

この度は、フラッシュ・セルフ・プログラミング・ライブラリ Type01 パッケージ Ver.4.00 をご使用いただきまして、誠にありがとうございます。

この添付資料では、フラッシュ・セルフ・プログラミング・ライブラリ Type01 パッケージ Ver.4.00 をお使いいただく上での制限事項および注意事項等を記載しております。ご使用の前に、必ずお読みくださいますようお願い申し上げます。

### 目次

第 1 章 対象製品 .....	2
第 2 章 ユーザーズマニュアル .....	2
第 3 章 バージョンアップ内容 .....	2
第 4 章 注意事項 .....	3
第 5 章 対応ツール .....	3
第 6 章 インストール .....	4
6.1 インストール方法 .....	4
6.2 アンインストール方法 .....	4
6.3 ファイル構成 .....	5
第 7 章 ビルド方法 .....	6
7.1 使用するソフトウェア .....	6
7.2 CS+(旧 CubeSuite+)でのビルド方法 .....	6
7.2.1 C 言語の場合 .....	6
7.2.2 アセンブリ言語の場合 .....	9
7.2.3 自動生成されたファイルの除外(CC-RL コンパイラ使用時のみ) .....	11
7.2.4 ビルド .....	12
7.3 e <sup>2</sup> studio でのビルド方法 .....	12
7.3.1 プロジェクトの作成 .....	12
7.3.2 C 言語の場合 .....	14
7.3.3 アセンブリ言語の場合 .....	18
7.4 ビルド時の注意事項 .....	20
7.4.1 CA78K0R コンパイラ使用時 .....	20
7.4.2 CC-RL コンパイラ使用時 .....	21
第 8 章 デバッグ方法 .....	22
第 9 章 サンプルプログラム .....	23
9.1 サンプルプログラムの初期設定 .....	23
9.2 オプション・バイトとオンチップ・デバッグの設定について .....	24
9.3 C 言語用サンプルプログラムのコンパイルスイッチ .....	26
9.4 内蔵 RAM 領域の定義 .....	27
9.4.1 CA78K0R コンパイラ使用時 .....	27
9.4.2 CC-RL コンパイラ使用時 .....	28
9.4.3 LLVM コンパイラ使用時 .....	32

## 第1章 対象製品

フラッシュ・セルフ・プログラミング・ライブラリ Type01 パッケージ Ver.4.00 では、LLVM コンパイラに対応したフラッシュ・セルフ・プログラミング・ライブラリ Type01 が追加されました。

本リリースノートの対象製品を示します。

製品名	Ver.	インストーラファイル名	Ver.
RL78 ファミリ CA78K0R コンパイラ用 フラッシュ・セルフ・プログラミング・ ライブラリ Type01	V2.20	RENESAS_RL78_FSL_T01_4V00.exe	V4.00
RL78 ファミリ CC-RL コンパイラ用 フラッシュ・セルフ・プログラミング・ライ ブラリ Type01	V2.21		
RL78 ファミリ LLVM コンパイラ用 フラッシュ・セルフ・プログラミング・ライ ブラリ Type01	V2.21		

## 第2章 ユーザーズマニュアル

本バージョンは下記のユーザーズマニュアルに対応しています。

マニュアル名	資料番号
RL78 ファミリ フラッシュ・セルフ・プログラミング・ライブラリ Type01 ユーザーズマニュアル	R01US0050JJ0110

## 第3章 バージョンアップ内容

本パッケージでのバージョンアップ内容を示します。

No.	パッケージ Ver.	対象	内容
1	V4.00	CA78K0R コンパイラ用 ライブラリ V2.20	パッケージ Ver.3.00 からライブラリ本体の変更はありません。
		CC-RL コンパイラ用 ライブラリ V2.21	パッケージ Ver.3.00 からライブラリ本体の変更はありません。
		LLVM コンパイラ用 ライブラリ V2.21	新規に追加されました。
		ユーザーズマニュアル	Rev.1.05 から Rev.1.10 に改版 詳細な改版内容につきましては、ユーザーズマニュアルの改版履歴 をご参照ください。

## 第4章 注意事項

本パッケージに含まれるフラッシュ・セルフ・プログラミング・ライブラリ Type01 の注意事項については、下記、記載の内容、及びユーザーズマニュアルを参照してください。

No.	内容
1	<p>・シミュレータでのデバッグについて</p> <p>フラッシュ・ライブラリのデバッグはシミュレータでは実行できません。デバッグを行う場合は、RL78 マイクロコントローラのオンチップ・デバッグ機能を使用するか、もしくは IECUBE をご用意ください。</p>
2	<p>・RL78/G13 サンプル品(量産製品は該当しません)でのフラッシュ・セルフ・プログラミング・ライブラリご使用時の制限について</p> <p>RL78/G13 サンプル品(量産製品は該当しません)の一部に、フラッシュ・セルフ・プログラミング・ライブラリ割り込みベクタ変更処理の制限となる製品があります。</p> <p>該当製品では、以下のフラッシュ関数を使用できませんのでご注意ください。</p> <ul style="list-style-type: none"> <li>・ FSL_ChangeInterruptTable</li> <li>・ FSL_RestoreInterruptTable</li> </ul>

## 第5章 対応ツール

フラッシュ・セルフ・プログラミング・ライブラリ Type01 を使用する場合には、下記のバージョンを使用してください。

対象ライブラリ	ツール名	バージョン
CA78K0R コンパイラ用ライブラリ	統合開発環境 : CubeSuite+	V1.00.00 以降
	統合開発環境 : CS+	V3.00.00 以降
CC-RL コンパイラ用ライブラリ	統合開発環境 : CS+	V3.01.00 以降
	統合開発環境 : e <sup>2</sup> studio	Version: 2023-10 より掲載 <sup>注</sup>
LLVM コンパイラ用ライブラリ	統合開発環境 : e <sup>2</sup> studio	Version: 2023-10 以降

注. CC-RL コンパイラ V1.00 以降搭載バージョンで使用可能

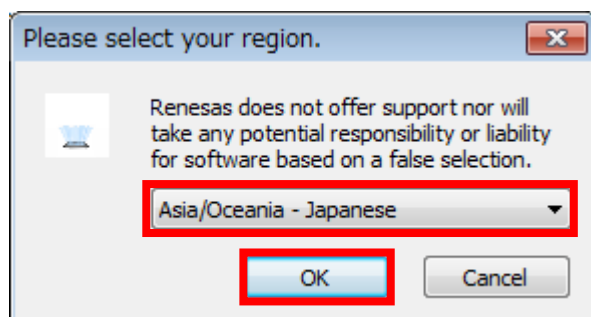
## 第6章 インストール

この章では、フラッシュ・セルフ・プログラミング・ライブラリ Type01 パッケージ Ver.4.00のインストールとアンインストールの手順について説明します。

### 6.1 インストール方法

フラッシュ・セルフ・プログラミング・ライブラリ Type01のインストールは次の手順で行います。

- (1) Windowsを起動します。
- (2) フラッシュ・セルフ・プログラミング・ライブラリ Type01 パッケージ の圧縮ファイルを解凍、インストーラを実行します。
- (3) プルダウンメニューから"Asia/Oceania – Japanese" を選択します。
- (4) "OK"ボタンを押下し、以降、インストーラの指示に従い、実行します。



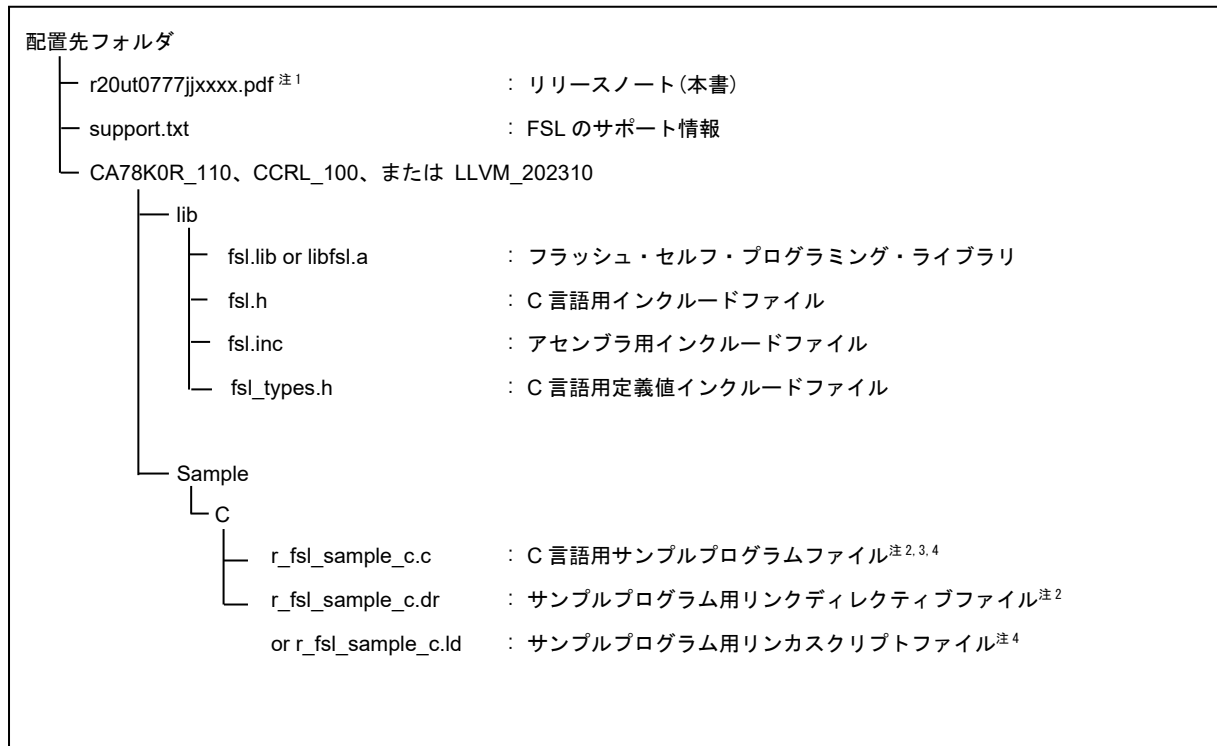
### 6.2 アンインストール方法

フラッシュ・セルフ・プログラミング・ライブラリ Type01のアンインストールは次の手順で行います。

- (1) Windowsを起動します。
- (2) ユーザ任意の場所に配置したフラッシュ・セルフ・プログラミング・ライブラリ Type01が入っているフォルダを削除します。

## 6.3 ファイル構成

フラッシュ・セルフ・プログラミング・ライブラリ Type01パッケージのインストーラを実行することによって、展開されるファイルの構成は、次のとおりです。



注 1. x はバージョン番号、または Rev 番号の為、省略しています。

2. CA78K0R 用のサンプルプログラムを使用する場合は、プログラムファイル(\*.c)とリンクディレクティブファイル(\*.dr)と一緒に組み込んでください。

3. CC-RL 用のサンプルプログラムを使用する場合は、プログラムファイル(\*.c)を組み込んでください。  
CC-RL 用サンプルプログラムのリンク情報は、CS+、または e<sup>2</sup> studio のリンクの設定画面で指定してください。

4. LLVM 用のサンプルプログラムを使用する場合は、プログラムファイル(\*.c)とリンクスクリプトファイル(\*.ld)と一緒に組み込んでください。

## 第7章 ビルド方法

この章では、フラッシュ・セルフ・プログラミング・ライブラリ Type01を用いたプログラムのビルドの手順を説明します。

### 7.1 使用するソフトウェア

フラッシュ・セルフ・プログラミング・ライブラリ Type01 を用いたプログラムをビルドする際には、次の統合開発環境が必要です。

- ・ CA78K0R コンパイラ用：統合開発環境 CS+ V3.00.00 以降、または CubeSuite+ V1.00.00 以降
- ・ CC-RL コンパイラ用：統合開発環境 CS+ V3.01.00 以降、または e<sup>2</sup> studio Version: 2023-10 より掲載<sup>注</sup>  
注. CC-RL コンパイラ V1.00 以降搭載バージョンで使用可能
- ・ LLVM コンパイラ用：統合開発環境 e<sup>2</sup> studio Version: 2023-10 以降

### 7.2 CS+(旧 CubeSuite+)でのビルド方法

CS+を用いてフラッシュ・セルフ・プログラミング・ライブラリ Type01 をユーザプログラムに組み込んで、ビルドする手順を説明します。CS+の対象コンパイラは CA78K0R コンパイラと CC-RL コンパイラです。

#### 7.2.1 C 言語の場合

##### (1) プロジェクトの作成とソースファイルの設定

CS+でプロジェクトを作成し、表示された画面の左側にある[ プロジェクト・ツリー ]から、[ ファイル ]を右クリックして表示されるリストの[ 追加 ]を選択し、[ 既存のファイルを追加 ]をクリックすると、[ 既存のファイルを追加 ]画面が表示されます（図 7-1）。

次に画面中にある[ ファイルの種類 ]のプルダウンメニューをクリックすると、ファイルの種類の一覧が表示されるので、その中にある[ C ソースファイル(\*.c) ]を選択し、ソースファイルとしてユーザプログラムのファイルを登録してください。

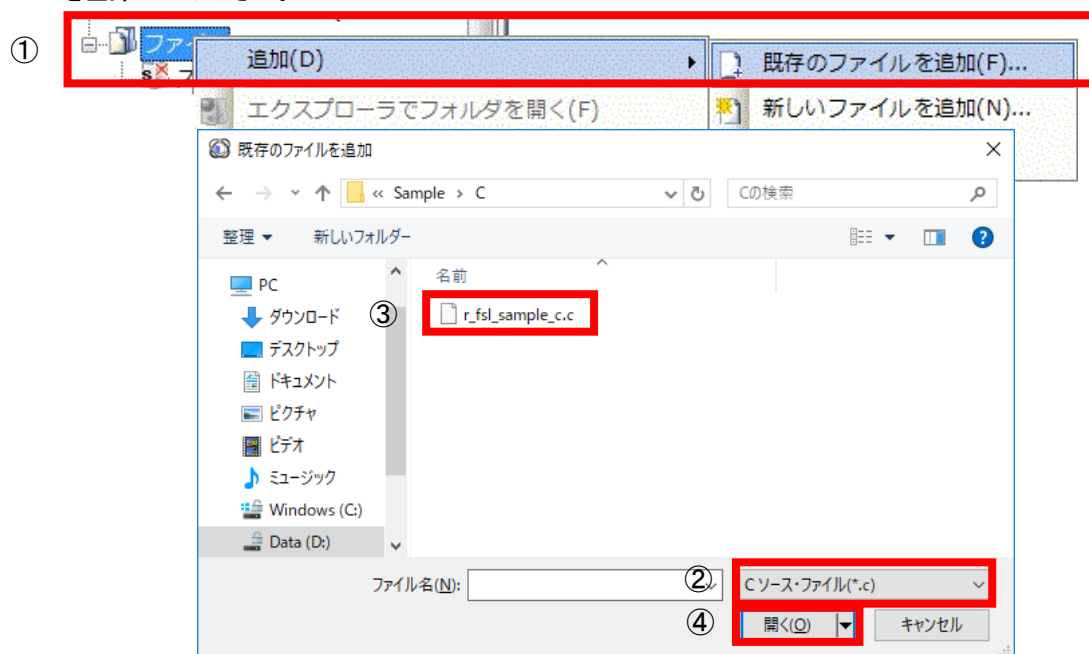


図 7-1 ユーザプログラムファイルの登録

## (2) インクルードファイルの設定

CS+の[ プロジェクト・ツリー ]にある、[ ファイル ]を右クリックして表示されるリストから[ 追加 ]を選択し、[ 既存のファイルを追加 ]をクリックすると、[ 既存のファイルを追加 ]画面が表示されます (図 7-2)。

次に画面中にある[ ファイルの種類 ]のプルダウンメニューをクリックすると、ファイルの種類の一覧が表示されるので、その中にある[ ヘッダファイル(\*.h; \*.inc) ]を選択し、フラッシュ・セルフ・プログラミング・ライブラリのヘッダファイル (fsl.h, fsl\_types.h) を登録してください。

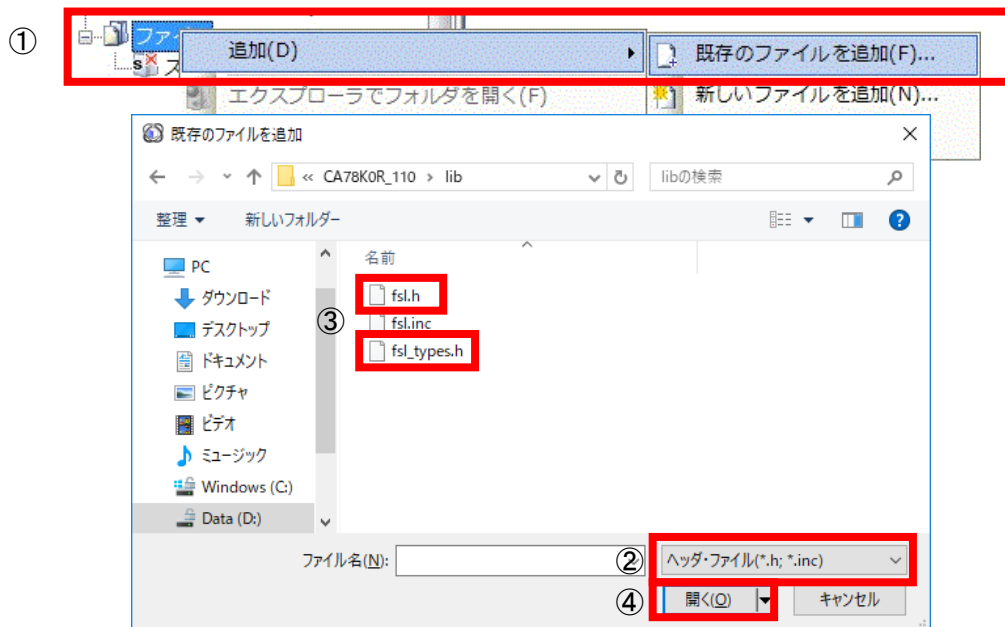


図 7-2 インクルードファイルの登録

## (3) ライブラリファイルの設定

CS+の[ プロジェクト・ツリー ]にある、[ ファイル ]を右クリックして表示されるリストから[ 追加 ]を選択し、[ 既存のファイルを追加 ]をクリックすると、[ 既存のファイルを追加 ]画面が表示されます (図 7-3)。

次に画面中にある[ ファイルの種類 ]のプルダウンメニューをクリックすると、ファイルの種類の一覧が表示されるので、その中にある[ ライブラリファイル(\*.lib) ]を選択し、フラッシュ・セルフ・プログラミング・ライブラリのファイル (fsl.lib) を登録してください。

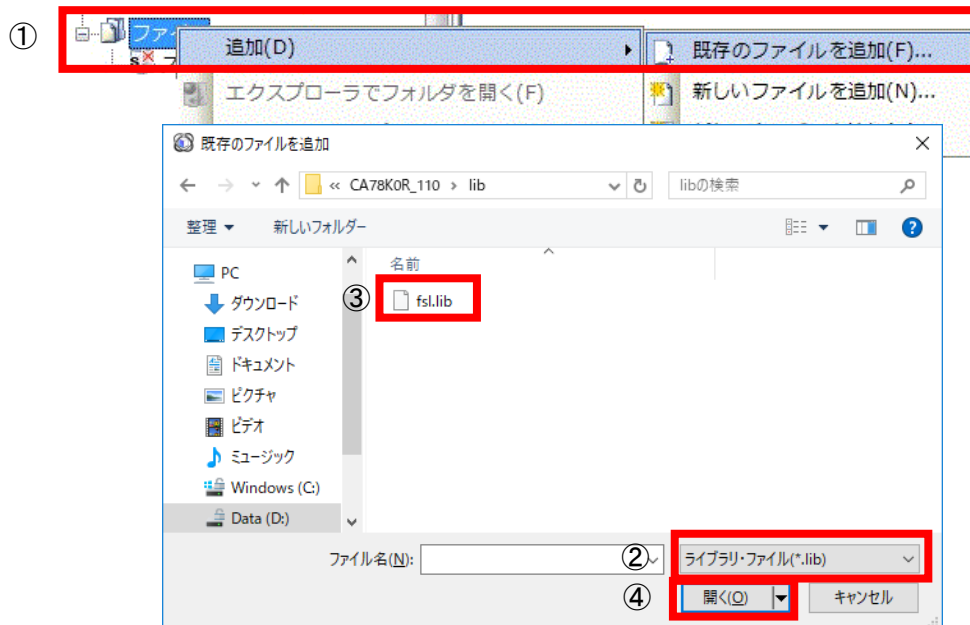


図 7-3 ライブラリファイルの登録

#### (4) リンクディレクティブファイルの設定(CA78K0R コンパイラ使用時のみ)

CS+の[ プロジェクト・ツリー ]にある、[ ファイル ]を右クリックして表示されるリストから[ 追加 ]を選択し、[ 既存のファイルを追加 ]をクリックすると、[ 既存のファイルを追加 ]画面が表示されます(図 7-4)。

次に画面中にある[ ファイルの種類 ]のプルダウンメニューをクリックすると、ファイルの種類の一覧が表示されるので、その中にある[ リンクディレクティブファイル(\*.dr; \*.dir)]を選択し、リンクディレクティブファイル名のユーザファイルを登録してください。

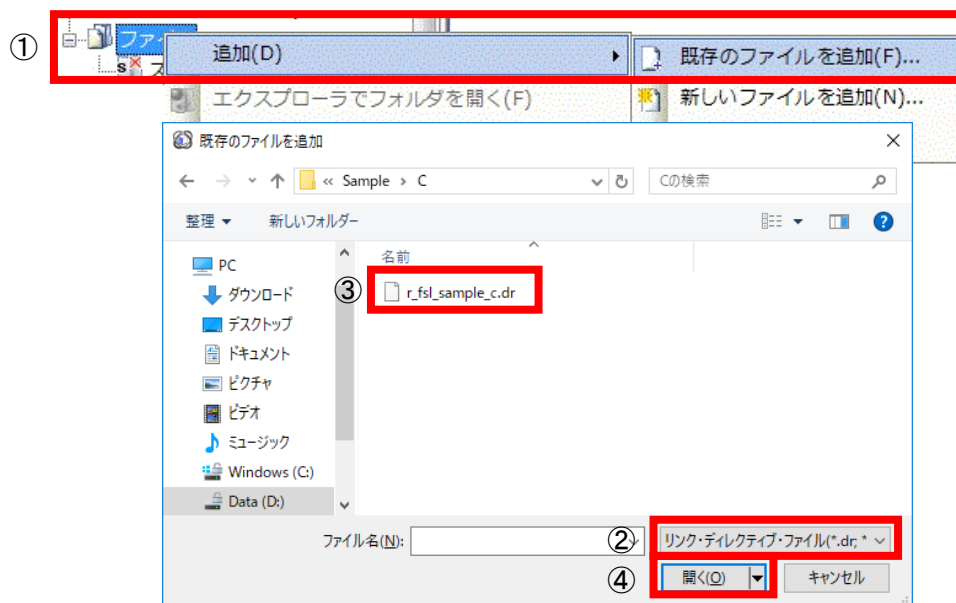


図 7-4 リンクディレクティブファイルの登録

## 7.2.2 アセンブリ言語の場合

### (1) プロジェクトの作成とソースファイルの設定

CS+でプロジェクトを作成し、表示された画面の左側にある[ プロジェクト・ツリー ]から、[ ファイル ]を右クリックして表示されるリストの[ 追加 ]を選択し、[ 既存のファイルを追加 ]をクリックすると、[ 既存のファイルを追加 ]画面が表示されます（図 7-5）。

次に画面中にある[ ファイルの種類 ]のプルダウンメニューをクリックすると、ファイルの種類の一覧が表示されるので、その中にある[ アセンブルファイル(\*.asm) ]を選択し、ソースファイルとしてユーザプログラムのファイルを登録してください。

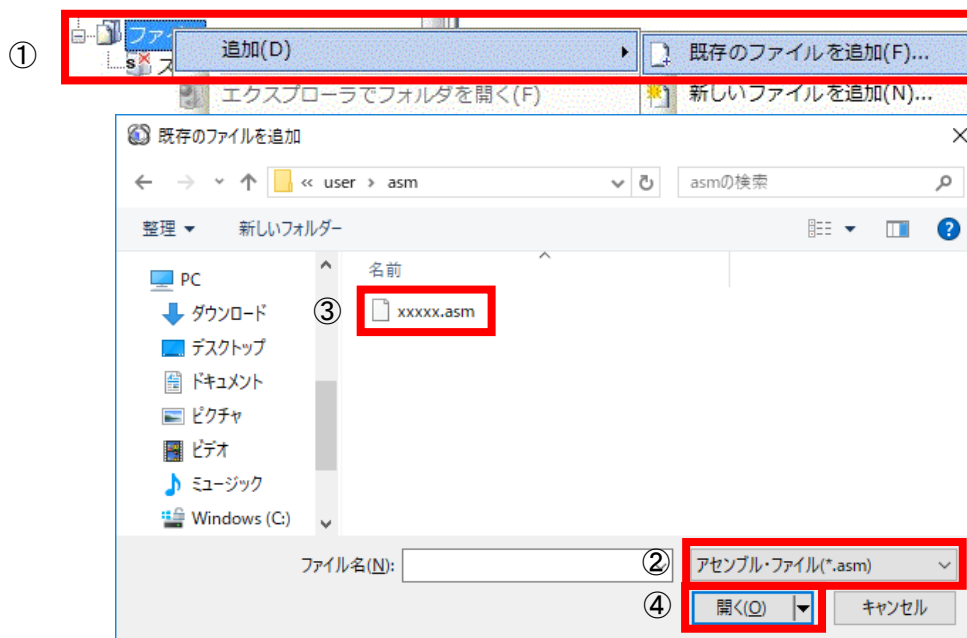


図 7-5 ユーザプログラムファイルの登録

### (2) インクルードファイルの設定

CS+の[ プロジェクト・ツリー ]にある、[ ファイル ]を右クリックして表示されるリストから[ 追加 ]を選択し、[ 既存のファイルを追加 ]をクリックすると、[ 既存のファイルを追加 ]画面が表示されます（図 7-6）。

次に画面中にある[ ファイルの種類 ]のプルダウンメニューをクリックすると、ファイルの種類の一覧が表示されるので、その中にある[ ヘッダファイル(\*.h; \*.inc) ]を選択し、フラッシュ・セルフ・プログラミング・ライブラリのヘッダファイル (fsl.inc) を登録してください。

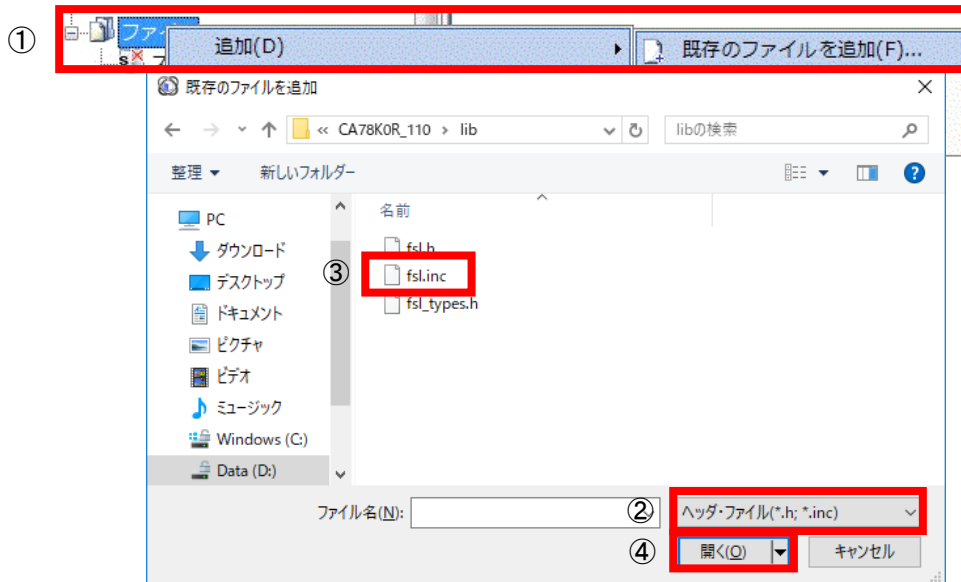


図 7-6 インクルードファイルの登録

### (3) ライブラリファイルの設定

CS+の[ プロジェクト・ツリー ]にある、[ ファイル ]を右クリックして表示されるリストから[ 追加 ]を選択し、[ 既存のファイルを追加 ]をクリックすると、[ 既存のファイルを追加 ]画面が表示されます(図 7-7)。

次に画面中にある[ ファイルの種類 ]のプルダウンメニューをクリックすると、ファイルの種類の一覧が表示されるので、その中にある[ ライブラリファイル(\*.lib) ]を選択し、フラッシュ・セルフ・プログラミング・ライブラリのファイル (fsl.lib) を登録してください。

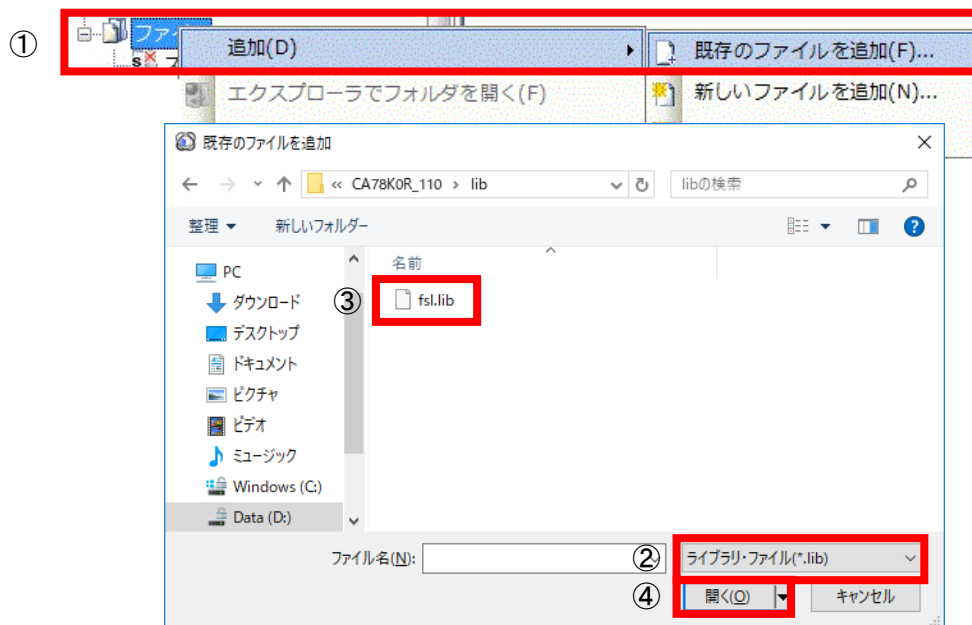


図 7-7 ライブラリファイルの登録

#### (4) リンクディレクティブの設定 (CA78K0R コンパイラ使用時のみ)

CS+の[ プロジェクト・ツリー ]にある、[ ファイル ]を右クリックして表示されるリストから[ 追加 ]を選択し、[ 既存のファイルを追加 ]をクリックすると、[ 既存のファイルを追加 ]画面が表示されます。(図 7-8)

次に画面中にある[ ファイルの種類 ]のプルダウンメニューをクリックすると、ファイルの種類の一覧が表示されるので、その中にある[ リンクディレクティブファイル(\*.dr; \*.dir) ]を選択し、リンクディレクティブファイル名のユーザファイルを登録してください。

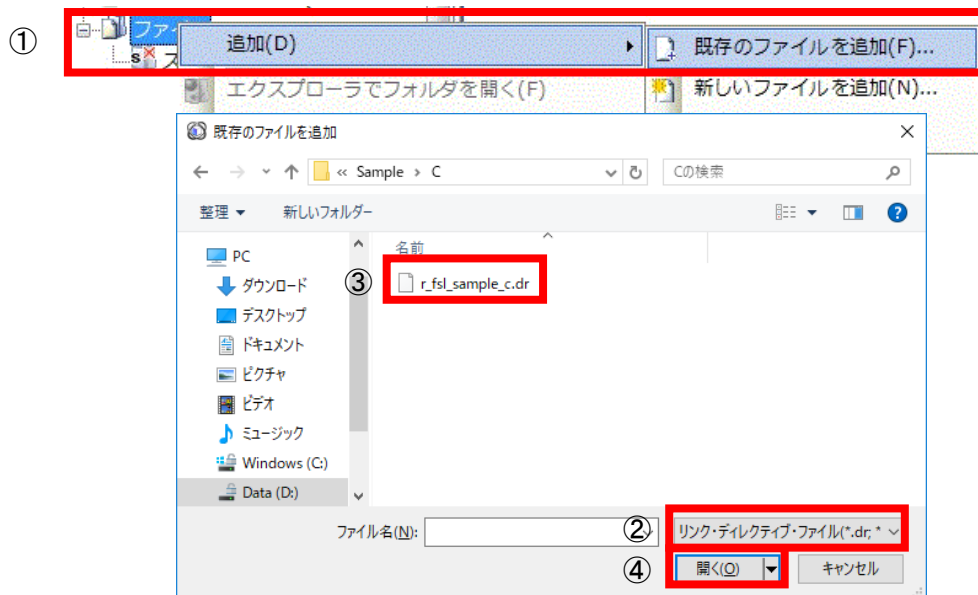


図 7-8 リンクディレクティブファイルの登録

#### 7.2.3 自動生成されたファイルの除外(CC-RL コンパイラ使用時のみ)

CC-RL コンパイラ用 CS+では、[ プロジェクト・ツリー ]の[ ファイル ]にいくつかの自動生成されるファイルがあります。この中で、"main.c"と"hdwinit.asm"の処理は、既にフラッシュ・セルフ・プログラミング・ライブラリのサンプルプログラムに含まれています。その為、この2つのファイルをプロジェクトから除外する必要があります。

アセンブリ言語を使用する場合は、サンプルプログラムを使用しないので"main.c"のみを除外します。

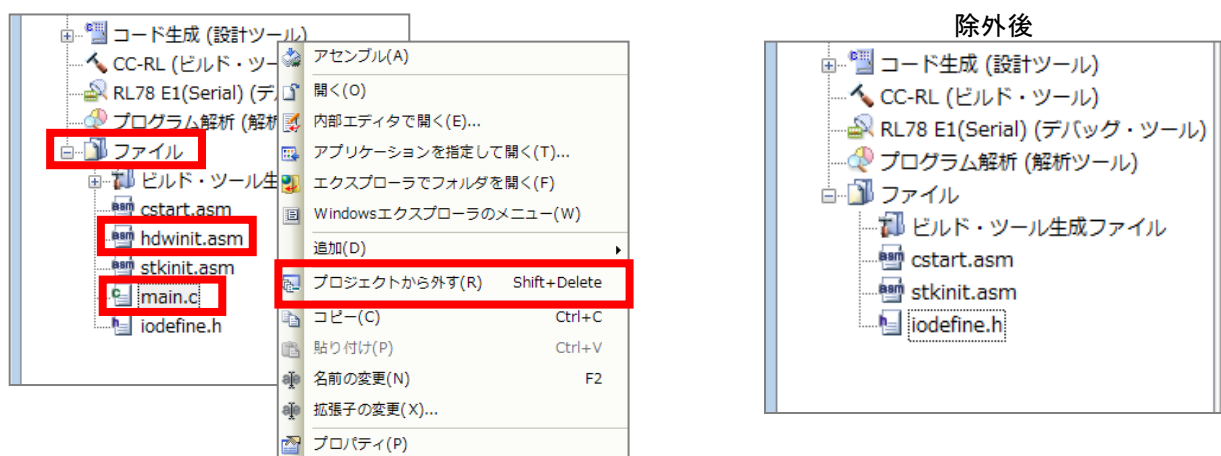


図 7-9 自動生成ファイルの除外

## 7.2.4 ビルド

CS+のメニュー [ ビルド ] → [ ビルド・プロジェクト ] をクリックして、ビルドを実行します。

## 7.3 e<sup>2</sup> studio でのビルド方法

e<sup>2</sup> studio を用いてフラッシュ・セルフ・プログラミング・ライブラリ Type01 をユーザプログラムに組み込んで、ビルドする手順を説明します。e<sup>2</sup> studio の対象コンパイラはCC-RL コンパイラと LLVM コンパイラです。

### 7.3.1 プロジェクトの作成

e<sup>2</sup> studio を起動し、[ファイル]メニューの[新規]から[C/C++ Project]を選択し、"新規 C/C++ プロジェクトのテンプレート"ウィンドウを起動します。

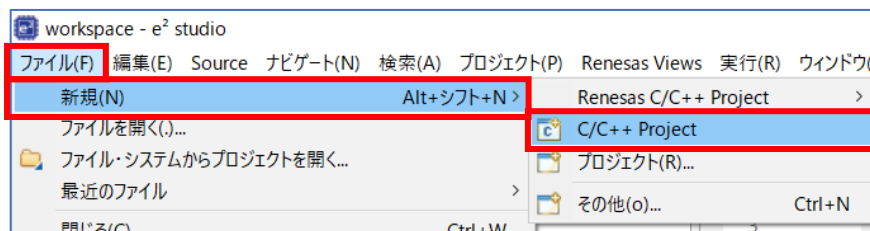


図 7-10 新規プロジェクトの作成

- CC-RL コンパイラを使用する場合は、[Renesas CC-RL C/C++ Executable Project]を選択、"次へ"ボタンを押します。

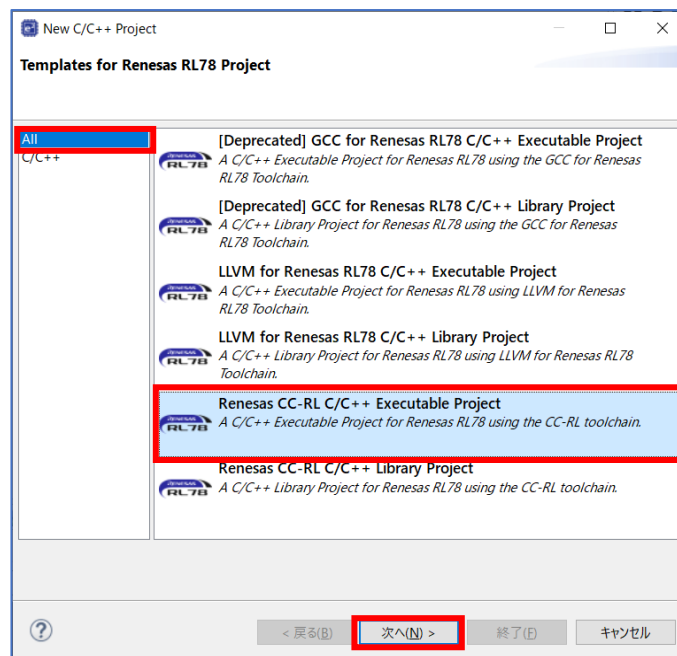


図 7-11 ツールチェーンに CC-RL コンパイラを選択する場合

"New Renesas CC-RL Executable Project"ウィンドウで、プロジェクト名を入力して"次へ"ボタンを押します。

- LLVM コンパイラを使用する場合は、[LLVM for Renesas RL78 C/C++ Executable Project]を選択、"次へ"ボタンを押します。

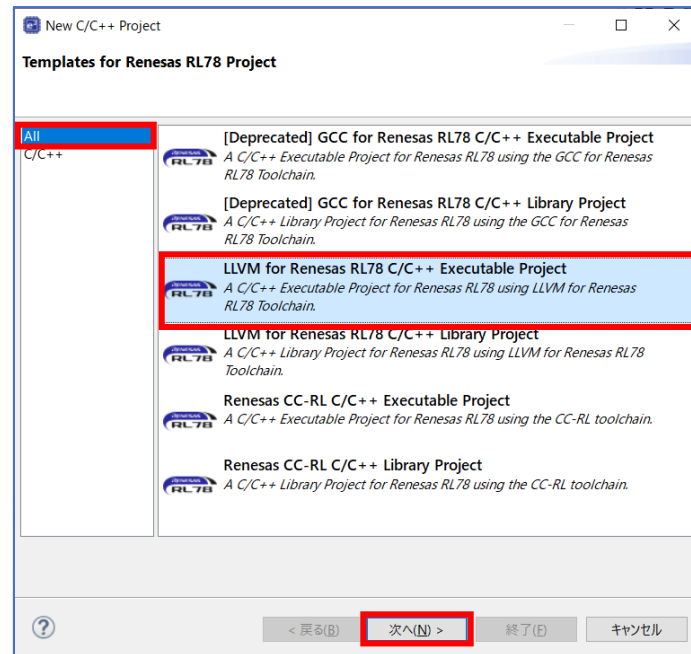


図 7-12 ツールチェーンに LLVM コンパイラを選択する場合

"New Renesas LLVM for Renesas RL78 Executable Project"ウィンドウで、プロジェクト名を入力して"次へ"ボタンを押します

[Device Settings]の[ターゲット・デバイスで、"RL78 - G13" - "R5F100LE"を選択します。(ターゲット・デバイスを RL78/G13 [型名:R5F100LE]とした場合)

[Configurations]で"Hardware Debug 構成を生成"にチェックが入った状態で E2 Lite (RL78)を選択し、[終了]ボタンを押します。デバッグ・ツールに E2 Lite を選択し、オンチップ・デバッグを実施することを前提としています。

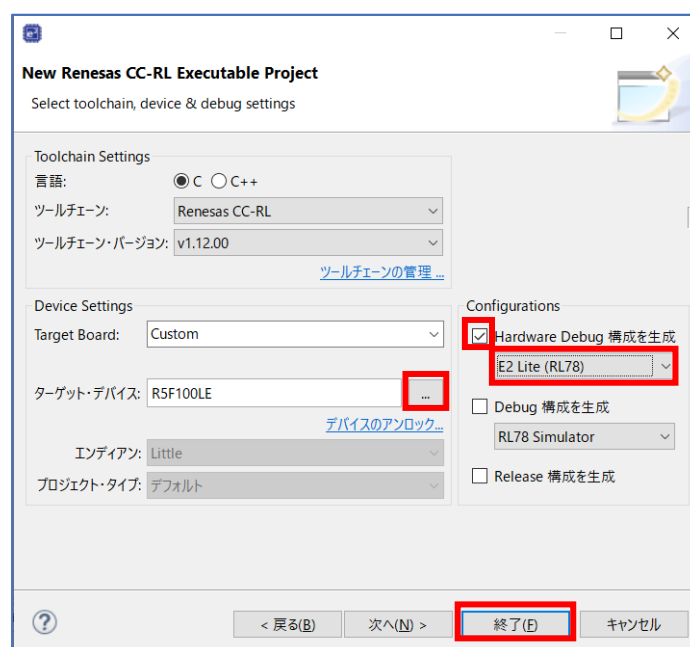


図 7-13 デバイスの選択

## 7.3.2 C 言語の場合

### (1) ソース、インクルードファイルの設定

- ・作成したプロジェクトへの、フラッシュ・セルフ・プログラミング・ライブラリの追加
- CC-RL では、e<sup>2</sup> studio が出力する"src"フォルダにフラッシュ・セルフ・プログラミング・ライブラリのファイル"fsl.h", "fsl\_types.h", "fsl.lib", "r\_fsl\_sample\_c.c"を登録します。(図 7-14)

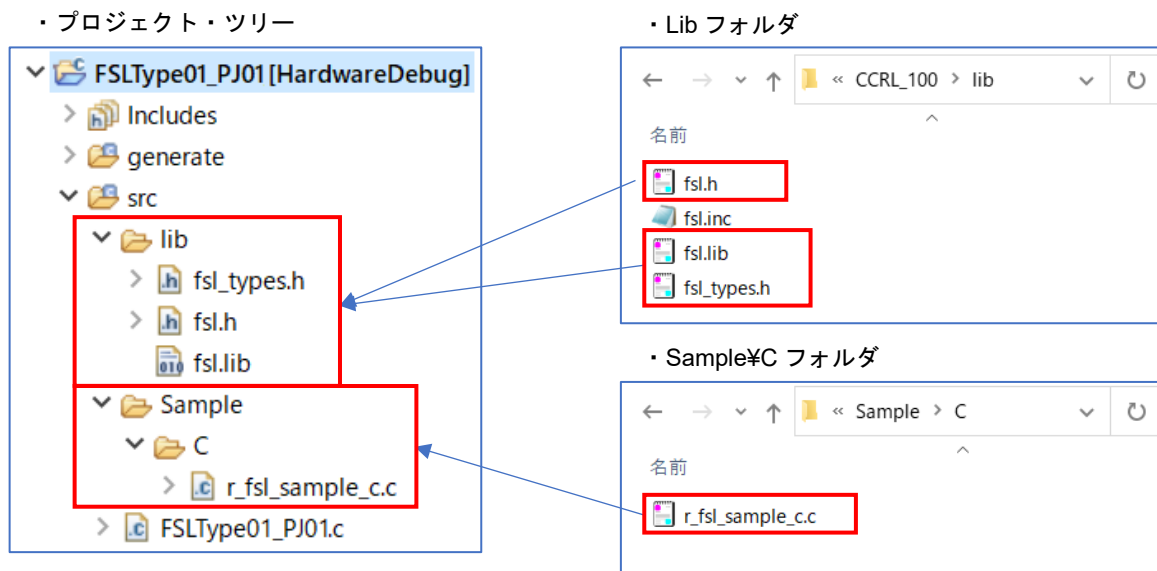


図 7-14 ソース、インクルードファイルの登録(CC-RL)

- LLVM では、e<sup>2</sup> studio が出力する"src"フォルダにフラッシュ・セルフ・プログラミング・ライブラリのファイル"fsl.h", "fsl\_types.h", "fsl.lib", "r\_fsl\_sample\_c.c", "r\_fsl\_sample\_c.ld"を登録します。(図 7-15)

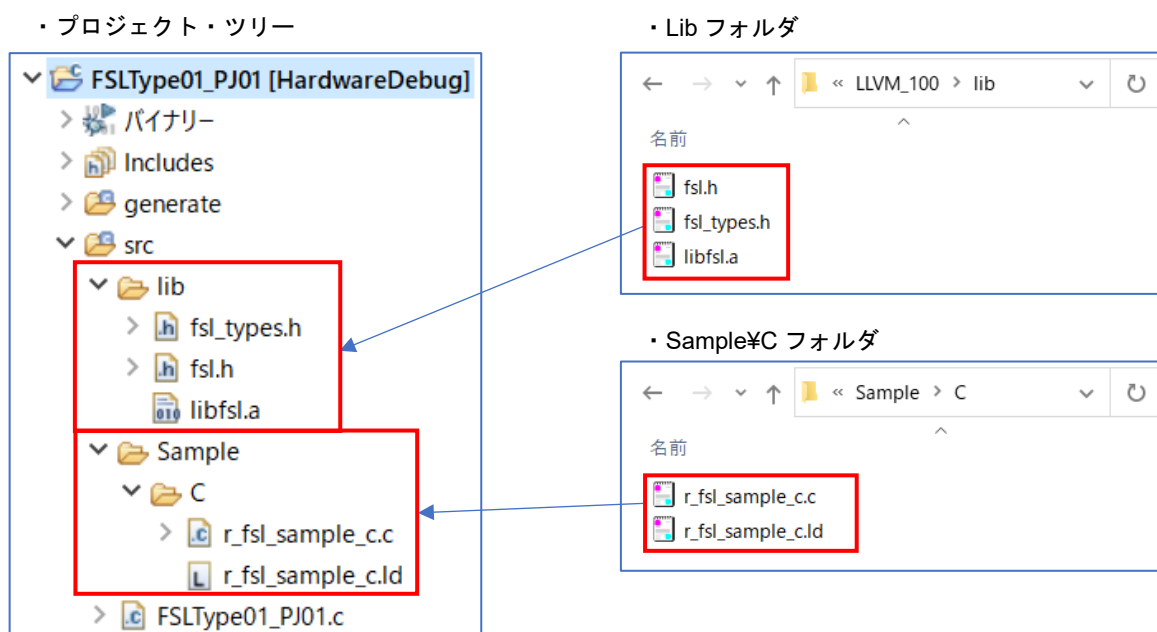


図 7-15 ソース、インクルードファイルの登録(LLVM)

・統合開発環境の機能により自動的に追加されたファイルの除外

作成されたプロジェクトには、自動的に追加されるファイルがあります。これらと同様のファイルは、FSL Type01 の"sample"フォルダ内にも存在するため、ツリーで各ファイルを選択し、各統合開発環境の機能を使用して、プロジェクトから外します。

プロジェクト・ツリー上のファイルをマウス右クリック、「プロパティ」で表示された[設定]画面で、「ビルドからリソースを除外」にチェックを入れ、対象ファイルを除外します(フォルダから削除も可能)。

- CC-RL コンパイラ使用時は、[プロジェクト名]/generate フォルダ内の"hdwinit.asm"と、[プロジェクト名]/src フォルダ内の[プロジェクト名].c(ここでは"FSLType01\_PJ01.c")が対象。

- LLVM コンパイラ使用時は、[プロジェクト名]/generate フォルダ内の"linker\_script.ld"と、[プロジェクト名]/src フォルダ内の[プロジェクト名].c(ここでは" FSLType01\_PJ01.c")が対象。

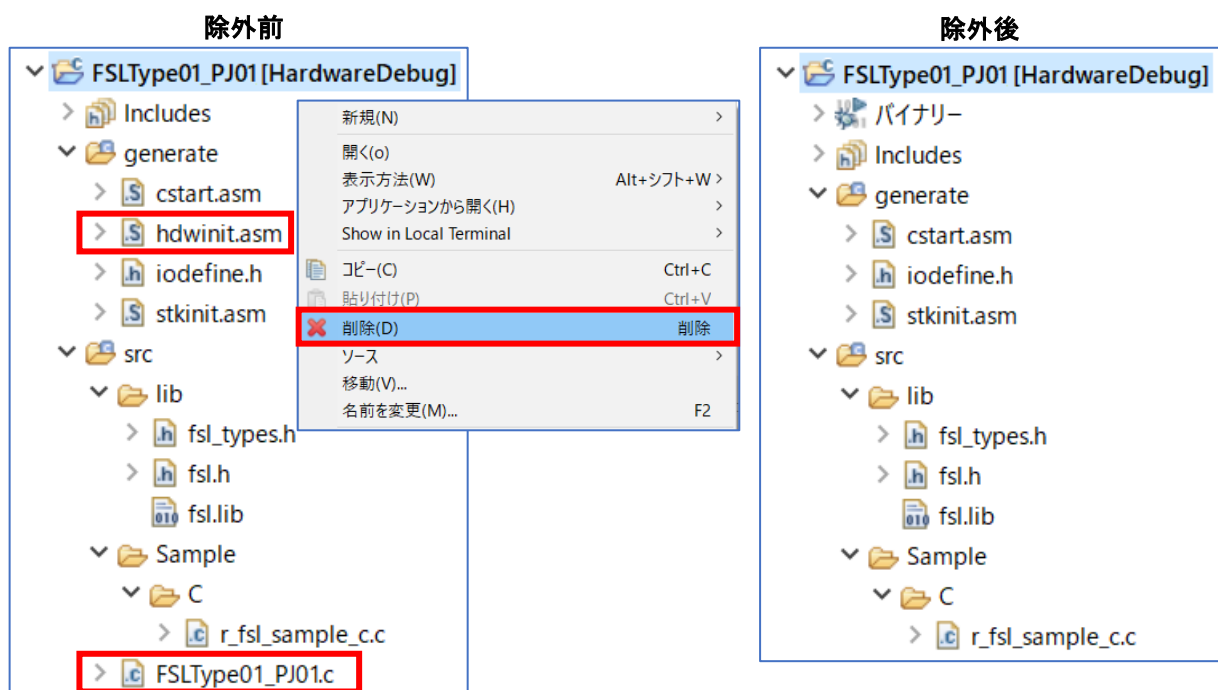


図 7-16 ファイルの除外例

## (2) ライブラリファイルの設定

- CC-RL コンパイラの場合、e<sup>2</sup> studio のプロジェクト・ツリーの[プロジェクト]上でマウスの右クリックで"プロパティ"を選択、"C/C++ビルド" [設定] - "Linker" [入力]で表示された画面の"リンクするリロケータブル・ファイル、オブジェクト・ファイル、およびライブラリー・ファイル"の右側にある+ボタンをクリックして表示された"ファイルの追加"ウィンドウで[形式]を"library"に変更してから、ライブラリファイルのパスを登録します。

フラッシュ・セルフ・プログラミング・ライブラリのファイル"fsl.lib"のファイルパスを登録してください。

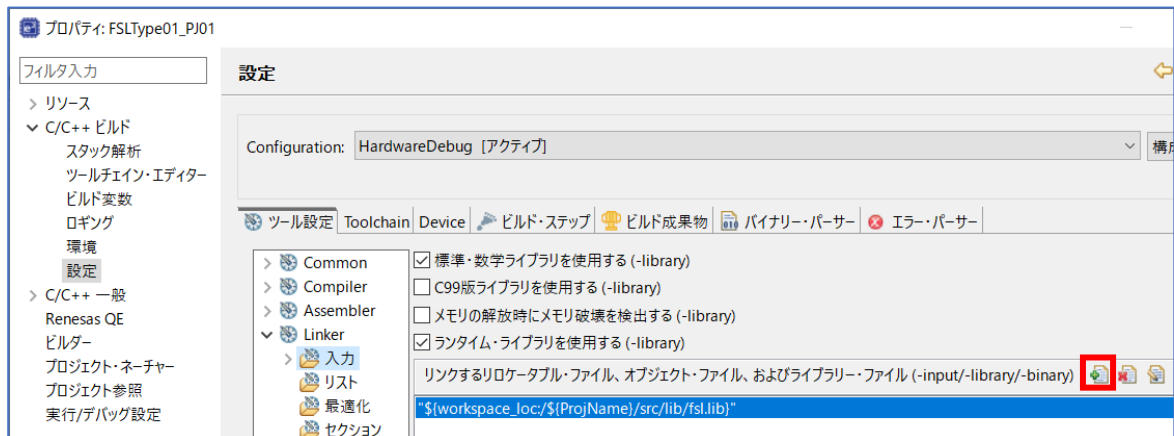


図 7-17(a) ライブラリファイルの登録(CC-RL)

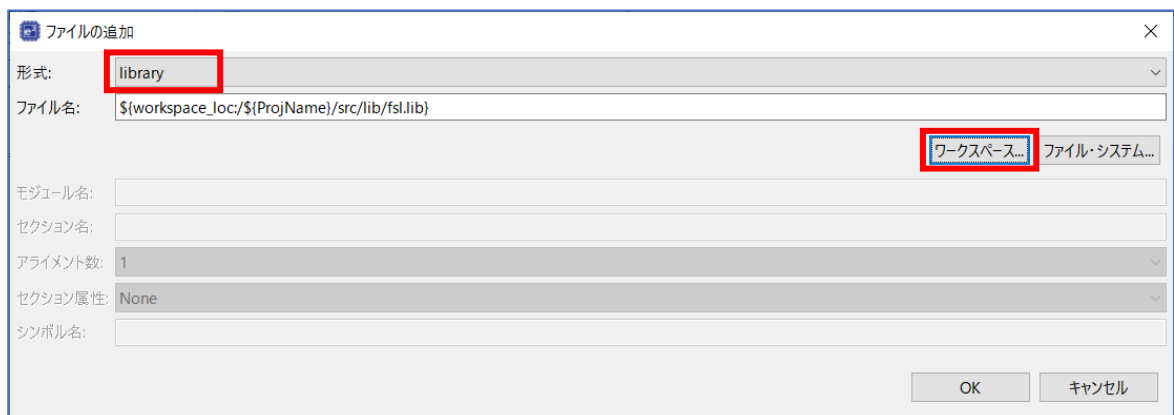


図 7-17(b) ライブラリファイルの登録(CC-RL)

- LLVM コンパイラの場合、e2 studio のプロジェクト・ツリーの[プロジェクト]上でマウスの右クリックで"プロパティ"を選択、"C/C++ビルド" [設定] - "Linker" [Source]で表示された画面の"Additional input files"欄に、ライブラリファイルのパスを登録します。

フラッシュ・セルフ・プログラミング・ライブラリのファイル"libfsl.a"のファイルパスを登録してください。

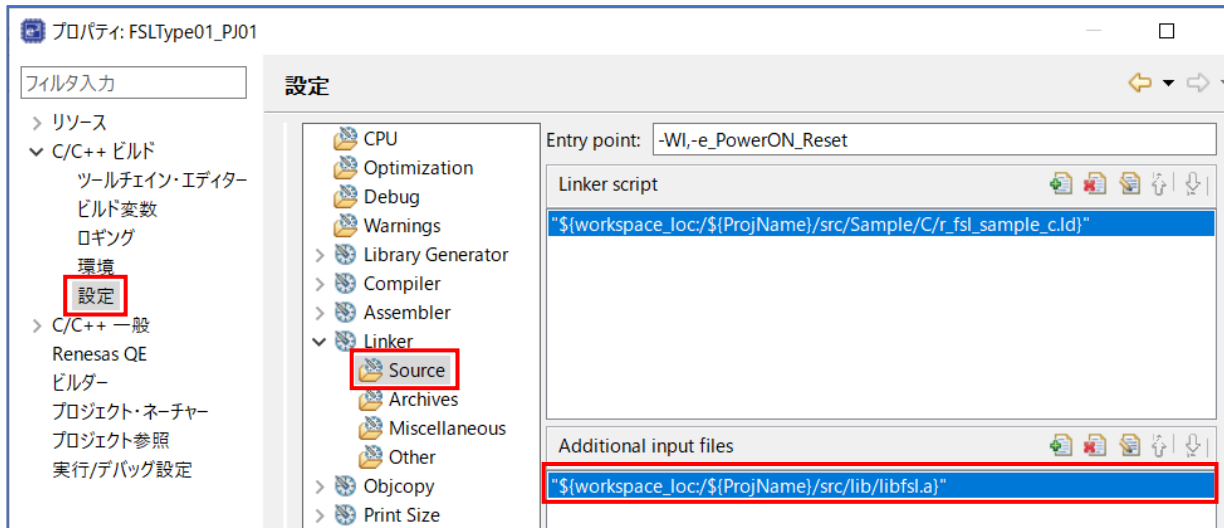


図 7-18 ライブラリファイルの登録(LLVM)

### (3) リンカスクリプトファイルの設定 (LLVM コンパイラ使用時のみ)

LLVM コンパイラの場合、e<sup>2</sup> studio のプロジェクト・ツリーの[プロジェクト]上でマウスの右クリックで"プロパティ"を選択、"C/C++ビルド" [設定] - "Linker" [Source]で表示された画面の"Linker script"欄に、リンカスクリプトファイルのパスを設定します。

ここでは、FSL Type01 用に準備されている"r\_fsl\_sample\_c.Id"ファイルを選択します。

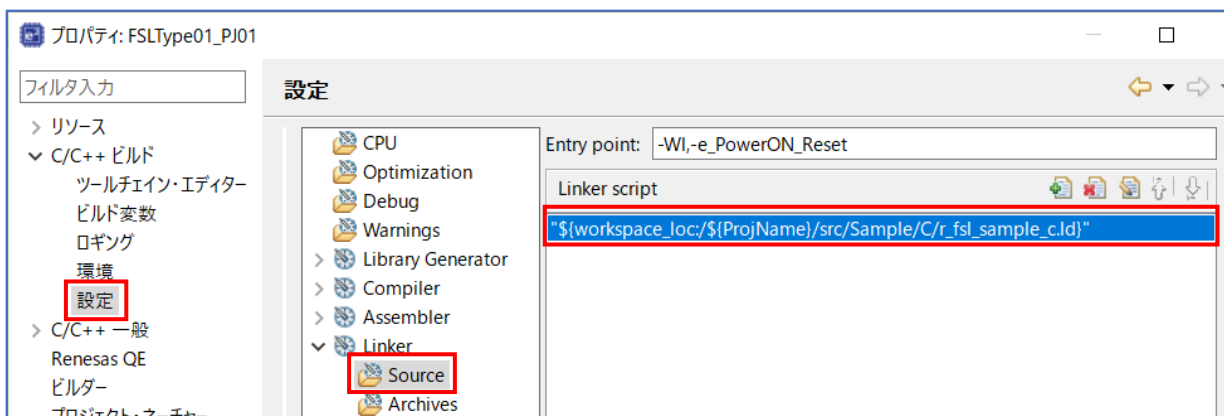


図 7-19 ライブラリファイルの登録(LLVM)

**注.** リンカスクリプトファイル(\*.Id)の記述内容、及び記述方法の詳細については、LLVM のリファレンスマニュアルをご参照ください。

### (4) ビルド

e<sup>2</sup> studio のプロジェクト・ツリーの[プロジェクト]上でマウスの右クリックで"プロジェクトのビルド"を選択してビルドを実行します。

### 7.3.3 アセンブリ言語の場合

#### (1) ソース、インクルードファイルの設定

- ・作成したプロジェクトへの、フラッシュ・セルフ・プログラミング・ライブラリの追加
  - CC-RL では、e<sup>2</sup> studio が出力する"src"フォルダにユーザプログラムのファイル("xxxxxx.asm")と、フラッシュ・セルフ・プログラミング・ライブラリのファイル"fsl.inc", "fsl.lib"を登録します。(図 7-20)

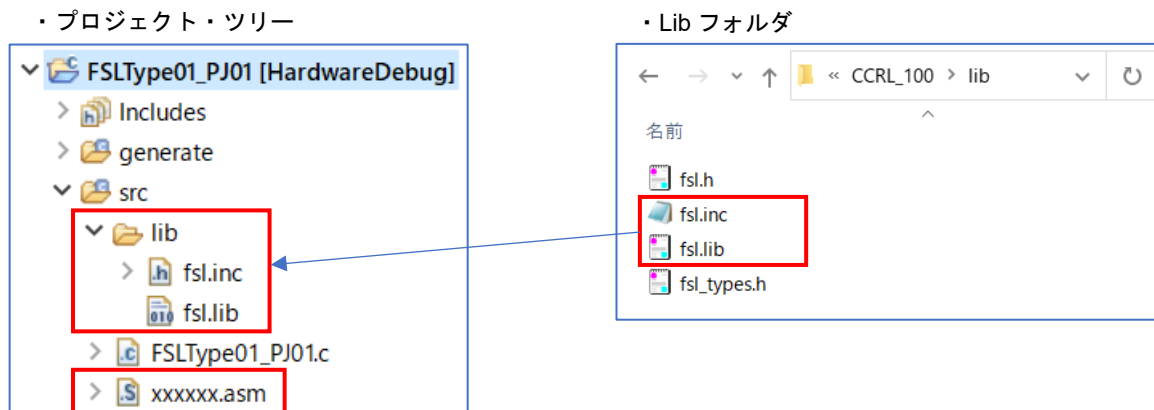


図 7-20 ライブラリファイルの登録(CC-RL)

- ・統合開発環境の機能により自動的に追加されたファイルの除外

作成されたプロジェクトには、自動的に追加されるファイルがあります。これらと同様のファイルは、FSL Type01 の"sample"フォルダ内にも存在するため、ツリーで各ファイルを選択し、各統合開発環境の機能を使用して、プロジェクトから外します。

プロジェクト・ツリー上のファイルをマウス右クリック、"プロパティ"で表示された[設定]画面で、"ビルドからリソースを除外"にチェックを入れ、対象ファイルを除外します(フォルダから削除も可能)。

- CC-RL コンパイラ使用時は、[プロジェクト名]/src フォルダ内の[プロジェクト名].c(ここでは "FSLType01\_PJ01.c")が対象。

## (2) ライブラリファイルの設定

- CC-RL コンパイラの場合、e<sup>2</sup> studio のプロジェクト・ツリーの[プロジェクト]上でマウスの右クリックで"プロパティ"を選択、"C/C++ビルド" [設定] - "Linker" [入力]で表示された画面の"リンクするリロケータブル・ファイル、オブジェクト・ファイル、およびライブラリ・ファイル"の右側にある+ボタンをクリックして表示された"ファイルの追加"ウィンドウで[形式]を"library"に変更してから、ライブラリファイルのパスを登録します。

フラッシュ・セルフ・プログラミング・ライブラリのファイル"fsl.lib"のファイルパスを登録してください。

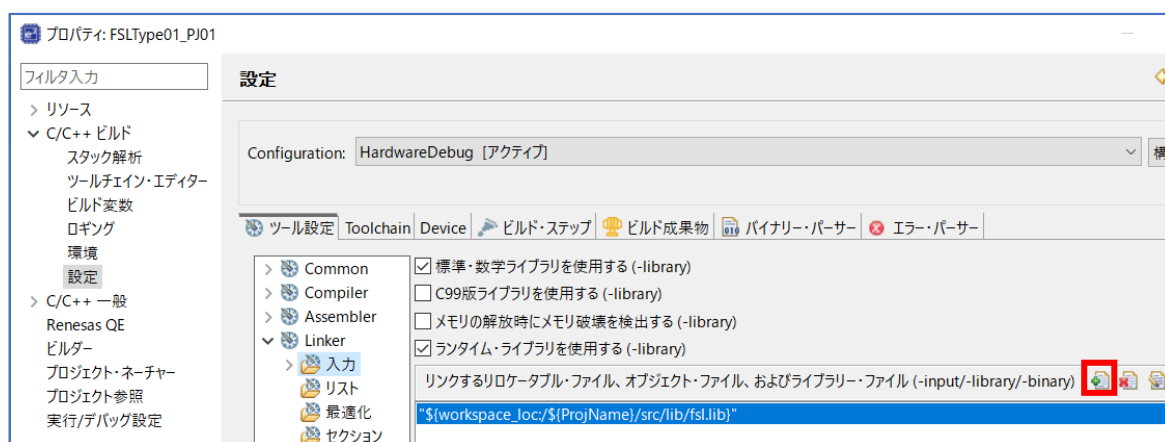


図 7-21(a) ライブラリファイルの登録(CC-RL)

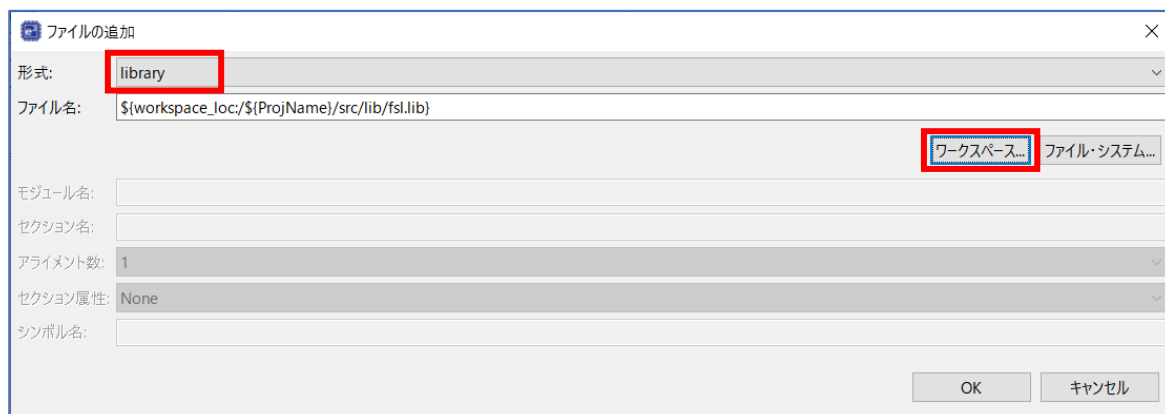


図 7-21(b) ライブラリファイルの登録(CC-RL)

## (3) ビルド

e<sup>2</sup> studio のプロジェクト・ツリーの[プロジェクト]上でマウスの右クリックで"プロジェクトのビルド"を選択してビルドを実行します。

## 7.4 ビルド時の注意事項

### 7.4.1 CA78K0R コンパイラ使用時

#### (1) オンチップ・デバッグ機能使用時

CS+でオンチップ・デバッグ機能を有効にした後、プログラムのビルドを行うと、以下のようなエラーが発生する場合があります。

```
RA78K0R error E3212: Default segment can't allocate to memory - ignored
Segment '??OCDROM' at xxxxxH-200H
```

このエラーは、オンチップ・デバッグ機能で使用するモニター領域(OCDROM)用のセグメント配置ができないために発生していますので、エラーを回避するためには、プロジェクトに組み込んでいるリンクディレクティブファイル(\*.dr)に、以下のような内容を追記し、セグメント配置用の領域を別途用意してください。

```
MEMORY OCD_ROM : ( 0xxxxxH, 00200H )
```

- 備考 1. xxxxx : エラーが発生した箇所の先頭アドレス  
2. 領域名「OCD\_ROM」は参考例としての表記となります。

#### (2) 再リンク機能使用時（フラッシュ領域側）

CS+の再リンク機能を使用し、フラッシュ領域側のプロジェクトにセクション名を指定する宣言のあるファイルを登録後、プログラムのビルドを行うと、以下のようなエラーが発生する場合があります。

```
GC78K0R error E0842: Unrecognized pragma SECTION '@@xxxxx'
```

このエラーは、再リンク機能を使用した場合にフラッシュ領域側のセクション名が通常とは異なるために発生していますので、エラーを回避するためには、指定しているセクション名を以下のように「@@xxxxx」から、「@Exxxxx」へ変更し、フラッシュ領域側のセクション名規則に合わせる必要があります。

```
#pragma section @Exxxxx CNST_DAT
```

- 備考 1. xxxxx : 任意のセクション名の文字列  
2. 変更後のセクション名「CNST\_DAT」は参考例としての表記となります。

## 7.4.2 CC-RL コンパイラ使用時

### (1) オンチップ・デバッグ機能使用時

CS+でオンチップ・デバッグ機能を有効にした後、プログラムのビルドを行うと、以下のようなエラーが発生する場合があります。

```
E0562321:Section ".monitor2" overlaps section "xxxxx"
```

このエラーは、オンチップ・デバッグ機能で使用するモニター領域(OCDROM)用のセクション配置ができないために発生しています。エラーを回避するためには、CS+の[プロジェクト・ツリー]から①[CC-RL(ビルド・ツール)]を選択、右クリックで表示されるプロパティを選択し、表示された②"CC-RLのプロパティ"の③"リンク・オプション"タブを選択します。④"セクション"項目内の⑤"セクションの開始アドレス"で、オンチップ・デバッグモニタ用のセクション配置領域(monitor2 : R5F100LE では、初期アドレス 0xFE00-0xFFFF)に、他のセクションが重複しないよう配置を修正してください。(図 7-22)

セクションの設定の詳細については、CC-RL コンパイラ ユーザーズマニュアルをご確認ください。

備考 1. xxxxx : セクション名

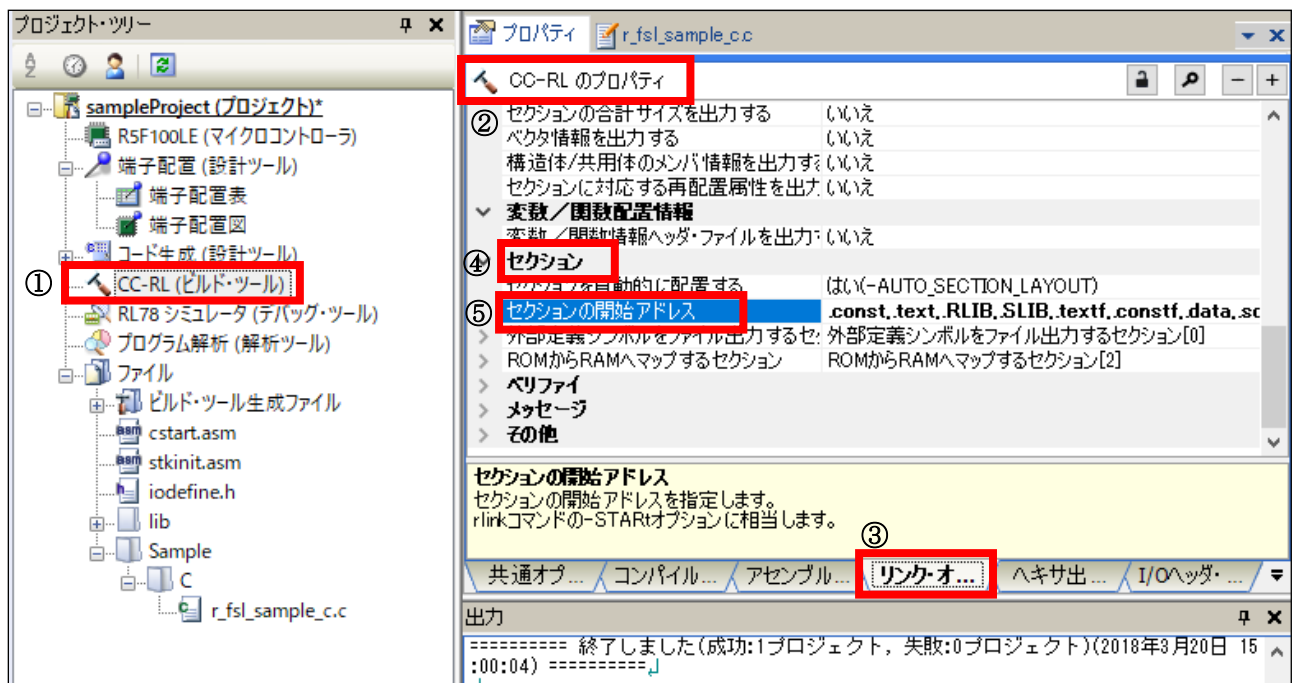


図 7-22 セクション配置の変更

## 第8章 デバッグ方法

IECUBE、またはオンチップ・デバッグ・エミュレータ E1、E2、E2 エミュレータ Lite、及び E20 を使用してデバッグを行う場合につきましては、以下の資料を参照してください。

タイトル
CubeSuite+ 統合開発環境 ユーザーズマニュアル RL78 デバッグ編[CS+ for CA,CX] <sup>注</sup>
CS+ 統合開発環境 ユーザーズマニュアル RL78 デバッグ・ツール編[CS+ for CC] <sup>注</sup>
e <sup>2</sup> studio 統合開発環境 ユーザーズマニュアル 入門ガイド

注. このドキュメントは、弊社HPの「統合開発環境 CS+ (旧 CubeSuite+)」、および「統合開発環境 e<sup>2</sup> studio」のページからダウンロードしてください。

## 第9章 サンプルプログラム

添付のサンプルプログラム( r\_fsl\_sample\_c.c )は、R5F100LEA(RL78/G13)を対象に QB-R5F100LE-TB ボードでフラッシュ・セルフ・プログラミング・ライブラリ Type01 の使用方法を簡単に確認することが可能なように用意しているプログラムです。あくまで参考例となりますので、必ずサンプルプログラム通りに作成する必要があるわけではありません。簡易的な動作確認用のプログラムとしてご使用ください。

- ・ CA78K0R コンパイラ用サンプルプログラムのリンクディレクティブファイル( r\_fsl\_sample\_c.dr )は、サンプルプログラムで使用するスタックやデータバッファ等を配置禁止領域<sup>注1</sup>に配置しないように指定する事を目的としています。サンプルプログラムを使用する場合は、こちらのファイルも一緒に組み込んでください。<sup>注2,3</sup>
- ・ CC-RL コンパイラでは、CS+、または e<sup>2</sup> studio の画面上で、サンプルプログラムで使用するスタックやデータバッファ等を配置禁止領域<sup>注1</sup>に配置しないように、[セクション]配置を設定する必要があります。<sup>注3</sup>
- ・ LLVM コンパイラ用リンクスクリプトファイル( r\_fsl\_sample\_c.ld )は、サンプルプログラムで使用するスタックやデータバッファ等を配置禁止領域<sup>注1</sup>に配置しないように指定する事を目的としています。サンプルプログラムを使用する場合は、こちらのファイルも一緒に組み込んでください。<sup>注3</sup>

注 1. 詳細については、ユーザズマニュアルの「2.2 ソフトウェア環境」を参照してください。

2. 添付のリンクディレクティブファイルは RAM 領域サイズが 512 バイトとなっていますが、RAM が 2k バイトより大きい製品でも、サンプルプログラム( r\_fsl\_sample\_c.c )を使用する場合に、定義されている領域の変更をする必要はありません。そのままビルド等を行う事ができます。
3. ご使用の環境やプログラムの変更によっては使用中のデータが意図しない領域へ配置される場合があります。実行モジュール作成後はマップファイルを確認し、プログラムやデータの配置状態を必ず確認してください。また、各コードやデータの定義方法や配置条件等については、使用するコンパイラのユーザズマニュアルを参照してください。

### 9.1 サンプルプログラムの初期設定

サンプルプログラムは以下の初期設定で動作します。変更が必要な場合は、サンプルプログラムを修正してください。

- ・ CPU の動作周波数 : 高速オンチップ・オシレータ 32MHz
- ・ 電圧モード : ハイスピード・モード

## 9.2 オプション・バイトとオンチップ・デバッグの設定について

### (1) CS+でCA78K0R、またはCC-RLコンパイラを使用する場合

オンチップ・デバッグを行う場合は「オンチップ・デバッグの許可/禁止をリンク・オプションで設定する」を「はい」に設定し、「オンチップ・デバッグ・オプション・バイト制御値」を「84」に指定してください。CC-RLコンパイラ用では、「デバッグ・モニタ領域を設定する」を「はい」に指定してください。また、サンプルプログラムは高速オンチップ・オシレータを32MHzに設定する事で正常に動作します。リンク・オプションから、「ユーザ・オプション・バイトを設定する」を「はい」に指定後、「ユーザ・オプション・バイト値」を「xxxxE8」に指定し、高速オンチップ・オシレータを32MHzに設定してください。

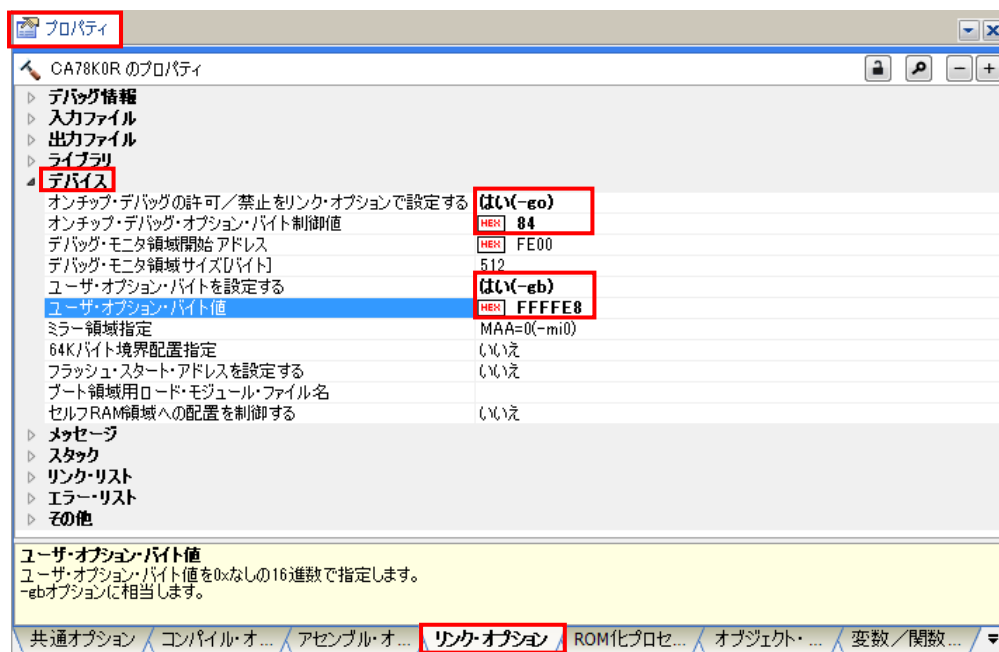


図 9-1(a) オプション・バイトの設定(CA78K0Rコンパイラ)

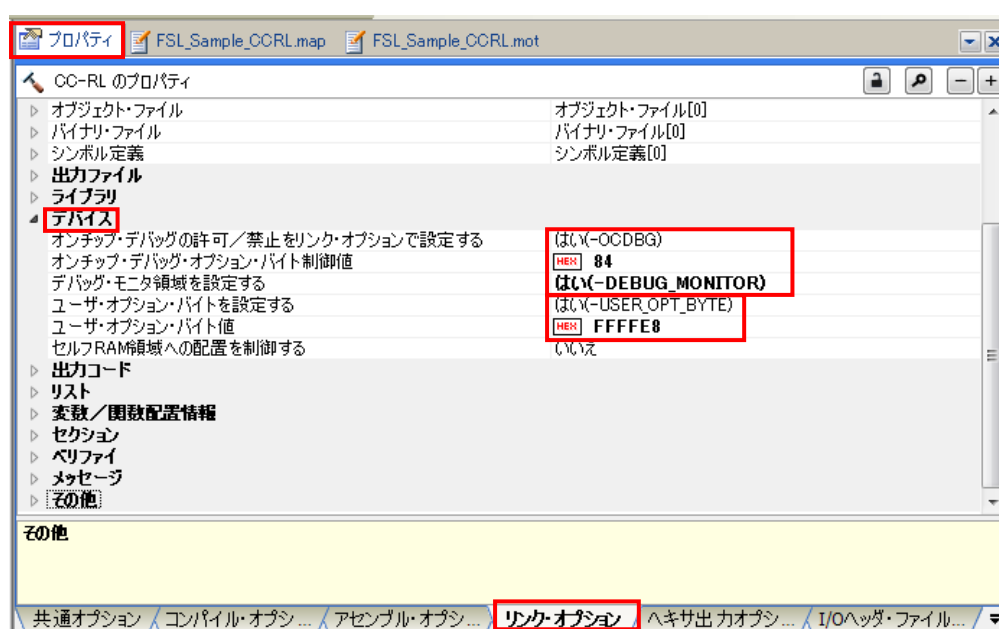


図 9-1(b) オプション・バイトの設定(CC-RLコンパイラ)

## (2) e<sup>2</sup> studioでCC-RLコンパイラを使用する場合

"C/C++ビルド" [設定] - "Linker" [デバイス] で表示された画面でデバイス項目の設定をします。

オンチップ・デバッグを行う場合は、[オプション・バイト領域のオンチップ・デバッグ・オプション・バイト]に値を設定する(-ocdbg)]にチェックをいれ、「オンチップ・デバッグ制御値」に「84」を指定してください。

また、サンプルプログラムは高速オンチップ・オシレータを32MHzに設定する事で正常に動作します。

[オプション・バイト領域のユーザ・オプション・バイト値を設定する(-user\_opt\_byte)]にチェックをいれ、[ユーザ・オプション・バイト値(-user\_opt\_byte=<value>)]に「xxxxE8」を指定し、高速オンチップ・オシレータを32MHzに設定してください。

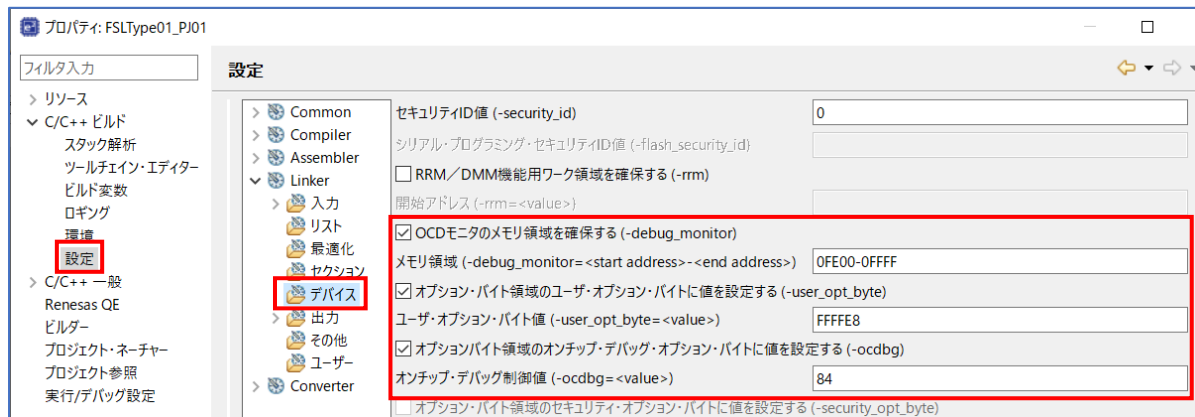


図 9-2 オプション・バイトの設定(LLVMコンパイラ)

## (3) e<sup>2</sup> studioでLLVMコンパイラを使用する場合

デバイス項目の設定は、e2 studio から出力される"vects.c"ファイルで設定します。

対象ファイルパス："プロジェクトフォルダ"¥generate¥vects.c

サンプルプログラムは、高速オンチップ・オシレータを32MHzに設定する事で正常に動作します。そのため、"vects.c"ファイルの"Option\_Bytes"にユーザ・オプション・バイト値「xxxxe8」とオンチップ・デバッグ・オプション・バイト値を次のように設定します。

[RL78/G13の設定例]

"0xff, 0xff, 0xe8, 0x84" (WDT動作許可, LVDオフ, HSモード/32MHz/, オンチップ・デバッグ動作許可)

```
const unsigned char Option_Bytes[] __attribute__((section(".option_bytes"))) = {
    0xff, 0xff, 0xe8, 0x84
};
```

注. オンチップ・デバッグを実施することを前提とした設定例です。

対象デバイスのユーザーズマニュアルで「オプション・バイト」の章の「ユーザ・オプション・バイト」「オンチップ・デバッグ・オプション・バイト」の内容をご確認いただき、設定値を書き込んでください。

### 9.3 C 言語用サンプルプログラムのコンパイルスイッチ

サンプルプログラムには以下のようなコンパイルスイッチが用意されています。コンパイルスイッチは QB-R5F100LE-TB ボードで動作確認用の LED を点灯させるために使用しています。使用する場合は、対象となる CPU ボードの「#define」宣言が有効になるように、[#if 0]を[#if 1]に変更してください。

```
/* **** */
/* サンプル・プログラム・プログラムスイッチ用シンボル */
/* **** */
/* QB-R5F100LE-TB 単体使用時 ON 可能 */
#if 0 ← QB-R5F100LE-TB の場合、#if 1 に設定可能
#define __QB_R5F100LE_TB__

/* その他ボード */
#else
#define __NON_TARGET__
#endif
```

## 9.4 内蔵 RAM 領域の定義

### 9.4.1 CA78K0R コンパイラ使用時

CA78K0R コンパイラ使用時の初期状態では、内蔵 RAM 領域については全域が"RAM"という名称の領域として自動的に定義されています。特にリンクディレクティブファイル等で指定しない限り、スタック等がこの領域に配置<sup>注</sup>される事になりますが、この場合、フラッシュ・ライブラリの使用禁止領域(セルフ RAM、FFE20H-FFEFFH の領域)にスタックやデータバッファ等が配置されてしまい、プログラムが正常に動作できなくなることがあります。

添付されているサンプルプログラム用リンクディレクティブファイルでは、解決策の一つとして"RAM"名の領域をフラッシュ・ライブラリの使用禁止領域を含めないように再定義し、スタック等が使用禁止領域に配置されないようにしています。

```
MEMORY RAM      : (0FF300H, 000B20H)
```

上記は、"RAM"名の領域を、FF300H から B20H バイトのサイズの領域 (FF300H-FFE1FH) <sup>注</sup>に再定義し、フラッシュ・ライブラリの使用禁止領域が"RAM"名の領域に含まれないようにしています。

ただし、この設定だけでは「FFE20H-FFEFFH」の領域が他の用途でも使用できなくなるため、別途以下の定義を追加する必要があります。この領域の名称に関しては特に制限等はありません。

```
MEMORY SADDR_RAM: (0FFE20H, 0000E0H)
```

また、セルフ RAM 領域がある場合は、その範囲を以下のように"SELFRAM"名の領域として定義する事により、この領域に対し、自動的に変数等が配置されないように制限する事が可能です。

```
MEMORY SELFRAM  : (0FEF00H, 000400H)
```

以下に、RL78/G13(RAM 4KB/ROM 64KB 製品)の場合の設定例を記載します。

```

; -----
; Define new memory entry for Self-RAM
; -----
MEMORY SELFRAM  : ( 0FEF00H, 000400H ) ← セルフ RAM 領域の定義
; -----
; Redefined default data segment RAM
; -----
MEMORY RAM      : ( 0FF300H, 000B20H ) ← 標準的に使用する RAM 領域の定義
; -----
; Define new memory entry for saddr area
; -----
MEMORY RAM_SADDR : ( 0FFE20H, 0000E0H ) ← FFE20H-FFEFFH 領域の定義

```

注. CA78K0Rのリンクでは、配置先が指定されないデータ(セグメント・タイプDSEGおよびBSEG)はそのデータの再配置属性に従い、内蔵RAM領域に配置されます。そのため、状況によっては特定のデータが"RAM"名の領域に配置されない場合があります。

各データの定義や配置方法等の詳細についてはCS+のユーザーズマニュアルを参照してください。

また、ビルド時に生成されるマップファイル(\*.map)を必ず参照し、各データの配置状態を確認してください。

## 9.4.2 CC-RL コンパイラ使用時

### (1) インクルード・パスの追加

CS+、e<sup>2</sup> studio は、初期状態ではインクルード・パスは設定されていません。フラッシュ・セルフ・プログラミング・ライブラリで使用するヘッダファイルのインクルード・パスを追加する必要があります。フラッシュ・セルフ・プログラミング・ライブラリで使用するヘッダファイルは、"fsl.h"、"fsl\_types.h"、及び CS+、または e<sup>2</sup> studio が自動生成する"iodefine.h"です。

- CS+では、[ コンパイル・オプション ]-[ プリプロセス ]-[ 追加のインクルード・パス ]で、各ファイルが存在するインクルード・パスを追加してください。

- e<sup>2</sup> studio では"プロパティ"ウインドウで、"C/C++ビルド" [設定] - "Compiler" [ソース] で表示された画面の"インクルード・ファイルを検索するフォルダ(-I)"の欄に、各ファイルが存在するインクルード・パスを追加してください。

### (2) セクションの定義

CC-RL コンパイラ用 CS+、または e<sup>2</sup> studio 使用時は、ROM 領域、RAM 領域で使用する各セクションを定義する必要があります。

- CS+でのセクションの定義は、CS+の画面の[ リンク・オプション ]-[ セクション ]で指定することができます。[セクションを自動的に配置する]を"いいえ"に設定し、[セクションの開始アドレス]の設定画面を開き、ROM 領域にフラッシュ・セルフ・プログラミング・ライブラリのセクション(ここでは、本サンプルプログラム動作に必要なセクション、FSL\_FCD, FSL\_RCD, FSL\_BCD, FSL\_BECD)を追加します。(図 9-3)

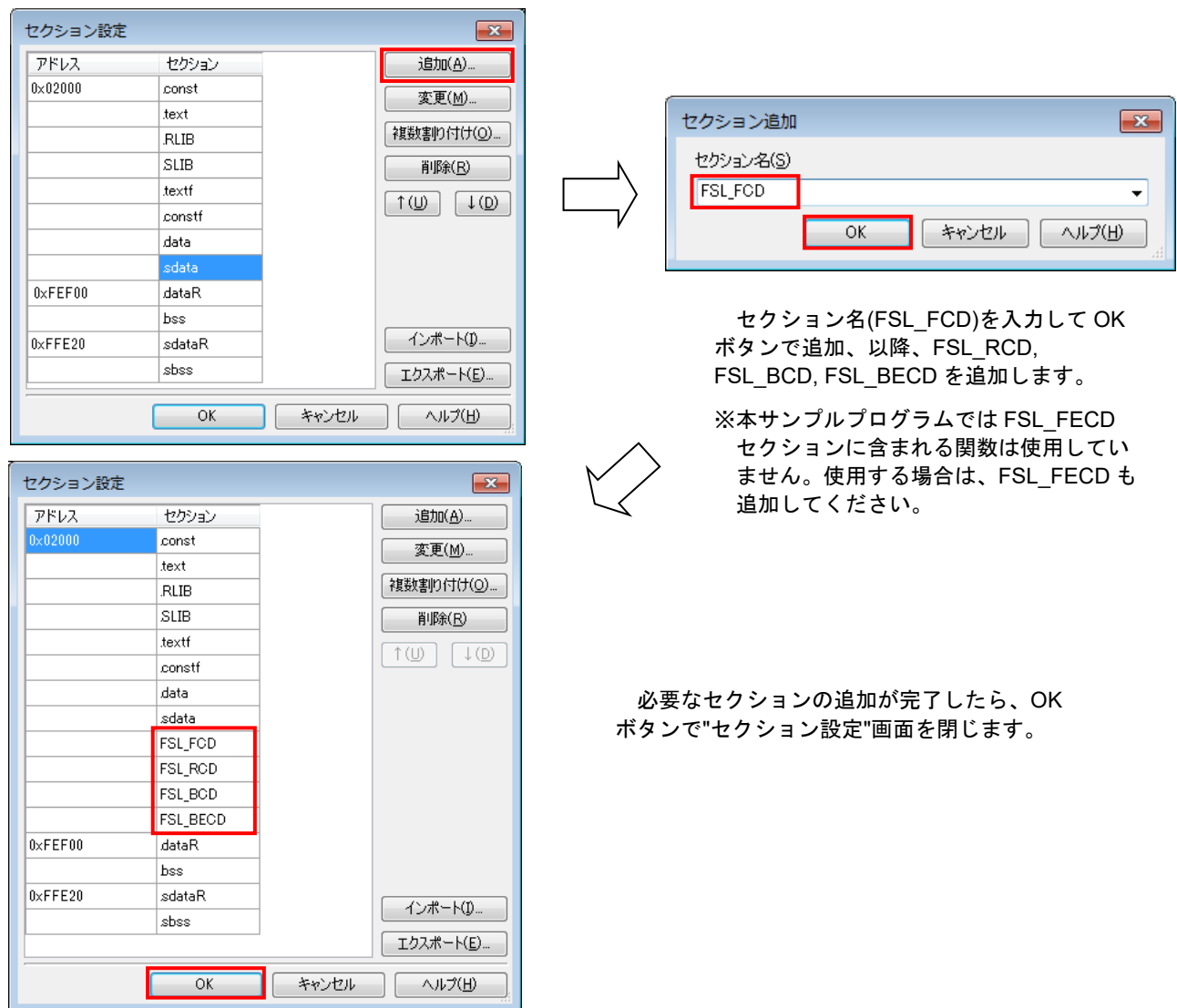


図 9-3 CS+使用時のフラッシュ・セルフ・プログラミング・ライブラリのセクション設定例(ROM領域)

- e<sup>2</sup> studio でのセクション項目の設定は、"プロパティ"ウインドウで設定します。"C/C++ビルド" [設定] - "Linker" [セクション] で表示した画面でセクション項目を設定します。[デバイス・ファイルの情報からセクションを自動的に配置する (-auto\_section\_layout)]のチェックを一度外し、[セクション(-start)]の最も右の"..."ボタンで、"セクションビューアー"画面を開き、ROM 領域にフラッシュ・セルフ・プログラミング・ライブラリのセクション(ここでは、本サンプルプログラム動作に必要なセクション、FSL\_FCD, FSL\_RCD, FSL\_BCD, FSL\_BECD)を追加します。(図 9-4)

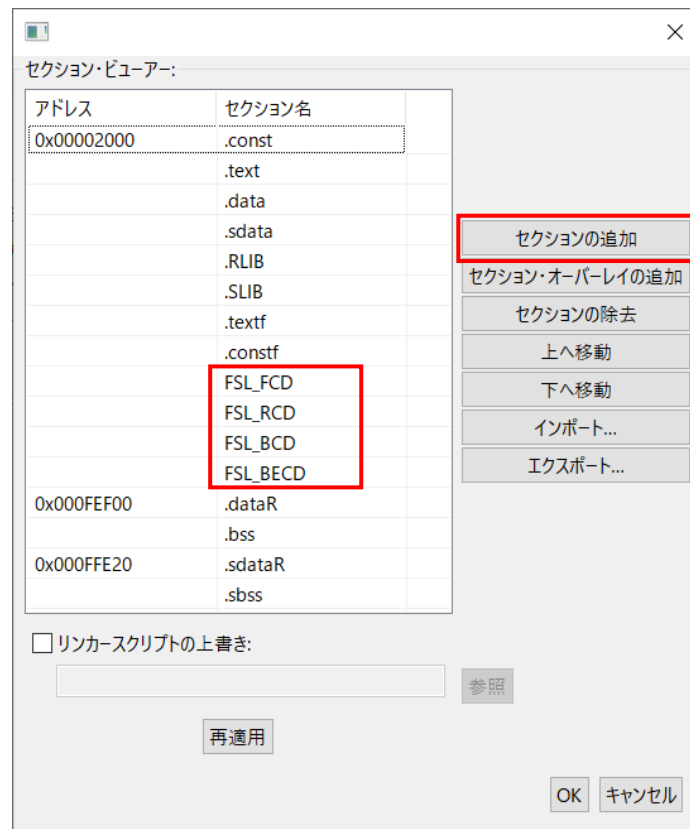


図 9-4 e<sup>2</sup> studio 使用時のフラッシュ・セルフ・プログラミング・ライブラリのセクション設定例(ROM 領域)

注. 本サンプルプログラムでは FSL\_FECD セクションに含まれる関数は使用していません。使用する場合は、FSL\_FECD も追加してください。

### (3) セルフ RAM 領域の確保

セクション設定の初期状態では、ユーザ RAM 領域が内蔵 RAM 領域の先頭(サンプルプログラム対象の R5F100LEA では 0xFE00)から確保されています。しかし、R5F100LEA では、フラッシュ・セルフ・プログラミング・ライブラリがセルフ RAM 領域として 0xFE00-0xFF2FF を使用する為、この領域を避けてユーザ RAM 領域を設定する必要があります。ここでは、アドレス 0xFE00 から設定されているユーザデータの先頭アドレスを 0xFF300 に変更する例を示します。

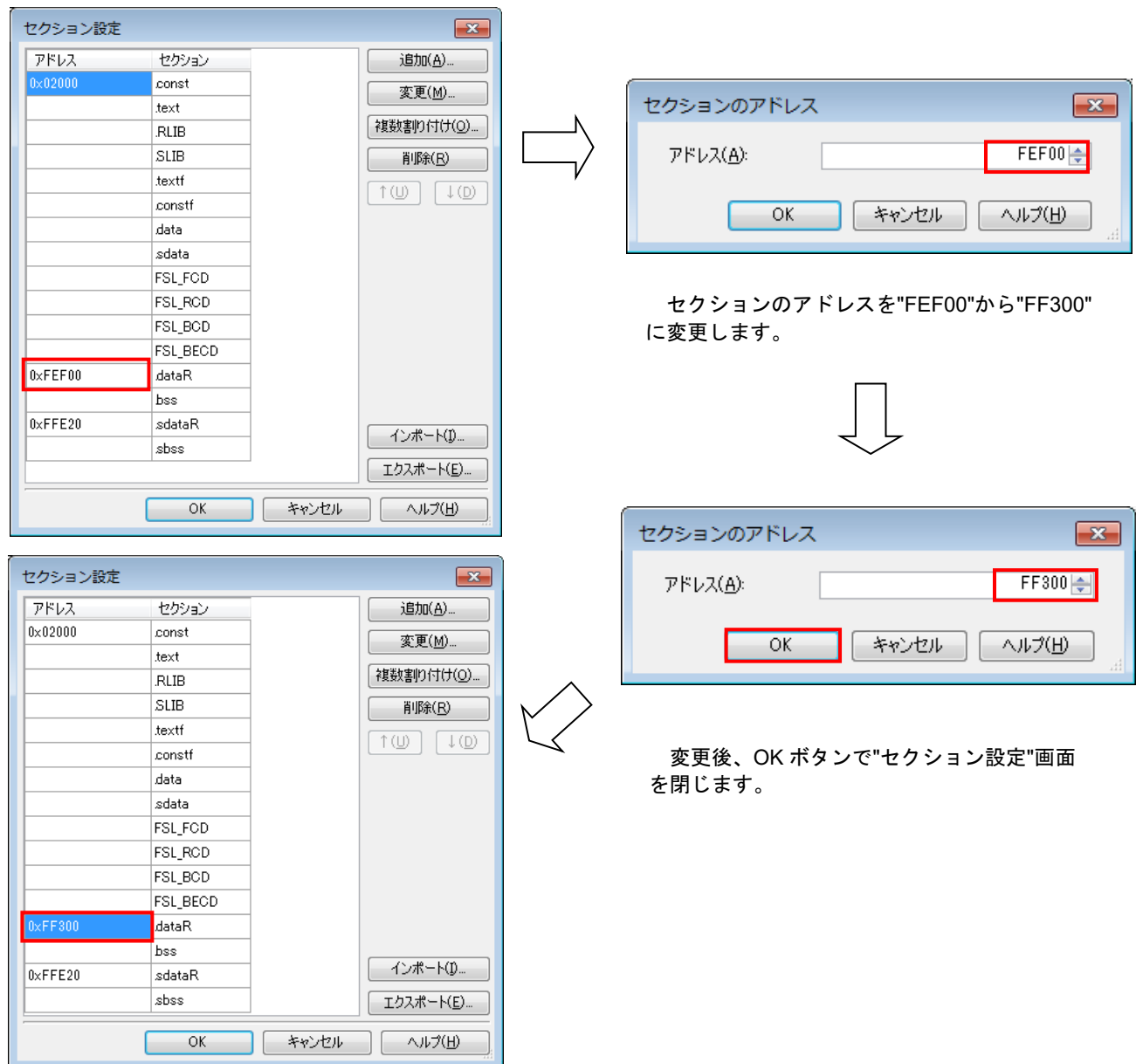


図 9-5 CS+使用時のユーザRAM領域範囲の変更例(RAM領域)

図 9-6 e<sup>2</sup> studio 使用時のユーザ RAM 領域範囲の変更例(RAM 領域)

注意) セクションの設定で[セクションを自動的に配置する]を一度"いいえ"にして、ユーザ RAM 領域範囲の変更をした後に、"はい"に設定することで、ユーザが定義したセクションも含め、再自動配置されますが、自動配置では、ユーザが定義していない領域に対してもセクションを配置する可能性があり、意図していなかった領域にデータが配置される可能性があります。必ず、フラッシュ・セルフ・プログラミング・ライブラリで使用するソフトウェアリソース(特に RAM 上データ)が、配置可能領域に配置されていることをマップファイル(\*.map)等で確認してください。

### 9.4.3 LLVM コンパイラ使用時

#### (1) インクルード・パスの追加

e<sup>2</sup> studio は、初期状態ではインクルード・パスは設定されていません。フラッシュ・セルフ・プログラミング・ライブラリで使用するヘッダファイルのインクルード・パスを追加する必要があります。フラッシュ・セルフ・プログラミング・ライブラリで使用するヘッダファイルは、"fsl.h"、"fsl\_types.h"、及び e<sup>2</sup> studio が自動生成する"iodefine.h"、"iodefine\_ext.h"です。

e<sup>2</sup> studio では、"プロパティ"ウィンドウで、"C/C++ビルド" [設定] - "Compiler" [includes] で表示された画面の"Include file directories (-I)"の欄に、各ファイルが存在するインクルード・パスを追加してください。

## (2) セルフ RAM 領域の確保

LLVM コンパイラでは、ビルドで実行するリンク設定をリンクスクリプトファイル(\*.ld)に記述します。

e<sup>2</sup> studio が出力するリンクスクリプトファイル(linker\_script.ld)では、内蔵 RAM 領域については全域が"RAM"という名称の領域として定義され、フラッシュ・セルフ・プログラミング・ライブラリで使用するソフトウェアリソースが必要なデバイスでは、"SELFRAM"という名称の領域として定義されています。

サンプルプログラム用に添付されているリンクスクリプトファイル(r\_fsl\_sample\_c.ld)では、"RAM"名の領域をフラッシュ・セルフ・プログラミング・ライブラリの使用禁止領域を含めないように再定義し、SELFRAM(セルフ RAM)領域を定義し確保することでユーザデータやスタック等がセルフ RAM 領域に配置されないようにしています。また、FSL Type01 の動作に必要なセクション、FSL\_FCD, FSL\_RCD, FSL\_BCD, FSL\_BECD が ROM に配置されるよう記載されています。

注. サンプルプログラムで提供している r\_fsl\_sample\_c.ld は、R5F100LE を使用することを前提に用意されています。そのほかのデバイスをご使用の場合は、セルフ RAM リストをご確認いただき、デバイスにあわせて修正を行ってください。

リンクスクリプトファイル(\*.ld)の記述内容、及び記述方法の詳細については、LLVM のリファレンスマニュアルをご参照ください。

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。

## 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。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、お客様の責任において、お客様の機器・システムを設計ください。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. 当社製品または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. 当社製品を組み込んだ製品の輸出入、製造、販売、利用、配布その他の行為を行うにあたり、第三者保有の技術の利用に関するライセンスが必要となる場合、当該ライセンス取得の判断および取得はお客様の責任において行ってください。
  5. 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。
  6. 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。
  7. あらゆる半導体製品は、外部攻撃からの安全性を 100%保証されているわけではありません。当社ハードウェア/ソフトウェア製品にはセキュリティ対策が組み込まれているものもありますが、これによって、当社は、セキュリティ脆弱性または侵害（当社製品または当社製品が使用されているシステムに対する不正アクセス・不正使用を含みますが、これに限りません。）から生じる責任を負うものではありません。当社は、当社製品または当社製品が使用されたあらゆるシステムが、不正な改変、攻撃、ウイルス、干渉、ハッキング、データの破壊または窃盗その他の不正な侵入行為（「脆弱性問題」といいます。）によって影響を受けないことを保証しません。当社は、脆弱性問題に起因またはこれに関連して生じた損害について、一切責任を負いません。また、法令において認められる限りにおいて、本資料および当社ハードウェア/ソフトウェア製品について、商品性および特定目的との合致に関する保証ならびに第三者の権利を侵害しないことの保証を含め、明示または黙示のいかなる保証も行いません。
  8. 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。
  9. 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。
  10. 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、当社は、一切その責任を負いません。
  11. 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。
  12. 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なります。
  13. 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。
  14. 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.5.0-1 2020.10)

## 本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

[www.renesas.com](http://www.renesas.com)

## 商標について

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の商標です。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。

## お問合せ窓口

弊社の製品や技術、ドキュメントの最新情報、最寄の営業お問合せ窓口に関する情報などは、弊社ウェブサイトをご覧ください。

[www.renesas.com/contact/](http://www.renesas.com/contact/)