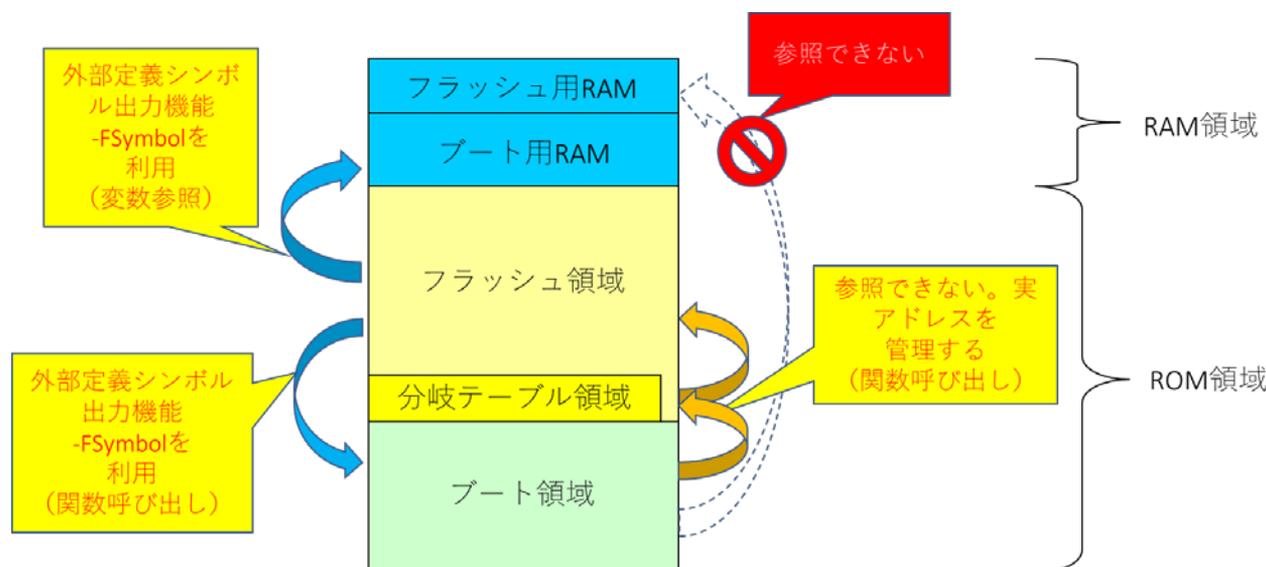


図 2 ブート領域、フラッシュ領域間の変数、関数の参照

## 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。

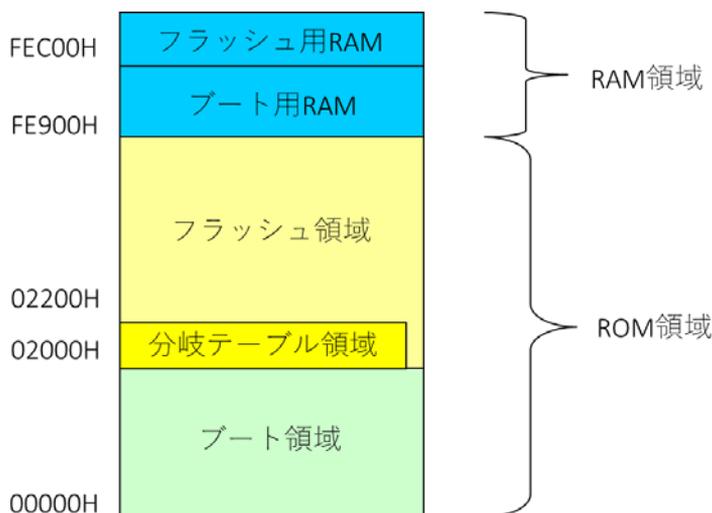


図 3 ブート領域とフラッシュ領域の配置例

## 1.3 ブート領域、フラッシュ領域の分割をするために

ブート領域とフラッシュ領域の分割を実現するためには、下記の手順でプロジェクトを作成・設定します。

1. ブート領域用のプロジェクトの作成
  - A. ブート領域のためのプログラムを作成
  - B. リンカのオプション設定
  - C. フラッシュ領域のプロジェクトをビルドする際に必要なため、フラッシュ領域用プロジェクトよりも前にビルドするよう設定
2. フラッシュ領域用のプロジェクトの作成
  - A. フラッシュ領域のためのプログラムを作成
  - B. リンカのオプション設定

### 1.4 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。

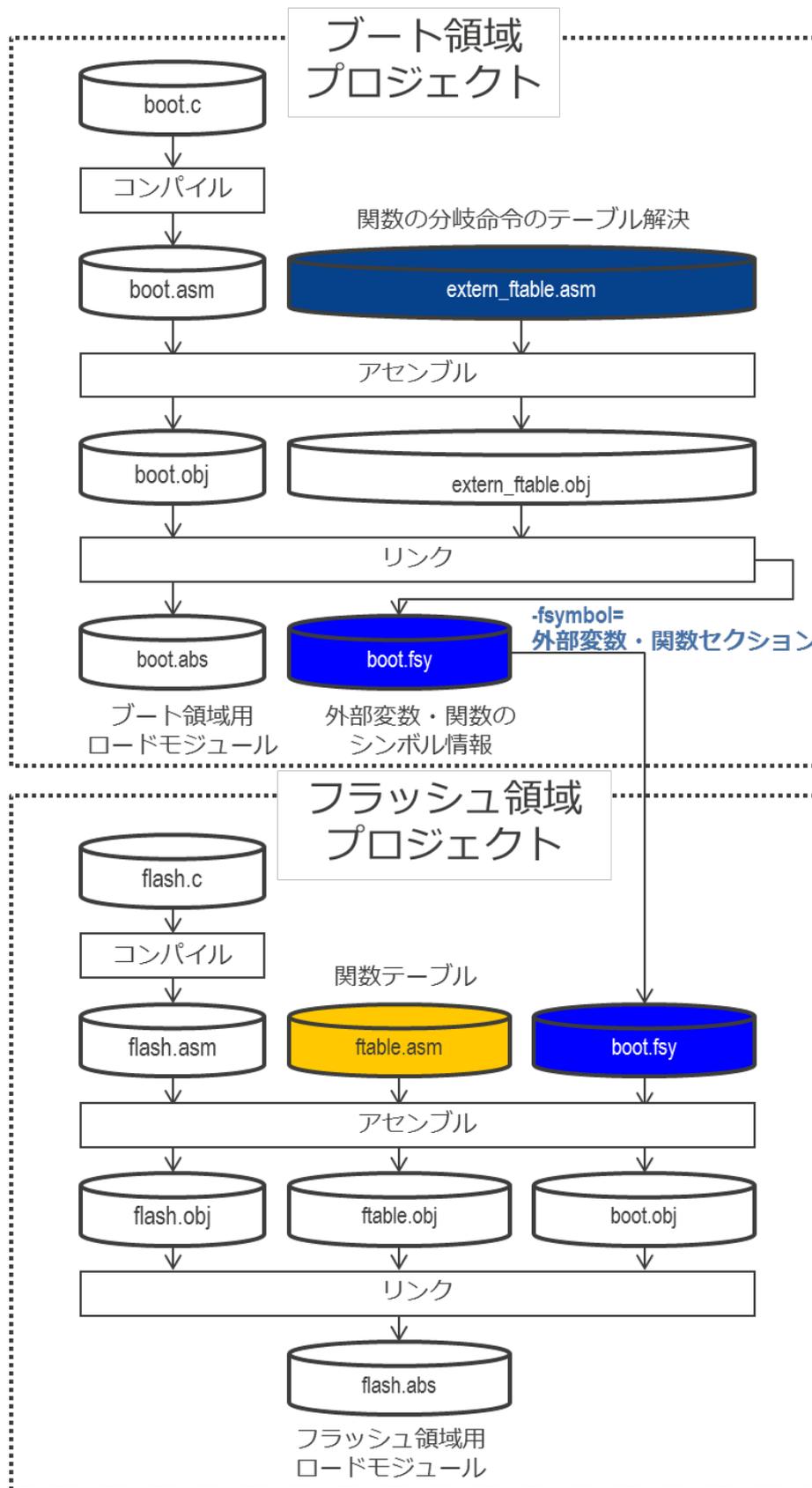


図 4 ブート領域、フラッシュ領域のビルドイメージ

## 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 e<sup>2</sup> studio

##### (1) プロジェクトの作成

フラッシュ領域のプロジェクトとブート領域のプロジェクトを「1.3 ブート領域、フラッシュ領域の分割をするために」を参考にし、それぞれ作成します。

フラッシュ領域のプロジェクトで、ブート領域のプロジェクトのプロジェクトを参照するように設定します。フラッシュ領域のプロジェクトビルド時に、ブート領域のプロジェクトが先にビルドされます。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参照を外すことが可能です。

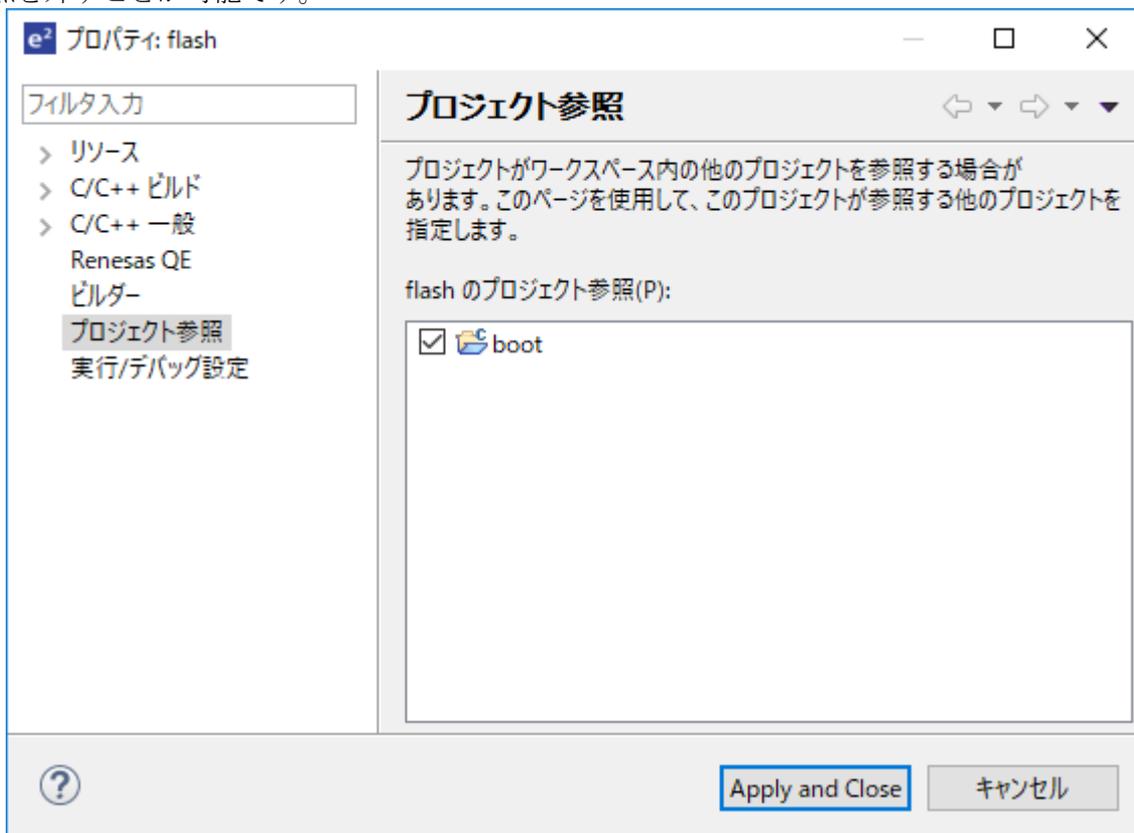


図 5 フラッシュ領域のプロジェクトからブート領域のプロジェクトを参照

##### (2) ビルド対象ファイルの追加

###### A. 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。

- extern\_ftable.asm
- ftable.inc
- ocdmon\_ce.asm

###### B. 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。

- boot.fsy(ブート領域のビルド後に生成される)
- ftable.asm
- ftable.inc
- int.c
- sub\_mot.txt

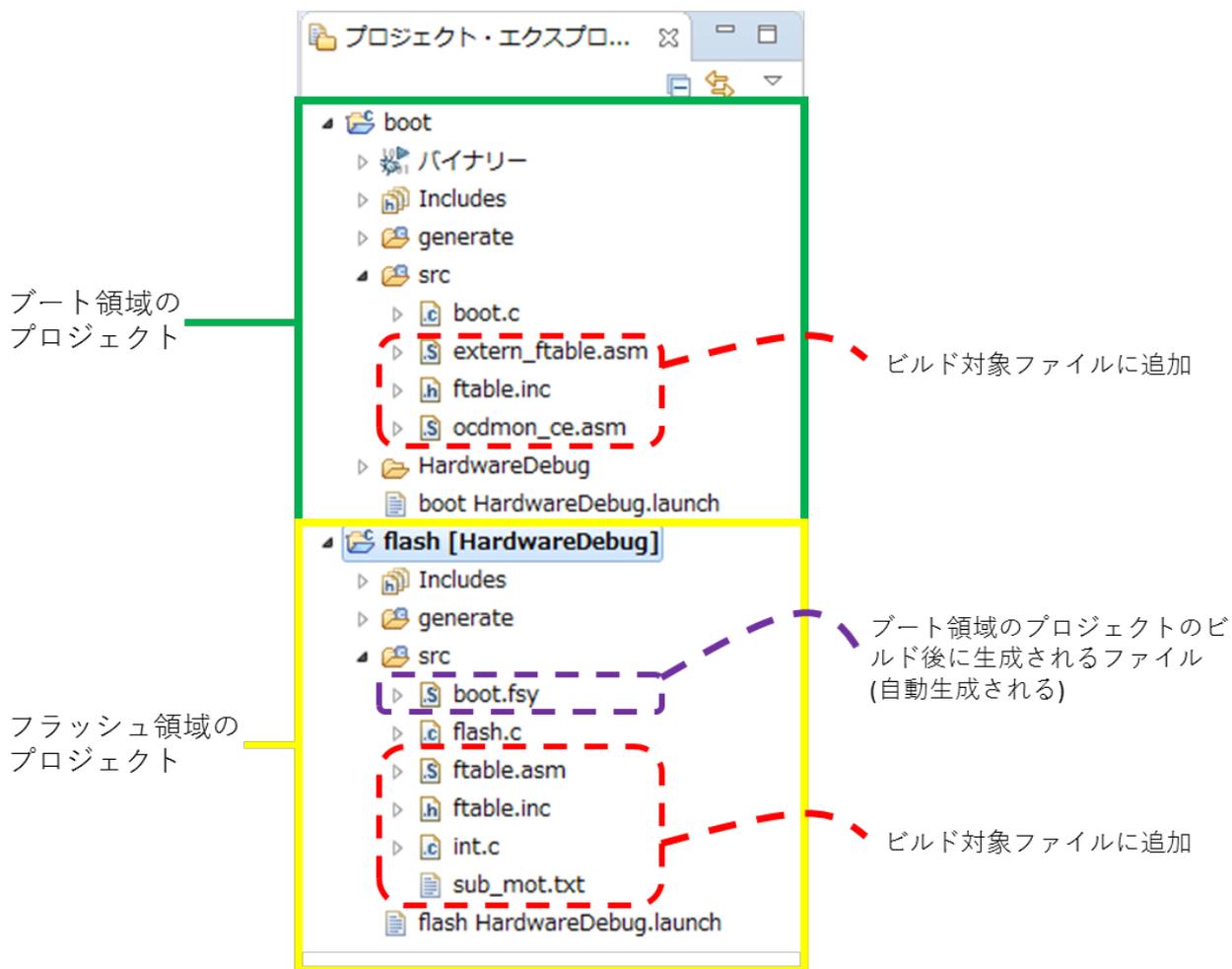


図 6 e2 studio でのプロジェクト作成例

### 2.1.2 CS+

#### (1) プロジェクトの作成

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) 自動生成ファイルの削除

ブート領域のプロジェクトおよびフラッシュ領域のプロジェクトから以下のファイルを削除します。

- main.c

#### (3) ビルド対象ファイルの追加

A. 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。

- boot.c
- extern\_ftable.asm
- ftable.inc
- ocdmon\_ce.asm

B. 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。

- flash.c
- ftable.asm
- ftable.inc
- int.c
- boot.fsy

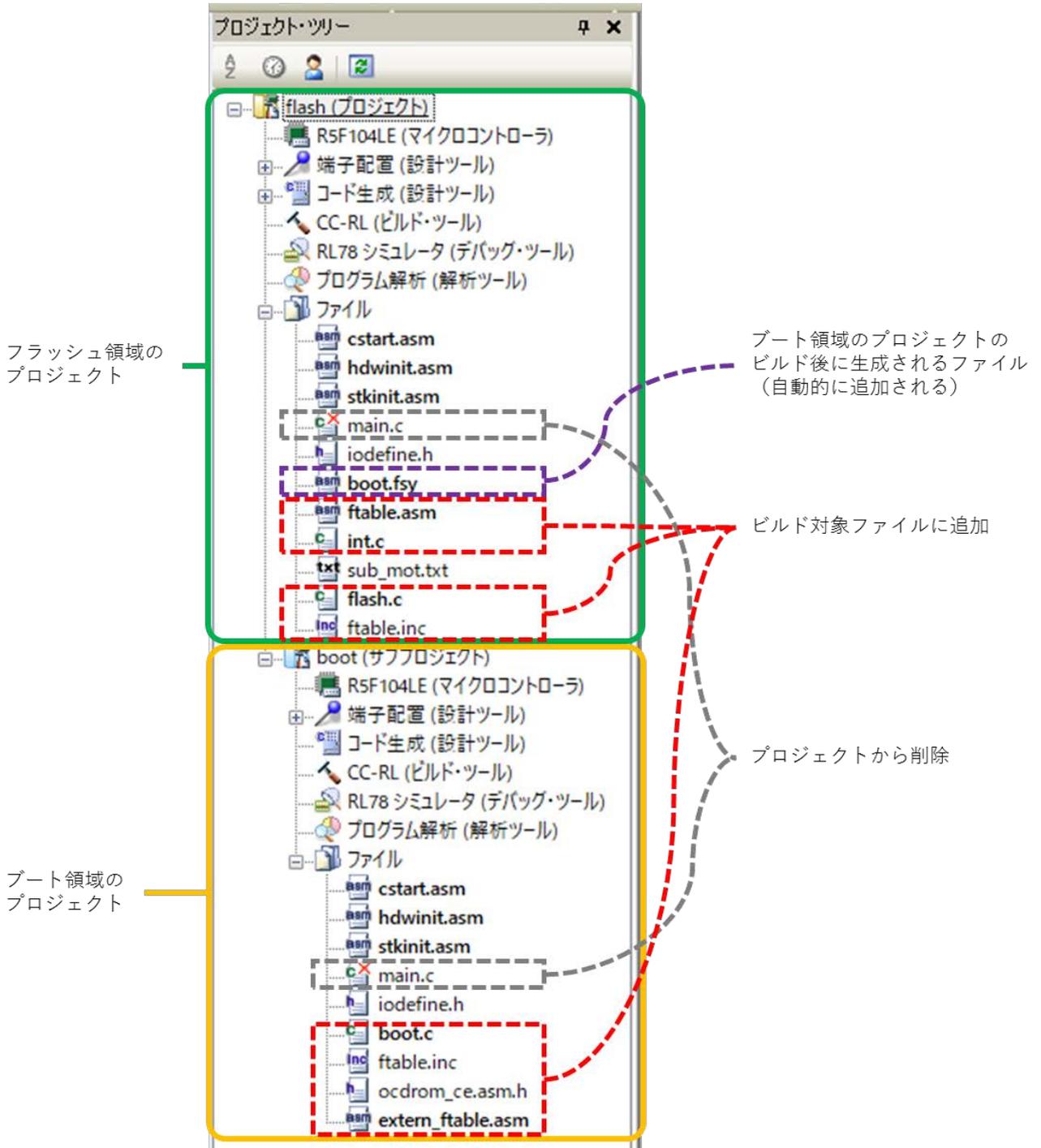


図 7 CS+でのプロジェクト作成例

## 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  0x2000
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 : `flash2000_ffff.mot`
- ブート領域用ヘキサファイル : `boot0000_1fff.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。

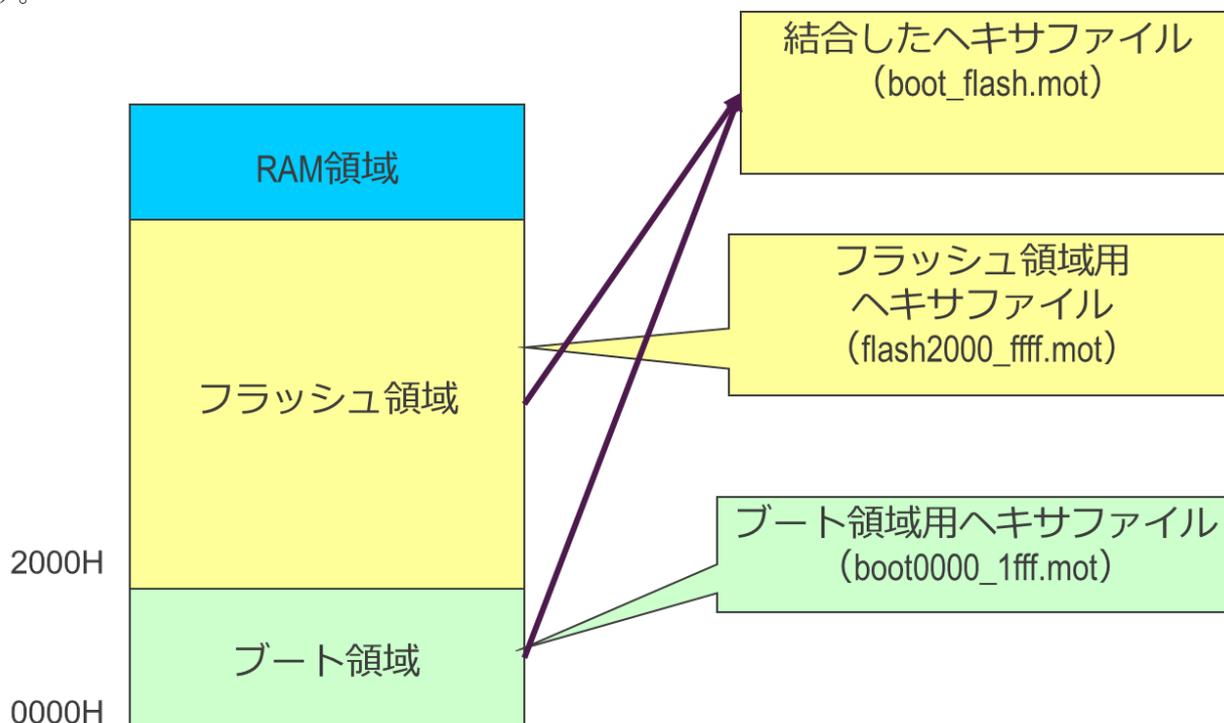


図 8 ブート領域用とフラッシュ領域用のヘキサファイル

## 2.4 初期化フロー

初期化フローを図 9 に示します。

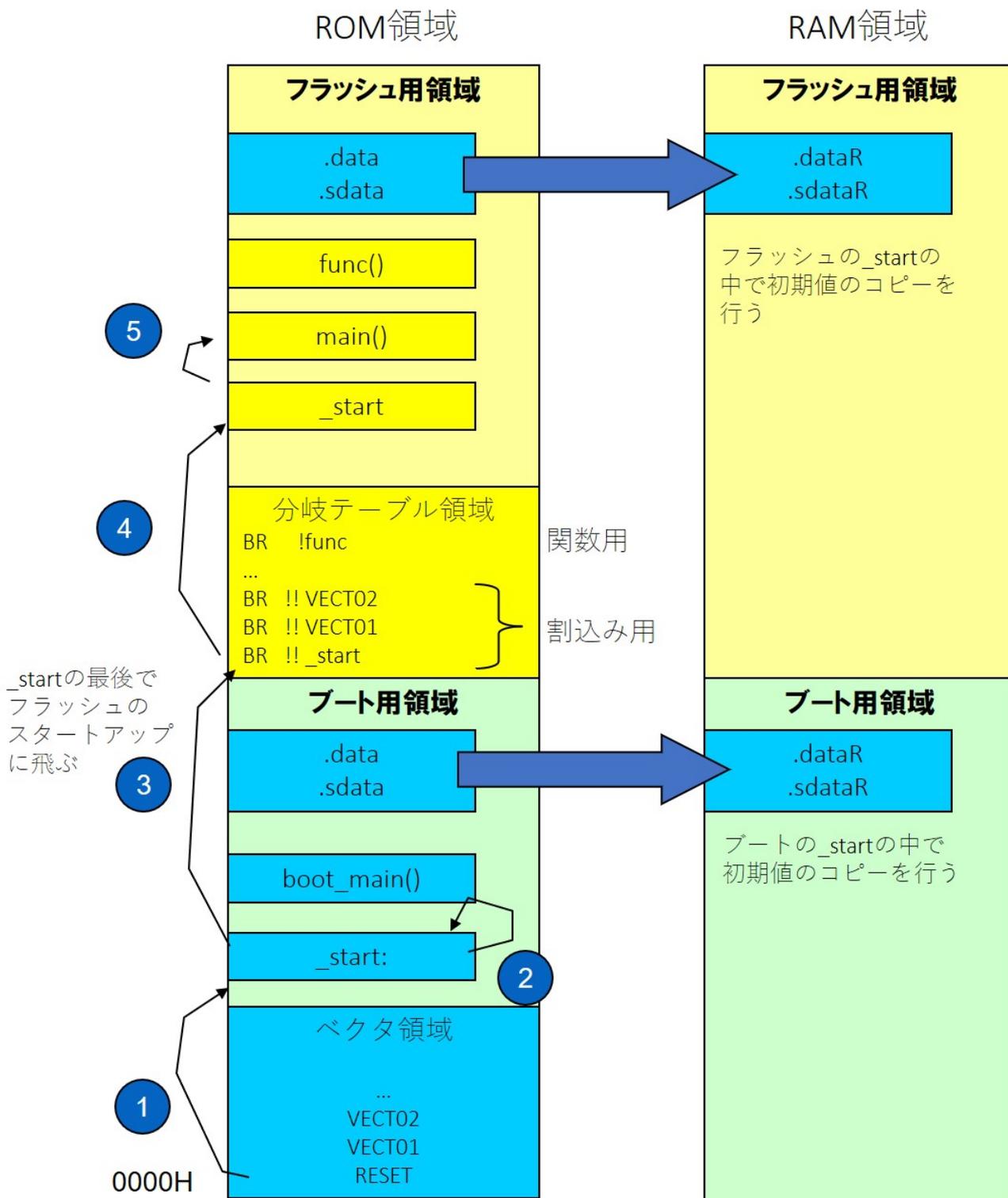


図 9 初期化フロー

### 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
- hdwinit.asm と stkinit.asm の変更
- オンチップ・デバッグ用領域の確保プログラムの作成
- 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(1/6)

```
$IFDEF __RENESAS_VERSION__
__RENESAS_VERSION__ .EQU 0x01000000
$ENDIF

$INCLUDE "ftable.inc"
```

2. 条件アセンブル命令をコメントアウトし、.stack\_bss セクションの定義を有効にします。

cstart.asm の変更例(2/6)

```
;$IF (__RENESAS_VERSION__ < 0x01010000); for CC-RL V1.00
;-----
; stack area
;-----
; !!! [CAUTION]!!!
; Set up stack size suitable for a project.
.SECTION .stack_bss, BSS
_stackend:
.DS 0x200
_stacktop:
;$ENDIF
```

3. 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(3/6)

```
;-----
; startup
;-----
.SECTION .btext, TEXT
_start:
```

4. 条件アセンブル命令等をコメントアウトし、明示的に確保した.stack\_bss セクションをスタックポインタに設定します。

## cstart.asm の変更例(4/6)

```

;-----
; setting the stack pointer
;-----
;$IF (__RENESAS_VERSION__ >= 0x01010000)
; MOVW    SP,#LOWW(__STACK_ADDR_START)
;$ELSE   ; for CC-RL V1.00
; MOVW    SP,#LOWW(_stacktop)
;$ENDIF

```

5. ブート領域用のメイン関数の呼び出しへの変更と、フラッシュ領域のスタートアップルーチンへの分岐命令を追加します。

## cstart.asm の変更例(5/6)

```

;-----
; call main function
;-----
CALL    !!_boot_main    ; main();
BR      !!FLASH_TABLE

```

6. ブート領域にミラー元領域が含まれない場合には、.const セクションの定義をコメントアウトします。

## cstart.asm の変更例(6/6)

```

; section
;-----
$IF (__RENESAS_VERSION__ >= 0x01010000)
.SECTION .RLIB, TEXTF
.L_section_RLIB:
.SECTION .SLIB, TEXTF
.L_section_SLIB:
$ENDIF
.SECTION .textf, TEXTF
.L_section_textf:
;.SECTION .const, CONST
;.L_section_const:

```

### 3.1.2 hdwinit.asm と stkinit.asm の変更

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。

hdwinit.asm と stkinit.asm の変更例

```
.btextf .CSEG TEXTF
```

### 3.1.3 オンチップ・デバッグ用領域の確保プログラム (ocdrom\_ce.asm) の作成

オンチップ・デバッグを使用する場合には、特定のメモリを開ける (0xff で埋める) 必要があります。リンカの -debug\_monitor オプションを使用すると、-device オプション指定のデバイスファイルの情報から、0x0002~0x0003 番地、0x00ce~0x00d7 番地、ROM の最後の 512 バイト分の領域を 0xff としたロードモジュールファイルを生成します。

ROM の最後の 512 バイト分の領域は、フラッシュ領域に含まれるため、ブート領域の設定を行うには、次のようにしてください。

- 0x0002~0x0003 番地
  - リンカの -VECTN オプションで 0xffff を設定してください。
- 0x00ce~0x00d7 番地
  - アセンブラソースで定義してください。(下記プログラム参照)

ocdrom\_ce.asm の作成例

```
MON_CE .CSEG AT 0x00ce
.DB8 0xffffffffffffffff
.DB2 0xffff
```

- ROM の最後の 512 バイト
  - フラッシュ領域で確保してください。

### 3.1.4 分岐テーブルのアドレス解決用のファイル (extern\_fable.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\_fable.asm の作成例

```
$INCLUDE "fable.inc"
.public _f1
_f1 .equ (FLASH_TABLE + INTERRUPT_OFFSET + (0 * 4))
.public _f2
_f2 .equ (FLASH_TABLE + INTERRUPT_OFFSET + (1 * 4))
```

## 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
- フラッシュ領域の割り込み関数に分岐させるためのベクタの設定
- オンチップ・デバッグを使用するための設定
- 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 e<sup>2</sup> studio

[プロパティ]→[C/C++ ビルド]→[設定]→[ツール設定]タブ

→[Linker]→[出力]→[外部定義シンボルをファイルに出力するセクション]

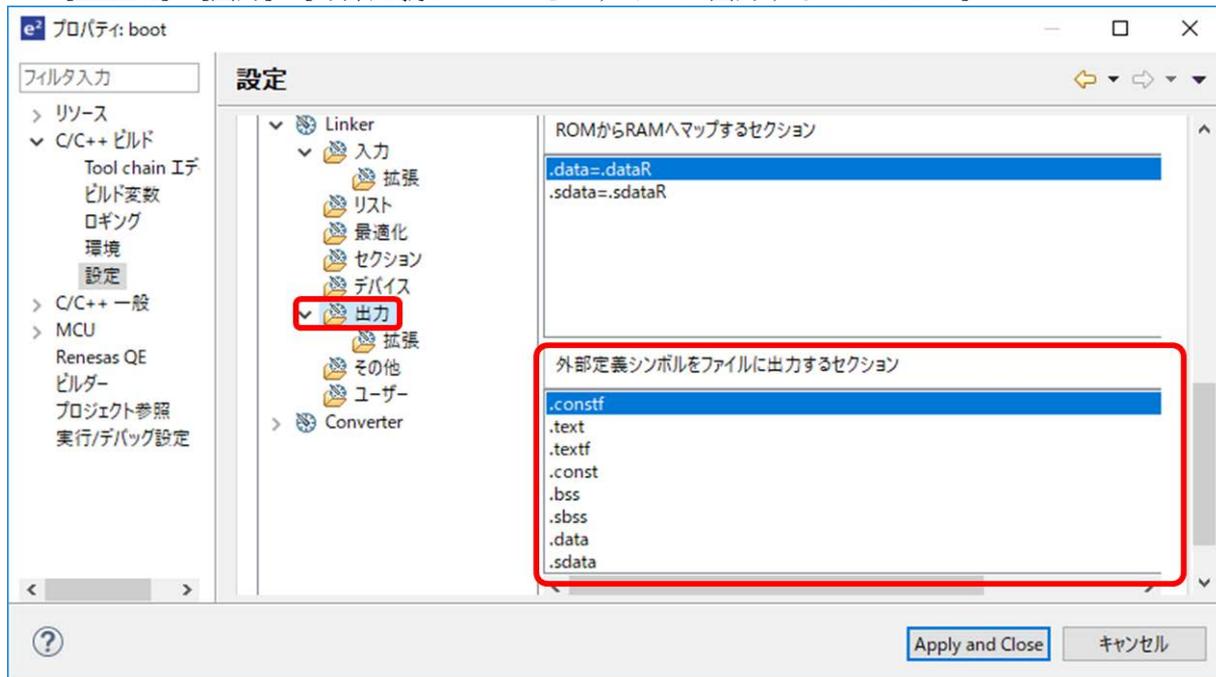


図 10 e<sup>2</sup> studio での設定例

例 CS+

[CC-RL (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に出力するセクション]

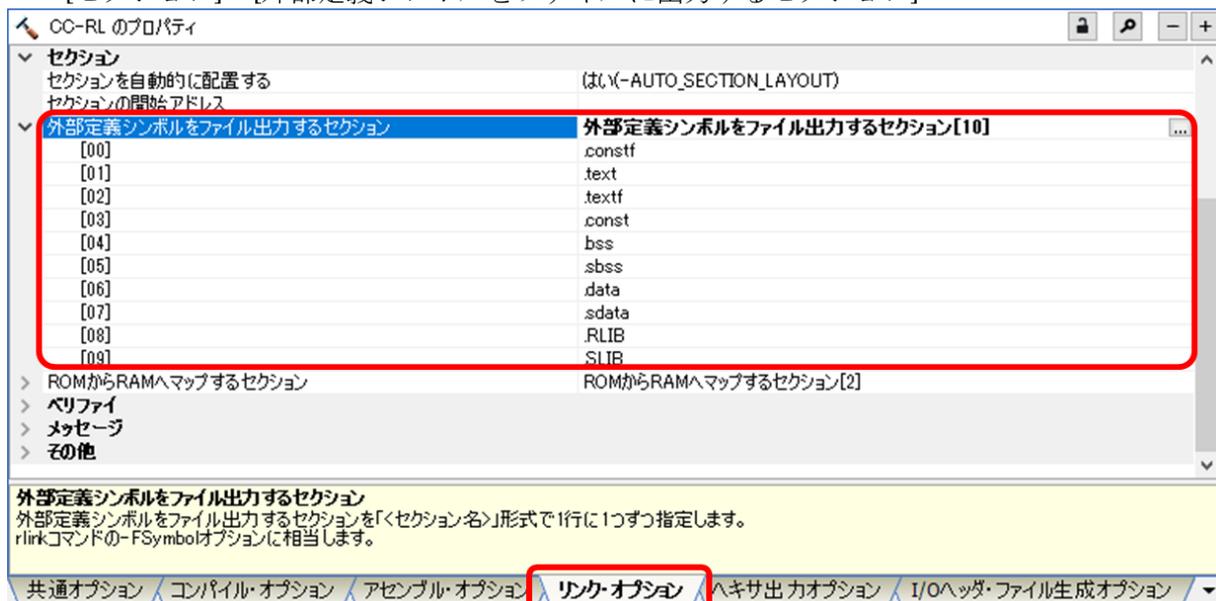


図 11 CS+での設定例

### 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 e<sup>2</sup> studio

[プロパティ]→[C/C++ ビルド]→[設定]→[ツール設定]タブ  
→[Linker]→[セクション]→[セクション・ビューアー]

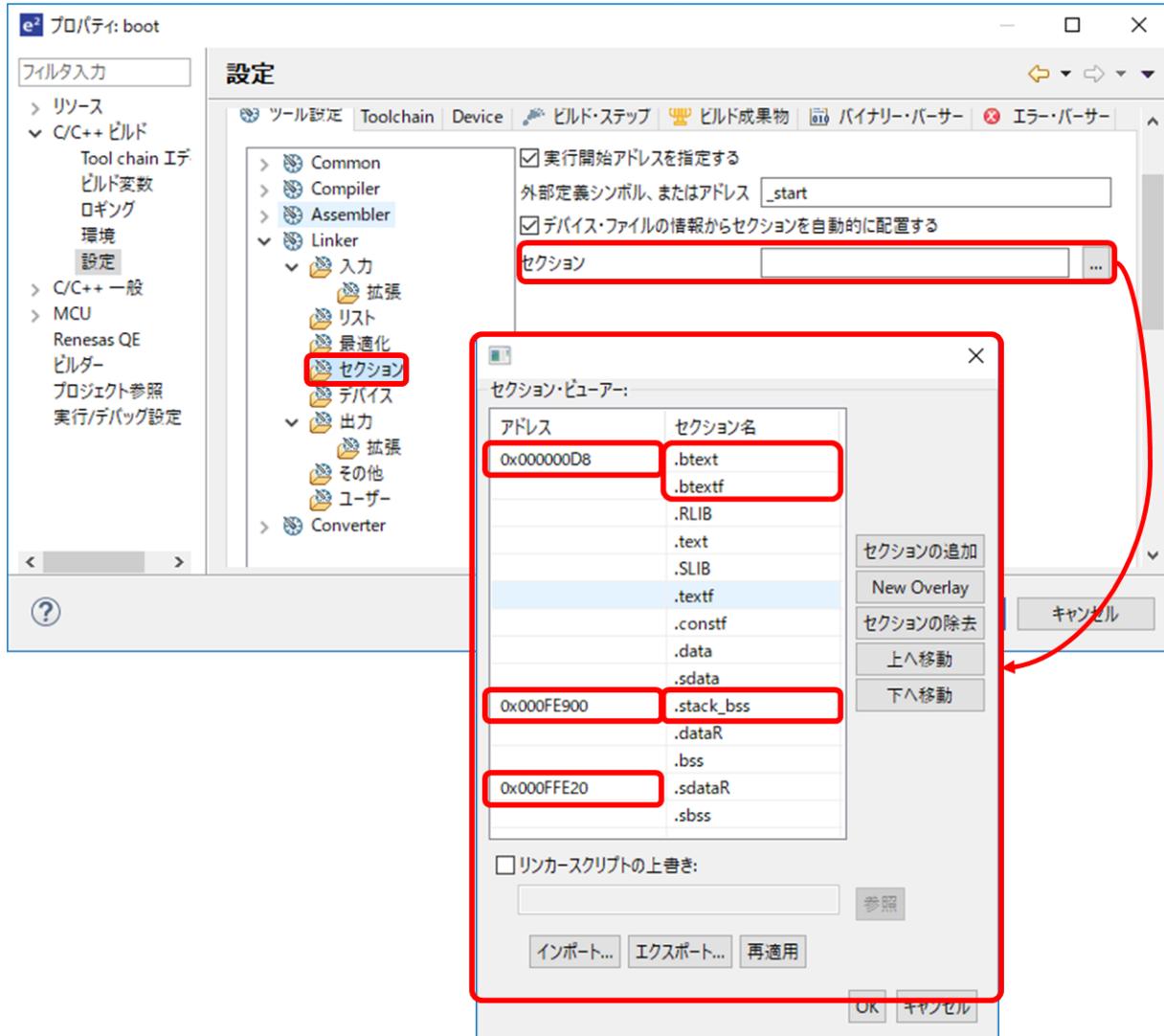


図 12 e<sup>2</sup> studio での設定例

例 CS+  
 [CC-RL (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]

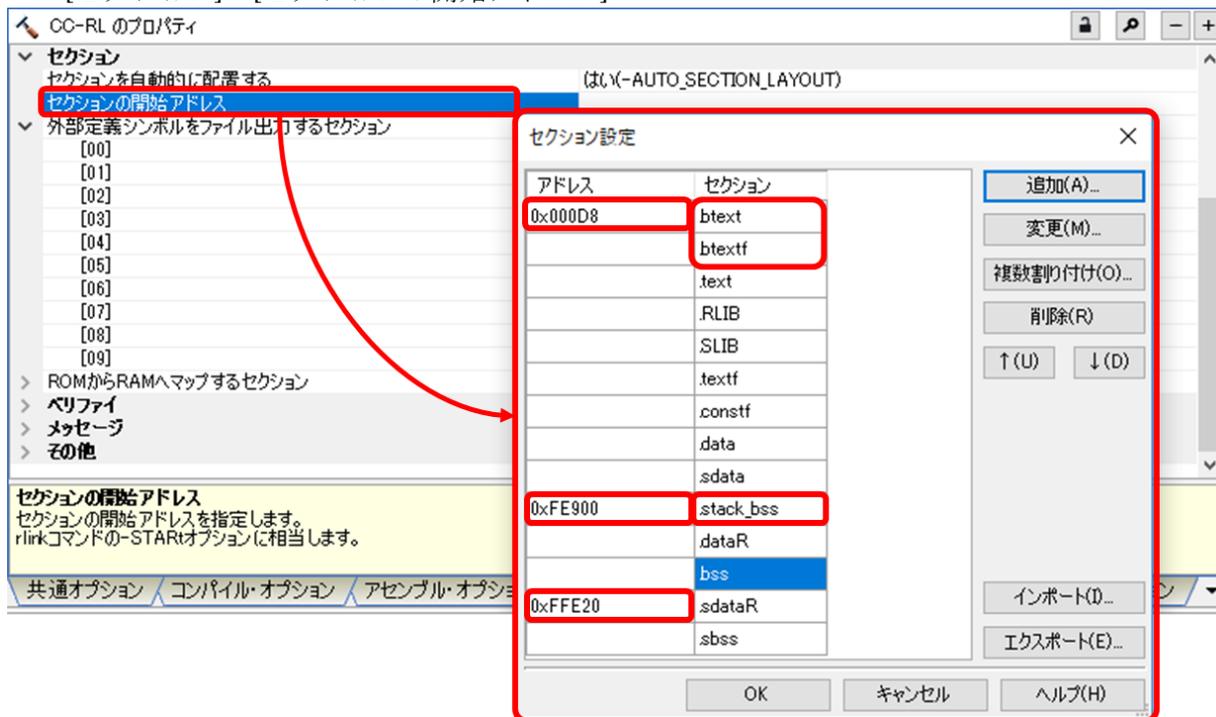


図 13 CS+での設定例

### 3.2.3 フラッシュ領域の割り込み関数に分岐させるためのベクタの設定

リンカの-VECTN オプションで分岐テーブルの該当アドレスを指定してください。

例 e2 studio  
 [プロパティ]→[C/C++ ビルド]→[設定]→[ツール設定]タブ  
 →[Linker]→[出力]→[可変ベクタ・テーブルに指定するベクタ番号とアドレス]  
 →8=0x2010 (8番地に 0x2010) を設定

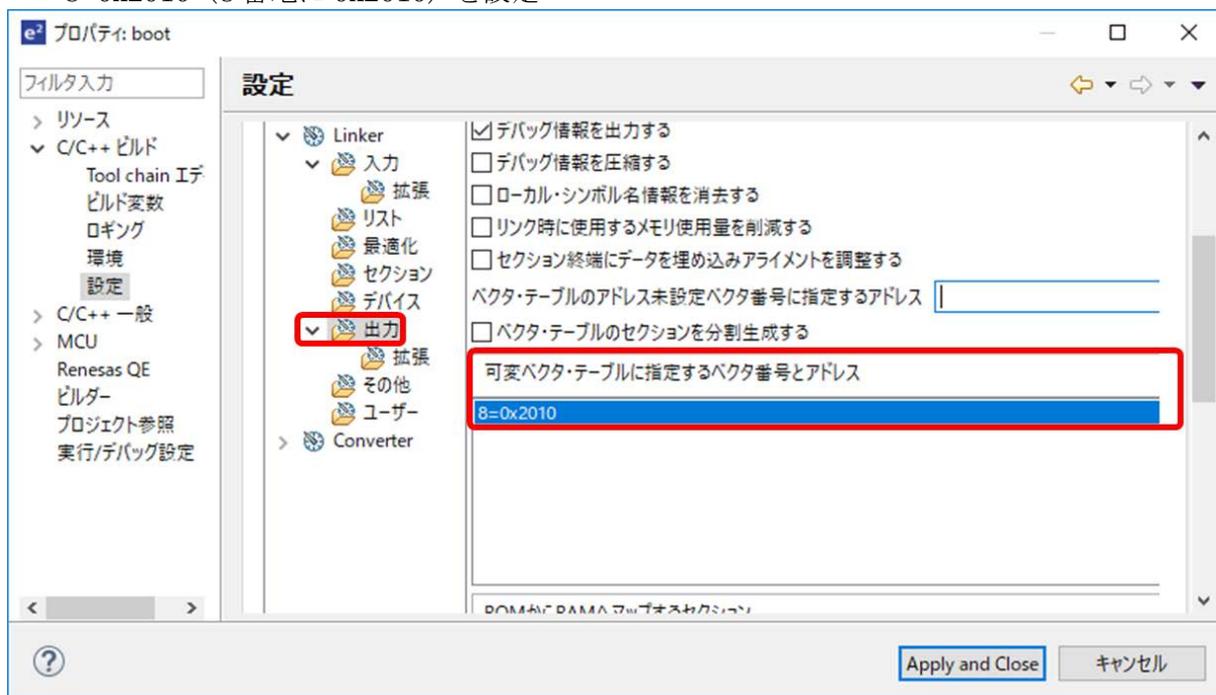


図 14 e2 studio での設定例

例 CS+

[CC-RL (ビルド・ツール)] → [リンク・オプション] タブ  
 → [セクション] → [特定ベクタ・テーブル・アドレスの領域のアドレス]  
 → 8=2010 (8番地に 0x2010) を設定

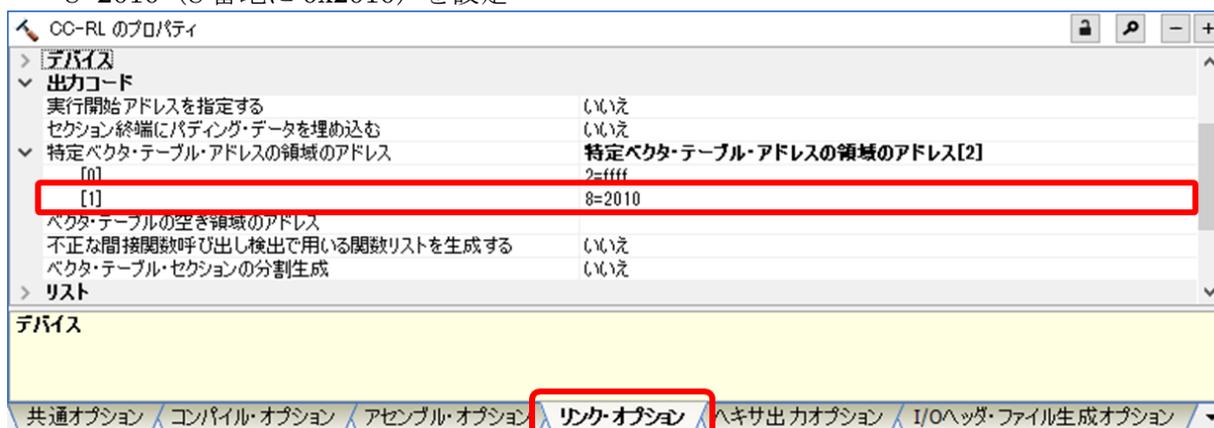


図 15 CS+での設定例

### 3.2.4 オンチップ・デバッグを使用するための設定

1. リンカの -VECTN オプションで 0x0002~0x0003 番地の領域を確保してください。ただし、e<sup>2</sup>studio では、自動で設定されています。
2. リンカの -OCDBG オプションを有効にし、オンチップ・デバッグオプションバイトの値を設定してください。

例 e<sup>2</sup> studio

[プロパティ] → [C/C++ ビルド] → [設定] → [ツール設定] タブ  
 → [Linker] → [デバイス] → [オプションバイト領域のオンチップ・オプション・バイトに値を設定する]

【注】 HardwareDebug ビルド構成 (“E1/E2”または“E2”または“E2 Lite”) の場合、[OCD モニタのメモリ領域を確保する]のチェックボックスのチェックを外してください。

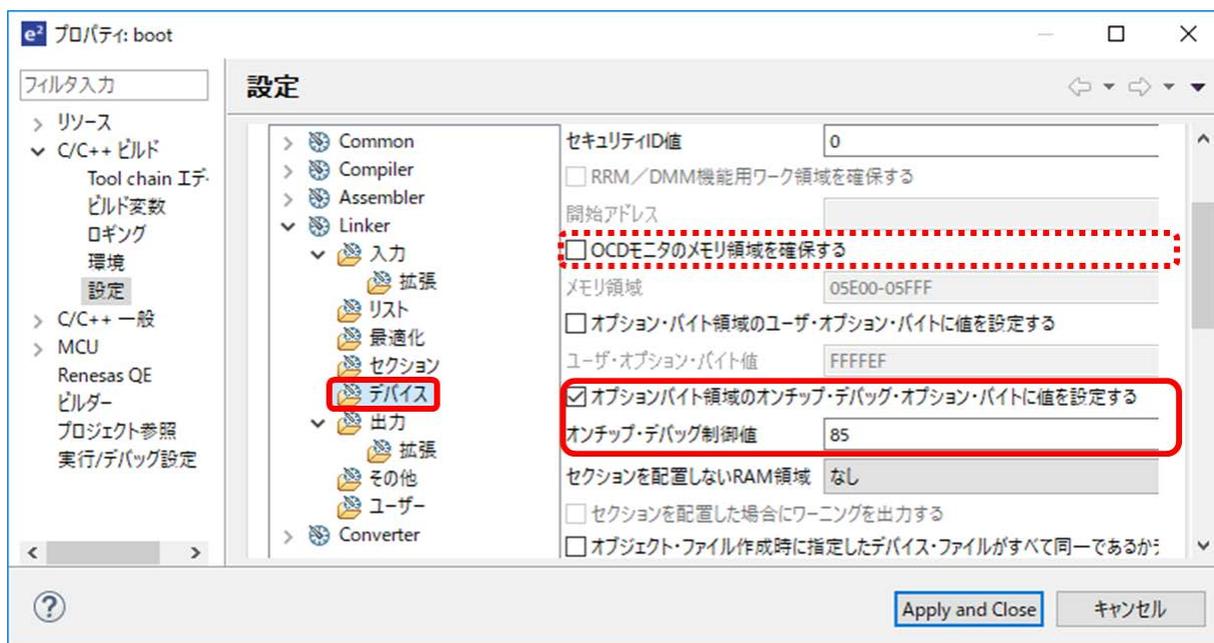


図 16 e<sup>2</sup> studio での設定例

例 CS+

- [CC-RL (ビルド・ツール) ]→[リンク・オプション]タブ
- [デバイス]→[オンチップ・デバッグ・オプション・バイト制御値]
- [セクション]→[特定ベクタ・テーブル・アドレスの領域のアドレス]
- 2=ffff (2番地に 0xffff) を設定

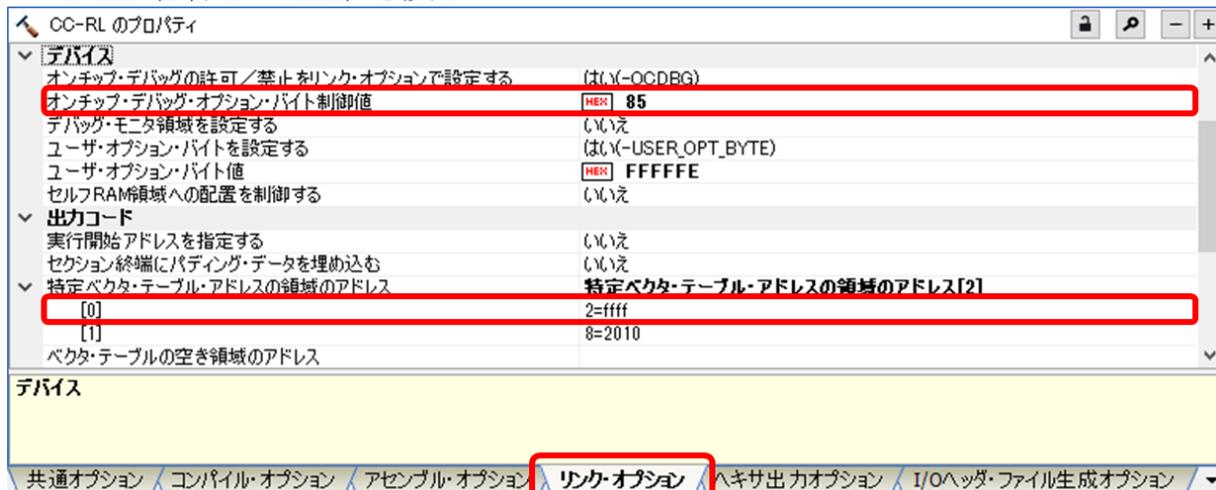


図 17 CS+での設定例

### 3.2.5 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 e<sup>2</sup> studio

- [プロパティ]→[C/C++ ビルド]→[設定]→[ツール設定]タブ
- [Converter]→[出力]
- [ロード・モジュール・コンバータを実行する]にチェック
- 出力ファイル形式を[モトローラ・Sタイプ・ファイルを出力する]を選択
- [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

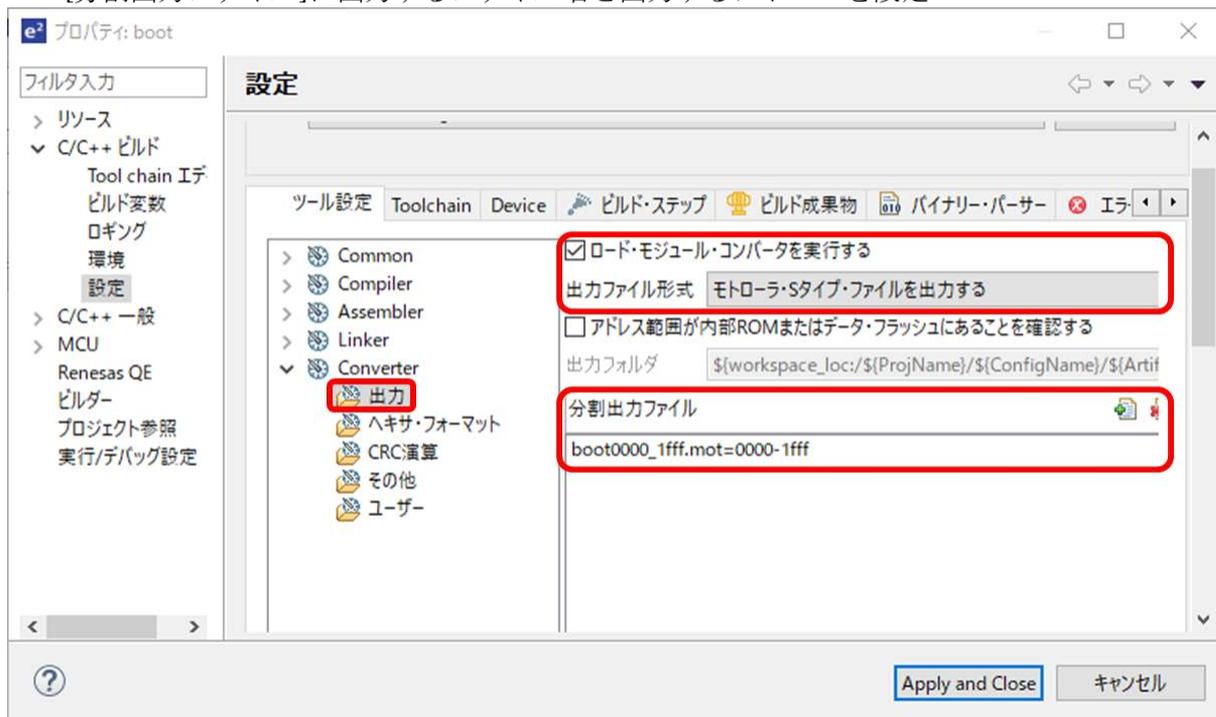


図 18 e<sup>2</sup> studio での設定例

例 CS+

[CC-RL (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

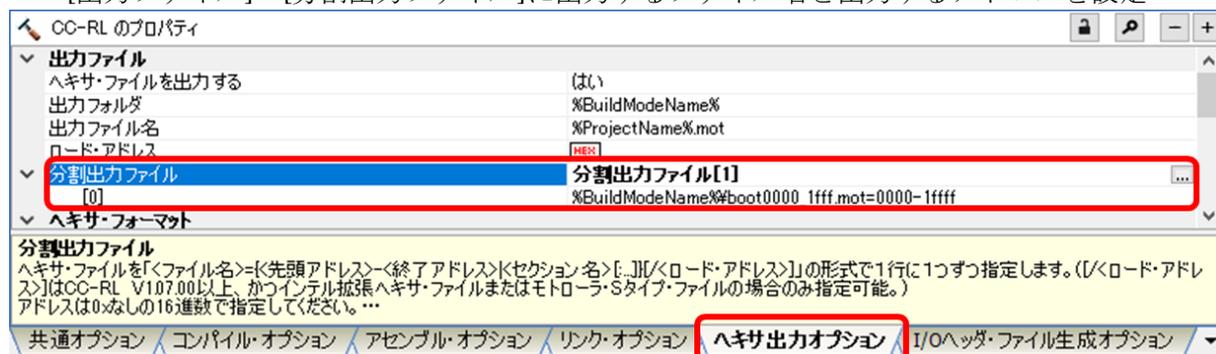


図 19 CS+での設定例

## 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

スタックポイン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スタックポインタを使用するため、フラッシュ領域では再設定しないでください。

例 cstart.asm

```

;-----
; setting the stack pointer
;-----
;$IF (__RENESAS_VERSION__ >= 0x01010000)
; MOVW    SP,#LOWW(__STACK_ADDR_START)
;$ELSE   ; for CC-RL V1.00
; MOVW    SP,#LOWW(_stacktop)
;$ENDIF

```

#### 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、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 _start
  .EXTERN _f1
  .EXTERN _f2
.jtext  .CSEG AT FLASH_TABLE
br      !!_start      ; RESET
.DB4    0xffffffff    ;
.DB4    0xffffffff    ; INTWDTI
.DB4    0xffffffff    ; INTLVI
br      !!_int_INTP0  ; INTP0
.DB4    0xffffffff    ; INTP1
.DB4    0xffffffff    ; INTP2
  ~ 省略 ~
.jtext2 .CSEG AT FLASH_TABLE+INTERRUPT_OFFSET
br      !!_f1
br      !!_f2

```

割り込み用

関数用

### 4.1.3 割り込み関数の定義

- 割り込みベクタはブート領域のプロジェクトで定義する必要があります。
- フラッシュ領域では、`#pragma interrupt` でベクタのアドレス (vect) は指定しないでください。

例 int.c

```
#include "iodefine.h"
#pragma interrupt int_INTP0
volatile char f;
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 e<sup>2</sup>studio

e<sup>2</sup>studio では、ブート領域のプロジェクトをビルド後にフラッシュ領域のプロジェクトにブート領域で作成した外部定義シンボルファイル (boot.fsy) ファイルを登録してください。

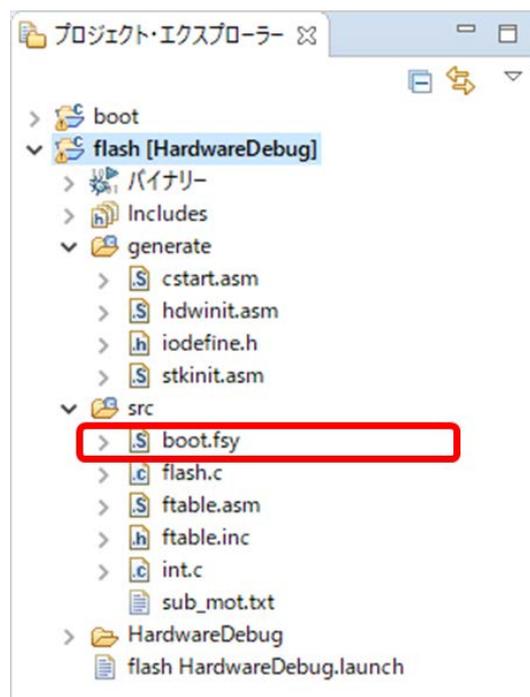


図 20 e<sup>2</sup>studio での設定例

例 CS+

CS+では、ブート領域のプロジェクトをビルドすると自動でプロジェクトに登録されます。

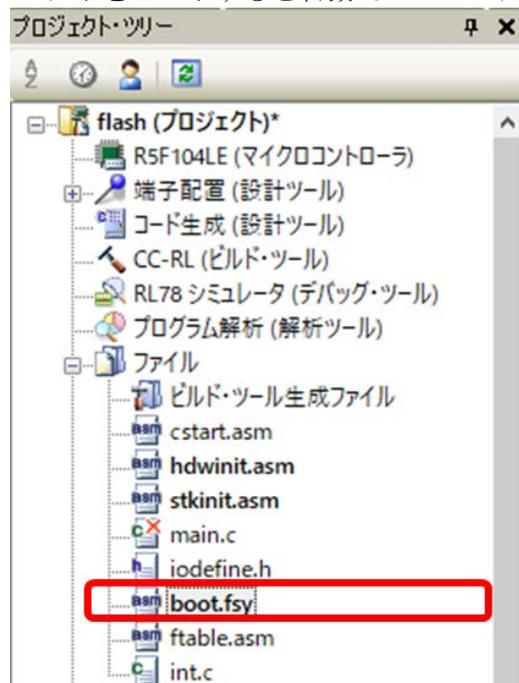


図 21 CS+での設定例

#### 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。

例 e<sup>2</sup> studio

[プロパティ]→[C/C++ ビルド]→[設定]→[ツール設定]タブ  
→[Linker]→[セクション]→[セクション・ビューアー]

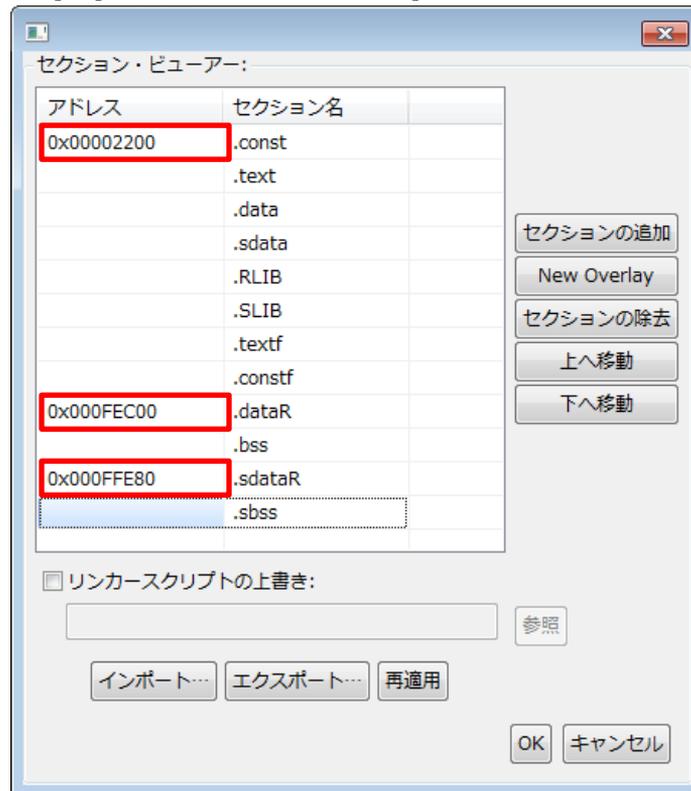


図 22 e<sup>2</sup> studio での設定例

例 CS+

[CC-RL (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 23 CS+での設定例

## 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 e<sup>2</sup> studio

[プロパティ]→[C/C++ ビルド]→[設定]→[ツール設定]タブ

→[Converter]→[出力]

→[ロード・モジュール・コンバータを実行する]にチェック

→出力ファイル形式を[モトローラ・Sタイプ・ファイルを出力する]を選択

→[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

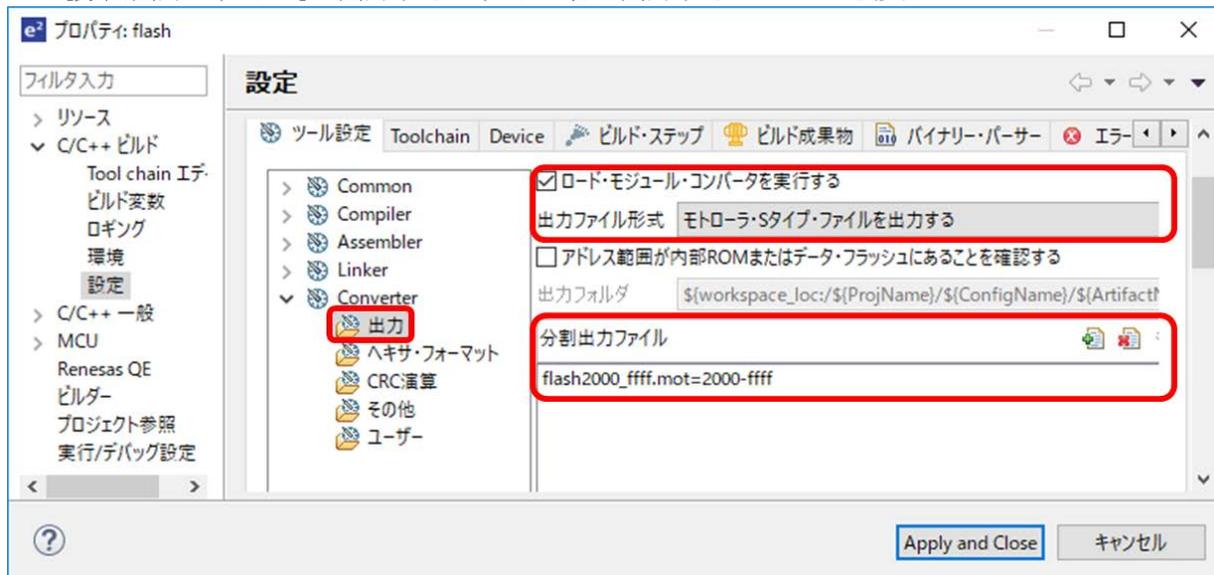


図 24 e<sup>2</sup> studio での設定例

例 CS+

[CC-RL (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

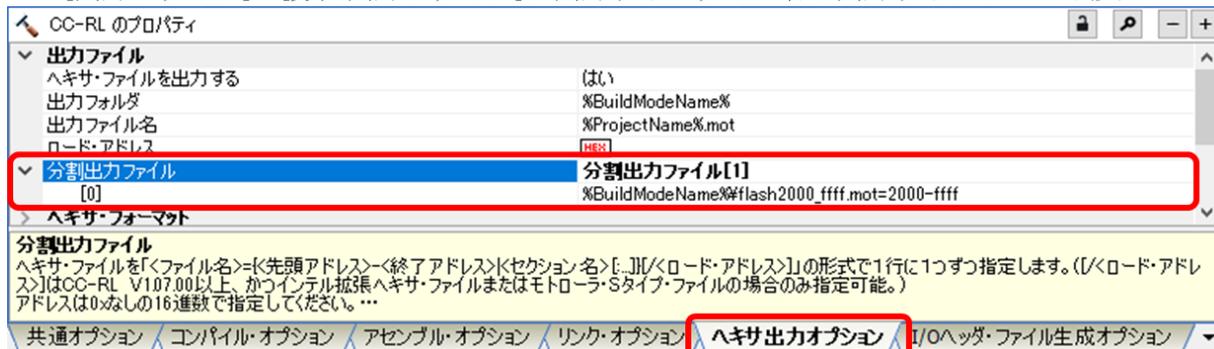


図 25 CS+での設定例

## 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 e<sup>2</sup> studio

[プロパティ]→[C/C++ ビルド]→[設定]→[ビルド・ステップ]タブ

→[ビルド後のステップ]→[コマンド:]

にリンカの実行 (rlink.exe -subcommand=..¥src¥sub\_mot.txt) を追加

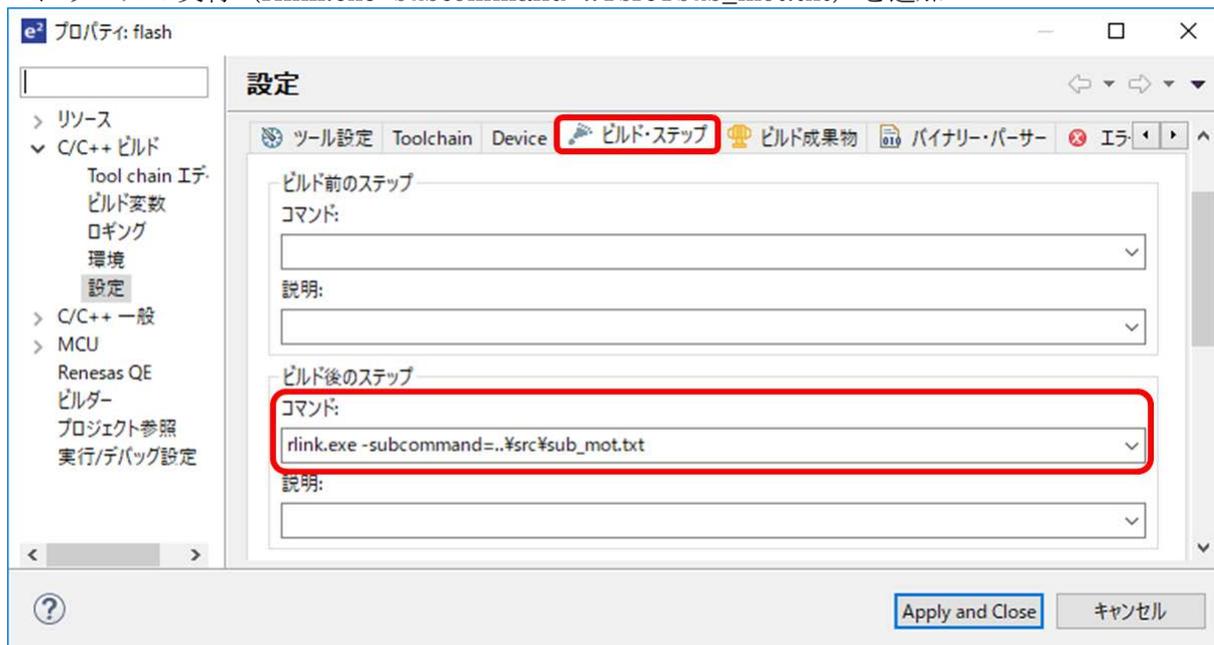


図 26 e<sup>2</sup> studio での設定例

例 CS+

[CC-RL (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-RL¥V1.08.00¥bin¥rlink.exe" -subcommand=sub\_mot.txt

を追加

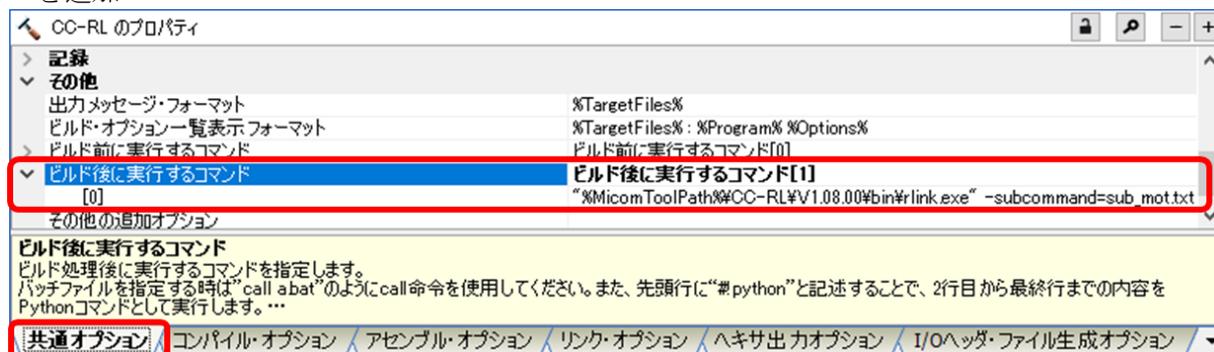


図 27 CS+での設定例

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 (e<sup>2</sup> studio)

```
-input=..¥..¥boot¥HardwareDebug¥boot0000_1fff.mot  
-input=flash2000_ffff.mot  
-form=stype  
-output=boot_flash.mot
```

## sub\_mot.txt の作成例 (CS+)

```
-input=..¥boot¥DefaultBuild¥boot0000_1fff.mot  
-input=..¥DefaultBuild¥flash2000_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

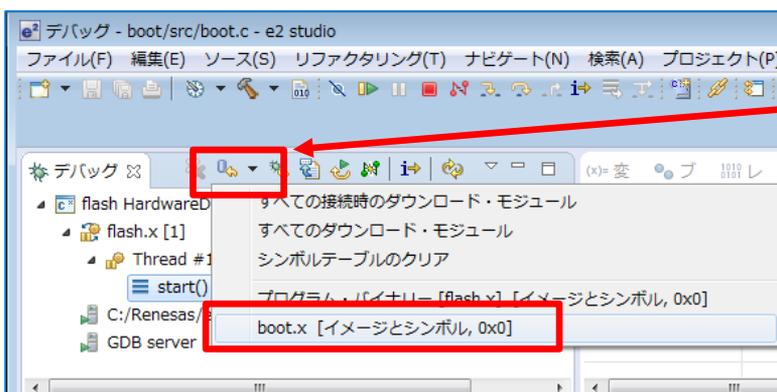
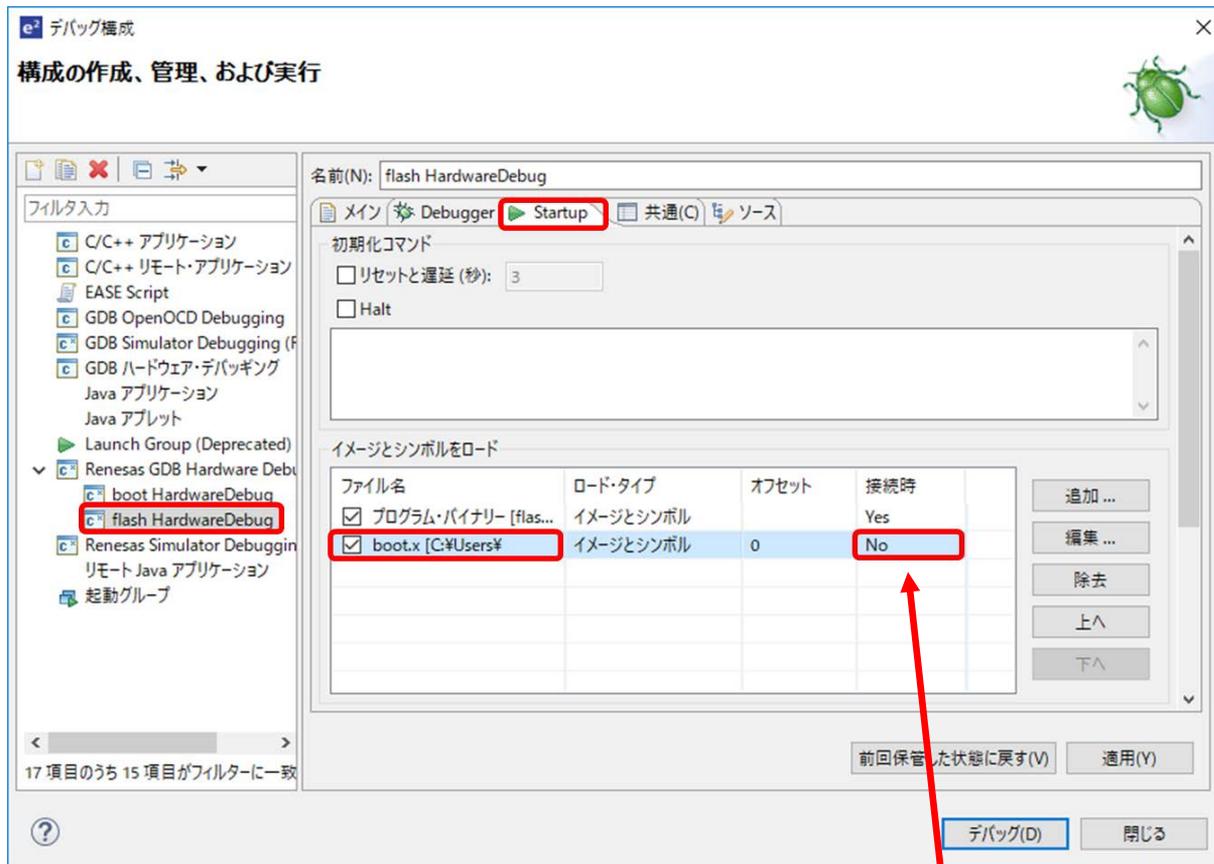
ロードモジュールファイル (\*.abs) は、ブート領域用、フラッシュ領域用の 2 つのファイルが生成されるので、デバッグツールでは、2 つのロードモジュールファイルをダウンロードしてください。

例 e<sup>2</sup> studio

[デバッグ]→[デバッグの構成]→[flash HardwareDebug]→[Startup]タブ

→[イメージとシンボルをロード]

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



※補足 e<sup>2</sup>studio

boot.x は、接続時は「No」にしてください。  
接続時に、ダウンロードしてください。

図 28 e<sup>2</sup> studio での設定例

例 CS+

[RL78 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

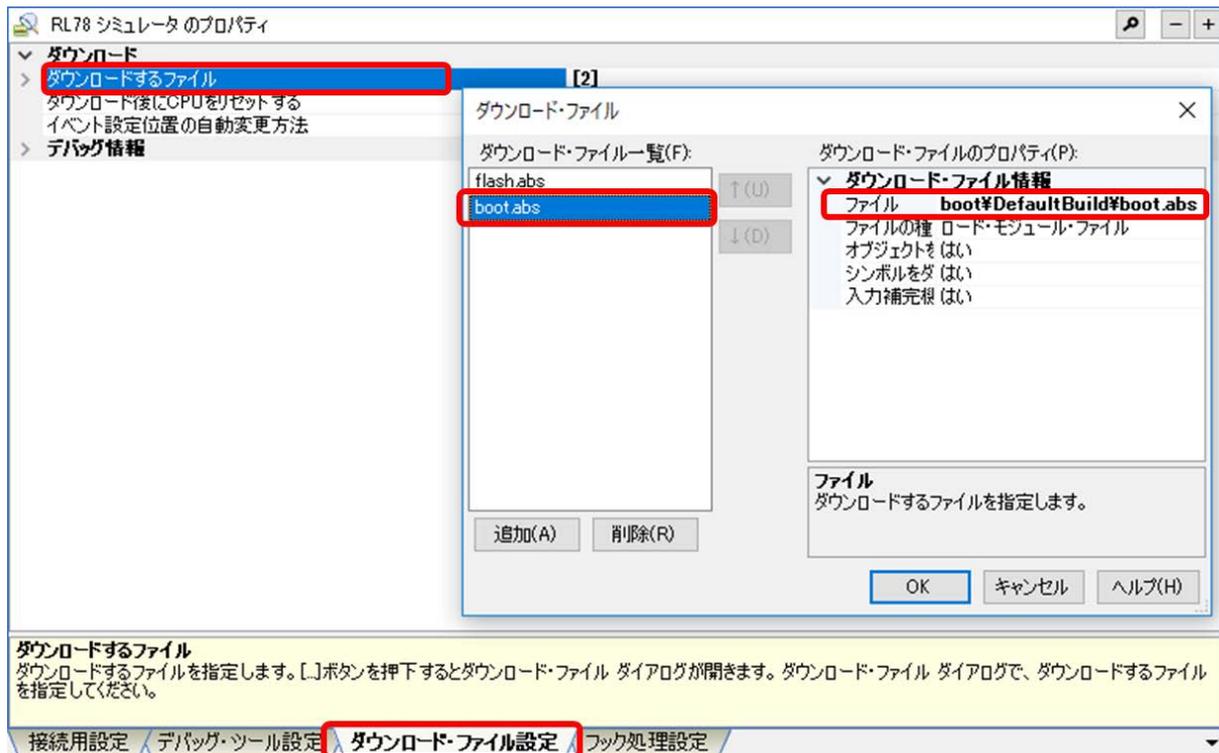


図 29 CS+の設定例

## 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 (boot.c)

```
#include "iodefine.h" /* SFR の定義ファイル */
#pragma interrupt int_INTP1(vect=INTP1) /* 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_INTP1(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 (flash.c)

```
#include "iodefine.h" /* SFR の定義ファイル */
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 */
}
```

## 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                       |
| 1.00 |     | -                    | 初版                         |
| 2.00 |     | 全般                   | テンプレートの変更                  |
|      |     | P3, P29,<br>P32, P47 | 対象ツールのバージョン変更              |
|      |     | P32, P33             | -VECTN オプションの指定値修正         |
|      |     | P40                  | _int_INTP0 への分岐命令の修正       |
| 3.00 |     | 全般                   | テンプレートの変更<br>対象ツールのバージョン変更 |

## 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、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/>