EPSON
EXCEED YOUR VISION

S1C31 Family Application Note

S1C31 Family
自己書き換えライブラリ
マニュアル

arm

セイコーエプソン株式会社

Rev.1.0

評価ボード・キット、開発ツールご使用上の注意事項

1. 本評価ボード・キット、開発ツールは、お客様での技術的評価、動作の確認および開発のみに用いられることを想定し設計されています。それらの技術評価・開発等の目的以外には使用しないで下さい。本品は、完成品に対する設計品質に適合していません。

2. 本評価ボード・キット、開発ツールは、電子エンジニア向けであり、消費者向け製品ではありません。お客様において、適切な使用と安全に配慮願います。弊社は、本品を用いることで発生する損害や火災に対し、いかなる責も負いかねます。通常の使用においても、異常がある場合は使用を中止して下さい。

3. 本評価ボード・キット、開発ツールに用いられる部品は、予告無く変更されることがあります。

本資料のご使用につきましては、次の点にご留意願います。

本資料の内容については、予告無く変更することがあります。

1. 本資料の一部、または全部を弊社に無断で転載、または、複製など他の目的に使用することは堅くお断りいたします。

2. 本資料に掲載される応用回路、プログラム、使用方法等はあくまでも参考情報であり、これらに起因する第三者の知的財産権およびその他の権利侵害あるいは損害の発生に対し、弊社はいかなる保証を行うものではありません。また、本資料によって第三者または弊社の知的財産権およびその他の権利の実施権の許諾を行うものではありません。

3. 特性値の数値の大小は、数直線上の大小関係で表しています。

4. 製品および弊社が提供する技術を輸出等するにあたっては「外国為替および外国貿易法」を遵守し、当該法令の定める手続きが必要です。大量破壊兵器の開発等およびその他の軍事用途に使用する目的をもって製品および弊社が提供する技術を費消、再販売または輸出等しないでください。

5. 本資料に掲載されている製品は、生命維持装置その他、きわめて高い信頼性が要求される用途を前提としていません。よって、弊社は本（当該）製品をこれらの用途に用いた場合のいかなる責任についても負いかねます。

6. 本資料に掲載されている会社名、商品名は、各社の商標または登録商標です。
Arm, Cortex, Keil および µVision は、Arm Limited(またはその子会社)の US またはその他の国における登録商標です。IAR Systems, IAR Embedded Workbench, C-SPY, I-jet, IAR および IAR システムズのロゴタイプは、IAR Systems AB が所有権を有する商標または登録商標です。SEGGER および J-Link は、SEGGER Microcontroller GmbH & Co. KG の商標または登録商標です。All rights reserved.
"Reproduced with permission from Arm Limited. Copyright © Arm Limited"

目　次

1. 概要 2
1.1 動作環境 2
1.2 使用上の注意事項 2
2. ライブラリの構成 3
2.1 フォルダ構成 3
2.2 ライブラリ機能 4
3. ライブラリの使い方 5
3.1 アプリケーションプログラムへの適用方法 5
3.2 内蔵 RAM 使用量 6
3.3 ライブラリ使用上の注意 6
3.4 サンプルソフトウェア 7
4. ライブラリ仕様 8
4.1 ライブラリ関数詳細 8
4.2 エラーコード定義 10
Appendix 11
A. ライブラリをプロジェクトへ組み込む方法(IAR EAWRM) 11
B. ライブラリをプロジェクトへ組み込む方法(MDK-ARM) 14
改訂履歴表 17

1. 概要

S1C31 自己書き換えライブラリは、アプリケーションプログラムから、対象機種の内蔵フラッシュメモリ内のプログラムコードやデータを書き換えるためのプログラムです。このライブラリをアプリケーションプログラムにリンクし、関数コールすることで、フラッシュメモリの消去および書き込み処理を実現します。

本ライブラリおよびサンプルソフトウェアは、S1C31xxx 周辺回路サンプルソフトウェアパッケージに同梱されています。S1C31xxx 周辺回路サンプルソフトウェアパッケージは、弊社 Web より入手可能です。

また、本マニュアルに加えて、“S1C31xxx テクニカルマニュアル”も併せてご参照ください。

1.1 動作環境

サンプルソフトウェアの書き込み時およびデバッグ時には、以下が必要です。

- 評価ボード
  - S1C31 シリーズ搭載 S5U1C31xxxTx 評価ボード
- デバッグプローブ *1*2
  - IAR Systems 社製 I-jet または SEGGER 社製 J-Link
- 統合開発環境
  - IAR Embedded Workbench for ARM®（IAR EWARM）または MDK-ARM®（µVision）
- S1C31SetupTool パッケージ
  - フラッシュローダおよび構成ファイル（.svd など）
- S1C31xxx 周辺回路サンプルソフトウェアパッケージ

*1:サンプルソフトウェアからライブラリの関数コールには、デバッグプローブは不要となります。

*2: I-jet は IAR EWARM でのみ使用可能です。J-Link は IAR EWARM と MDK-ARM の両方で使用可能です。

上記の詳細については、それぞれに付属するマニュアルをご参照ください。

1.2 使用上の注意事項

S1C31 自己書き換えライブラリおよびサンプルソフトウェアは、参考資料です。これらに起因する不具合が発生した場合、弊社は如何なる責任についても負いません。製品上でお使いになる場合には、十分な動作検証を実施してください。

本マニュアルは、S1C31 シリーズの各機種に用意している自己書き換えライブラリ共通の資料です。機種ごとに異なる仕様（セクタ情報、RAM 使用量等）については、S1C31xxx 周辺回路サンプルソフトウェアパッケージに同梱しているリードミーをご参照ください。

2. ライブラリの構成

2.1 フォルダ構成

S1C31xxx 周辺回路サンプルソフトウェアパッケージに含まれる S1C31 自己書き換えライブラリおよびサンプルソフトウェア、関連プログラムの構成は以下の通りです。

```
S1C31xxxSamplePKG_very_yy.zip
[S1C31xxxSamplePKG_very_yy]
    |- [Licenses]
    |- [Drivers] : ドライバ群
    |     |- [board] : 評価ボード関連ドライバ
    |     |- [CMSIS] : CMSIS ドライバ
    |     |     |- [Device]
    |     |     |     |- [S1C31xxx]
    |     |     |           |- [Include]
    |     |     |           |     |-S1C31xxx.h : CMSIS 周辺回路アクセスレイヤヘッダファイル
    |     |     |           |     |- …
    |     |     |           |- [Source]
    |     |     |                 |- [ARM]
    |     |     |                 |- [IAR]
    |     |     |                 |     |- startup_S1C31xxx.s : CMSIS スタートアッププログラム
    |     |     |                 |- system_S1C31xxx.c : CMSIS 周辺回路アクセスレイヤプログラム
    |     |     |- [Driver]
    |     |     |     |- [Include]
    |     |     |     |     |- Driver_Common.h :共通ドライバ定義
    |     |     |     |     |- Driver_Flash.h : CMSIS 自己書き換えライブラリドライバ定義
    |     |     |     |     |- …
    |     |     |     |- [Source]
    |     |     |- [SVD]
    |     |- [sePeripheralLibrary] : 周辺回路ライブラリ
    |
    |- [Middlewares] : ミドルウェア群
    |     |- [seFlashLibrary] : 自己書き換えライブラリ
    |     |     |- [Device]
    |     |     |     |- [S1C31xxx]
    |     |     |           |- seFlashLibraryS1C31xxx.a : IAR EAWRM 用ライブラリ
    |     |     |           |- seFlashLibraryS1C31xxx.lib : MDK-ARM 用ライブラリ
    |     |     |- flashLibraryForS1c31xxx_readme_e.txt : リードミー
    |     |     |- flashLibraryForS1c31xxx_readme_j.txt
    |     |- …
    |- [Projects] : サンプルソフトウェア群
    |     |- [Applications] : 各種アプリケーションソフトウェア
    |     |     |- [FLASH] : 自己書き換えライブラリ用サンプルソフトウェア
    |     |     |     |- [ARM] : MDK-ARM 用プロジェクト
    |     |     |     |- [IAR] : IAR EWARM 用プロジェクト
    |     |     |     |- main.c
    |     |     |- …
    |     |- …
    |
    README_e.txt
    README_j.txt
```

図 2.1.1 S1C31xxx サンプルソフトウェアパッケージの構成

2.2 ライブラリ機能

本ライブラリが提供する関数は、Drivers¥CMSIS¥Driver¥Include¥Driver_Flash.h で定義をしています。本ライブラリが提供する関数は、以下の通りです。

表 2.2.1 S1C31 自己書き換えライブラリが提供する関数

関数名	機能概要
int32_t Initialize (ARM_Flash_SignalEvent_t cb_event)	本ライブラリの初期化
int32_t Uninitialize (void)	本ライブラリの初期化前の設定に戻す
int32_t EraseSector (uint32_t addr)	内蔵フラッシュメモリを消去
int32_t ProgramData (uint32_t addr, const void *data, uint32_t cnt)	内蔵フラッシュメモリを書き込む
int32_t ReadData (uint32_t addr, unsigned char *data, int32_t cnt)	内蔵フラッシュメモリを読む込む
ARM_DRIVER_VERSION GetVersion (void)	本ライブラリバージョンの取得
ARM_FLASH_INFO * GetInfo (void)	内蔵フラッシュメモリの情報取得

ベリファイ機能については、ProgramData 関数および EraseSector 関数にも組み込まれています。

3. ライブラリの使い方

S1C31 自己書き換えライブラリおよびサンプルソフトウェアの使用方法について説明します。

3.1 アプリケーションプログラムへの適用方法

アプリケーションプログラム上で本ライブラリを使用する方法を説明します。尚、本ライブラリをアプリケーションプログラムのプロジェクトに組み込む方法については“Appendix x. ライブラリをプロジェクトへ組み込む方法”ご参照ください。

1. ヘッダファイル宣言

本ライブラリを使用するソースファイルに、“Driver_Flash.h”をインクルード宣言します。

```
/* include */
#include <stdio.h>
#include <string.h>
#include "Driver_Flash.h"
```

2. 関数の追加

本ライブラリを使用するソースファイルに、ライブラリが提供する関数を追加します。関数の仕様については“4 章 ライブラリ仕様”をご参照ください。

```
extern   ARM_DRIVER_FLASH Driver_Flash;
…
int main(void)
{
   unsigned char compbuf[16];

   //disable Interrupt
     asm("CPSID i");

   ARM_FLASH_INFO *Info = Driver_Flash.GetInfo();

   Driver_Flash.GetVersion();

   //Initialize
   Driver_Flash.Initialize(NULL);

   //Erase at 0x1D000
   if (Driver_Flash.EraseSector(0x1D000)==ARM_DRIVER_OK)
   {
      printf("Erase: OK¥n");
      //Write at 0x1D000
      if (Driver_Flash.ProgramData(0x1D000, updateLineBit,16)==ARM_DRIVER_OK)
      {
         printf("Program: OK¥n");
         //Read at 0x1D000
         Driver_Flash.ReadData(0x1D000, compbuf, 16);
         if (memcmp(updateLineBit, compbuf, 16) == 0) {
            printf("Verify: OK¥n");
         } else {
            printf("Verify: NG¥n");
         }
      } else {
         printf("Program: NG¥n");
```

周辺回路の割り込みを禁止

ライブラリの初期化（使用する周辺回路初期化）

内蔵フラッシュメモリの消去

内蔵フラッシュメモリの書き込み

内蔵フラッシュメモリの読み込み

```
        }
    } else {
        printf("Erase NG¥n");
    }

    printf("Exit¥n");

    //Uninitialize
    Driver_Flash.Uninitialize();

    //enable Interrupt
      asm("CPSIE i");

      return 0;
}
```

ライブラリの初期化前の設定に戻す

周辺回路の割り込みを許可

3.2 内蔵 RAM 使用量

本ライブラリでは内蔵 RAM 領域を使用します。各機種の自己書き換えライブラリの RAM 使用量については、S1C31xxx 周辺回路サンプルソフトウェアパッケージに同梱しているリードミーをご参照ください。

3.3 ライブラリ使用上の注意

本ライブラリの使用にあたり、以下の点に注意してください。

- 本ライブラリが提供する関数を使用する前に必ず割込み禁止をしてください。
- 本ライブラリ実行時は、ライブラリを配置した領域を壊さないようにしてください。
- 本ライブラリを使用する際は、フラッシュメモリの書き換え可能回数に注意してください。フラッシュメモリの仕様については“S1C31xxx テクニカルマニュアル”をご参照ください。
- 本ライブラリを使用する際は、全ての周辺回路を停止してください。本ライブラリでは、以下の動作を行います。
  1. S1C31D01/S1C31D5x/S1C31W74 では、16 ビットタイマ(T16)の ch.0 を使用します。そのため 16 ビットタイマ ch.0 のレジスタの内容が変更されます。16 ビットタイマを使用しているアプリケーションプログラムで、本ライブラリを使用する場合はご注意ください。
  2. 本ライブラリでは、システムクロックを高速クロック(OSC3 or IOSC)に変更します。そのため、クロックジェネレータ (CLG) の制御レジスタ内容が変更されますのでご注意ください。
- 機種ごとに異なる仕様（セクタ情報、RAM 使用量等）については、S1C31xxx 周辺回路サンプルソフトウェアパッケージに同梱しているリードミーをご参照ください。
- 本ライブラリ使用時には、“S1C31xxx テクニカルマニュアル”に記載の基本外部結線図のとおりに Vpp 端子にコンデンサを接続し、Vpp 端子と他の端子との接続を切り離してください。

3.4 サンプルソフトウェア

1. サンプルソフトウェア仕様

本サンプルソフトウェアでは、S1C31 自己書き換えライブラリを使用して、アドレス 0x1D000 の領域をセクタ消去後、16 バイト書き込みます。

2. 準備

本サンプルソフトウェアのプロジェクトを実行する方法については、“S1C31xxx 周辺回路サンプルソフトウェアマニュアル”をご参照ください。

3. 動作概要

① 周辺回路の割り込みを禁止します。
② 内蔵フラッシュメモリの情報を取得します。（任意）
③ 本ライブラリのバージョンを取得します。（任意）
④ 本ライブラリを初期化します。（使用する周辺回路の初期化）
⑤ 内蔵フラッシュメモリ（0x1D000）を消去します。
⑥ 内蔵フラッシュメモリ（0x1D000）に更新用データupdateLineBit[]（16byte）を書き込みます。

```
0F 0E 0D 0C 0B 0A 09 08 07 06 05 04 03 02 01 00
```

⑦ 内蔵フラッシュメモリ（0x1D000）を読み込みます。
⑧ 読み込みデータcmpbuf[]と、更新用データupdateLineBit[]を比較し、結果を表示します。（ベリファイ）
⑨ 本ライブラリの初期化前の設定に戻します。
⑩ 周辺回路の割り込みを許可します。（任意）

4. ライブラリ仕様

4.1 ライブラリ関数詳細

本ライブラが提供する関数の詳細について記します。

関数名		
int32_t Initialize (ARM_Flash_SignalEvent_t cb_event)		
引数		
cb_event	ARM_Flash_SignalEvent_t	通常は NULL を設定
戻り値		
int32_t	ARM_DRIVER_OK (0)	
機能		
本ライブラリで使用する周辺回路の初期化を行う。 ① システムクロックの変更 ② T16 Ch.0 の初期化（S1C31D01/S1C31D5x/S1C31W74 のみ）		
備考		
本関数使用前に、周辺回路の割り込みを禁止してください。		

関数名	
int32_t Uninitialize (void)	
戻り値	
int32_t	ARM_DRIVER_OK (0)
機能	
Initialize 関数で初期化される前の設定に戻す。 ① T16 Ch.0 の設定（S1C31D01/S1C31D5x/S1C31W74 のみ） ② システムクロックの変更	
備考	
本関数使用後に、必要に応じて周辺回路の割り込みを許可してください。	

関数名		
int32_t EraseSector (uint32_t addr)		
引数		
addr	uint32_t	消去先セクタの先頭アドレス
戻り値		
int32_t	消去結果（エラーコード）	
機能		
内蔵フラッシュメモリを消去する。 ① 引数が内蔵フラッシュメモリの最終アドレス以下か確認する。 ② 消去先セクタが消去済み（0xffff）であるか確認する。 ③ ②で消去済みではない場合、セクタ消去を行う。 ④ ③で消去を実行した場合、消去先セクタが消去済み（0xffff）であるか確認する。（ベリファイ） ⑤ 消去結果を返す。		
備考		
1. 本関数の消去は、1 セクタ単位です。複数のセクタを消去する場合には、本関数を複数回コールしてください。 2. 本関数使用前に、周辺回路の割り込みを禁止してください。 3. 引数には、セクタの先頭アドレスを指定してください。セクタの先頭アドレス以外のアドレスを指定した場合、ベリファイエラーになる場合があります。 4. セクタ情報については、S1C31xxx 周辺回路サンプルソフトウェアパッケージ同梱のリードミーをご参照ください。		

関数名		
int32_t ProgramData (uint32_t addr, const void *data, uint32_t cnt)		
引数		
addr	uint32_t	書き込み先アドレス

data	const void *	書き込みデータ 書き込みデータへのポインタを表す。同ポインタは、RAM 空間を指していなければなりません。
cnt	uint32_t	書き込みデータサイズ
戻り値		
uint32_t	書き込み結果（エラーコード）	
機能		
内蔵フラッシュメモリを書き込む。 ① 引数が内蔵フラッシュメモリの最終アドレス以下かチェックする。 ② 指定した書き込み先アドレスに対し書き込みデータの書き込みを行う。 ③ 書き込み先アドレスが書き込みデータであるか確認する。（ベリファイ） ④ 書き込み結果を返す。		
備考		
1. 本関数の書き込みは、「バイト（8bit）」単位です。 2. 書き込み先は消去済み（0xffff）であることが前提になります。 3. 本関数使用前に、周辺回路の割り込みを禁止してください。		

関数名		
int32_t ReadData (uint32_t addr, unsigned char *data, int32_t cnt)		
引数		
addr	uint32_t	読み込み先アドレス
data	const void *	読み込みデータサイズ 読み込みデータへのポインタを表す。同ポインタは、RAM 空間を指していなければなりません。
cnt	uint32_t	読み込みデータサイズ
戻り値		
uint32_t	ARM_DRIVER_OK (0)	
機能		
内蔵フラッシュを読み込む。 ① 引数が内蔵フラッシュメモリの最終アドレス以下かチェックする。 ② 指定した読み込み先アドレスに対し読み込みを行う。 ③ 読み込み結果を返す。		
備考		
1. 本関数の読み込みは、「バイト（8bit）」単位です。 2. 本関数使用前に、周辺回路の割り込みを禁止してください。		

関数名	
ARM_DRIVER_VERSION GetVersion (void)	
戻り値	
ARM_DRIVER_VERSION	本ライブラリのバージョン
機能	
本ライブラリバージョンを取得する。	
備考	
なし	

関数名	
ARM_FLASH_INFO * GetInfo (void)	
戻り値	
ARM_FLASH_INFO *	内蔵フラッシュメモリの情報
機能	
内蔵フラッシュメモリの情報として以下を取得する。 ・セクタ数 ・セクタサイズ	
備考	
なし	

4.2 エラーコード定義

各関数の戻り値で使用されているエラーコードは、以下の通りです。

表 4.2.1 エラーコード

定義	値	意味
ARM_DRIVER_OK	0	正常終了
ARM_DRIVER_ERROR_TIMEOUT	-3	タイムアウト／ベリファイエラー
ARM_DRIVER_ERROR_UNSUPPORTED	-4	サポートしていない操作
ARM_DRIVER_ERROR_PARAMETER	-5	引数エラー

これらは、Drivers¥CMSIS¥Driver¥Include¥Driver_Common.h で定義をしています。

Appendix

A. ライブラリをプロジェクトへ組み込む方法(IAR EAWRM)

統合開発環境 IAR EWARM で作成したアプリケーションプログラムのプロジェクトに、本ライブラリを組み込む方法を以下に記します。IAR EWARM の詳細については、付属のマニュアルをご参照ください。

1. ライブラリの追加

(1) IAR EWARM メニューの［プロジェクト］＞［オプション］を選択します。
(2) 表示されたダイアログの［カテゴリ］リストから［リンカ］を選択します。
(3) ［ライブラリ］タブから、“追加ライブラリ”にS1C31xxx周辺回路サンプルソフトウェアパッケージの以下に同梱されている本ライブラリを追加します。
Middlewares¥seFlashLibrary¥Device¥S1C31xxx¥seFlashLibraryS1C31xxx.a

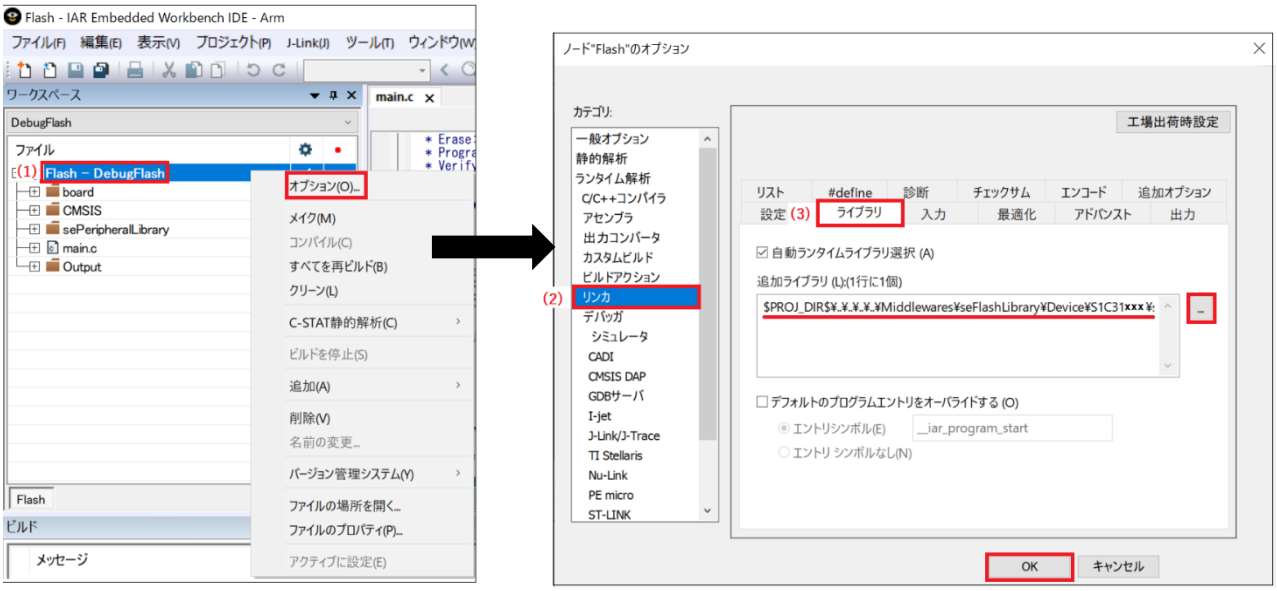

図 A.1　ライブラリの追加

2. インクルードパスの追加

(1) IAR EWARM メニューの［プロジェクト］＞［オプション］を選択します。
(2) 表示されたダイアログの［カテゴリ］リストから［C/C++コンパイラ］を選択します。
(3) ［プリプロセッサ］タブから、“追加インクルードディレクトリ”にS1C31xxx周辺回路サンプルソフトウェアパッケージに同梱されているドライバ定義の以下のインクルードパスを追加します。
Drivers¥CMSIS¥Driver¥Include

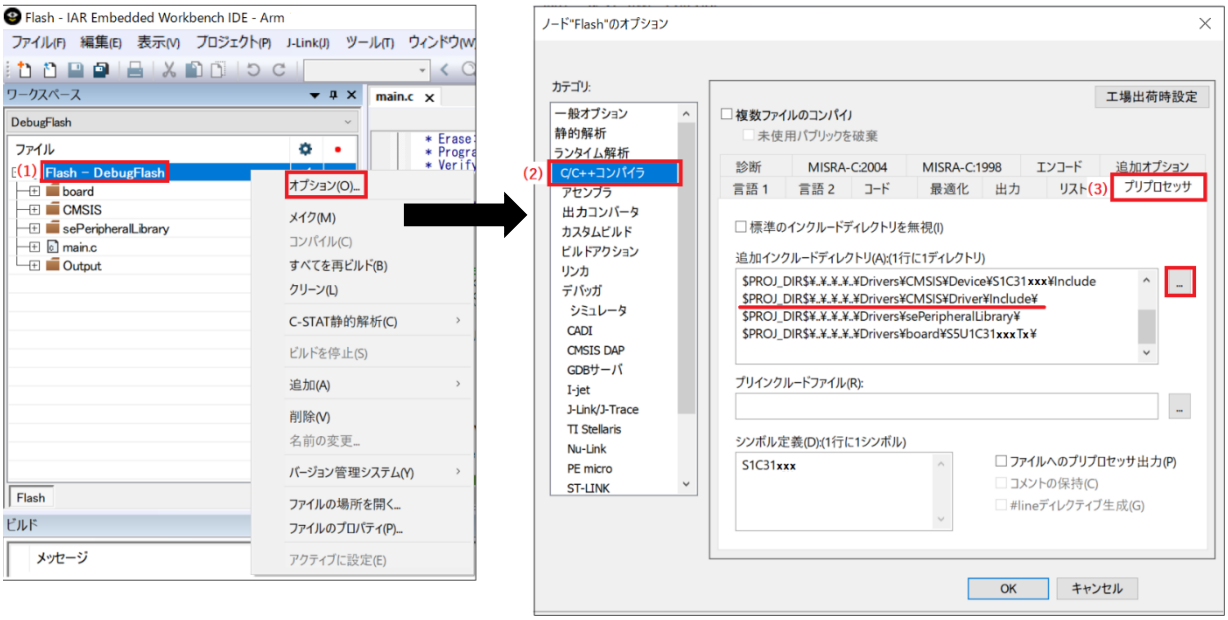

図 A.2　インクルードパスの追加

3. リンカスクリプトの設定

(1) プロジェクト内に同梱されているリンカスクリプトファイル（.icf）を編集します。

(2) S1C31xxx周辺回路サンプルソフトウェアパッケージ同梱のサンプルソフトウェアのリンカスクリプトファイル（S1C31xxx_fp_flash.icf）を参考に、以下のセクションを追加します。

```
/*###ICF### Section handled by ICF editor, don't touch! ****/
…
initialize by copy { readwrite };

initialize manually with packing = none { section .flash_common_text};

//initialize by copy with packing = none { section __DLIB_PERTHREAD }; // Required in a multi-threaded application
do not initialize   { section .noinit };

place at address mem:__ICFEDIT_intvec_start__ { readonly section .intvec };

place in ROM_region     { readonly };
place in RAM_region     { readwrite,
                          block CSTACK, block HEAP };

place in ROM_region     { section .flash_common_text_init};
place in RAM_region     { section .flash_common_text };
```

flash common tex セクションの生成

ROM 領域のコピー元セクションの指定

RAM 領域のコピー先セクションの指定

上記を追加することで、本ライブラリのコードを RAM 領域に配置します。

(3) IAR EWARMメニューの［プロジェクト］＞［オプション］を選択します。

(4) 表示されたダイアログの［カテゴリ］リストから［リンカ］を選択します。

(5) ［設定］タブから、“デフォルトのオーバライド”にチェックを入れ、編集したリンカスクリプトファイルを指定します。

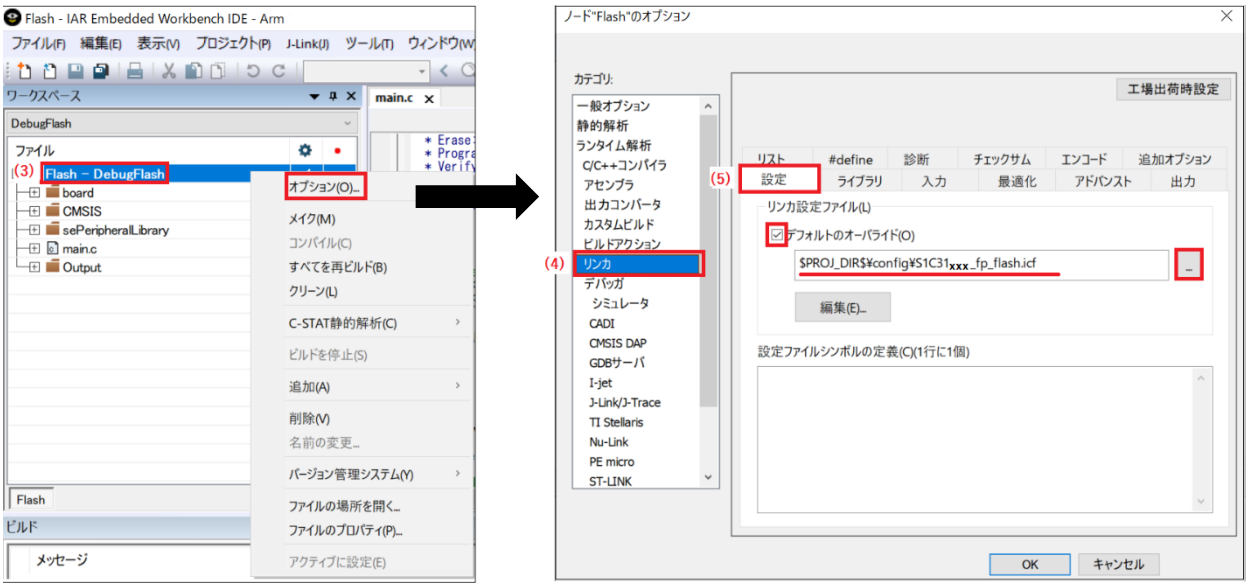

図 A.3　リンカスクリプトファイルの設定

B. ライブラリをプロジェクトへ組み込む方法(MDK-ARM)

統合開発環境 MDK-ARM（uVision）で作成したアプリケーションプログラムのプロジェクトに、S1C31 自己書き換えライブラリを組み込む方法を以下に記します。MDK-ARM の詳細については、付属のマニュアルをご参照ください。

1. ライブラリの追加

(1) uVision の［Project］ウィンドウから対象のソースフォルダを右クリックし、［Add Existing Files to Group ‘xxx’…］を選択します。

(2) 表示されたダイアログからS1C31xxx周辺回路サンプルソフトウェアパッケージの以下に同梱されている本ライブラリを追加します。
Middlewares¥seFlashLibrary¥Device¥S1C31xxx¥seFlashLibraryS1C31xxx.lib

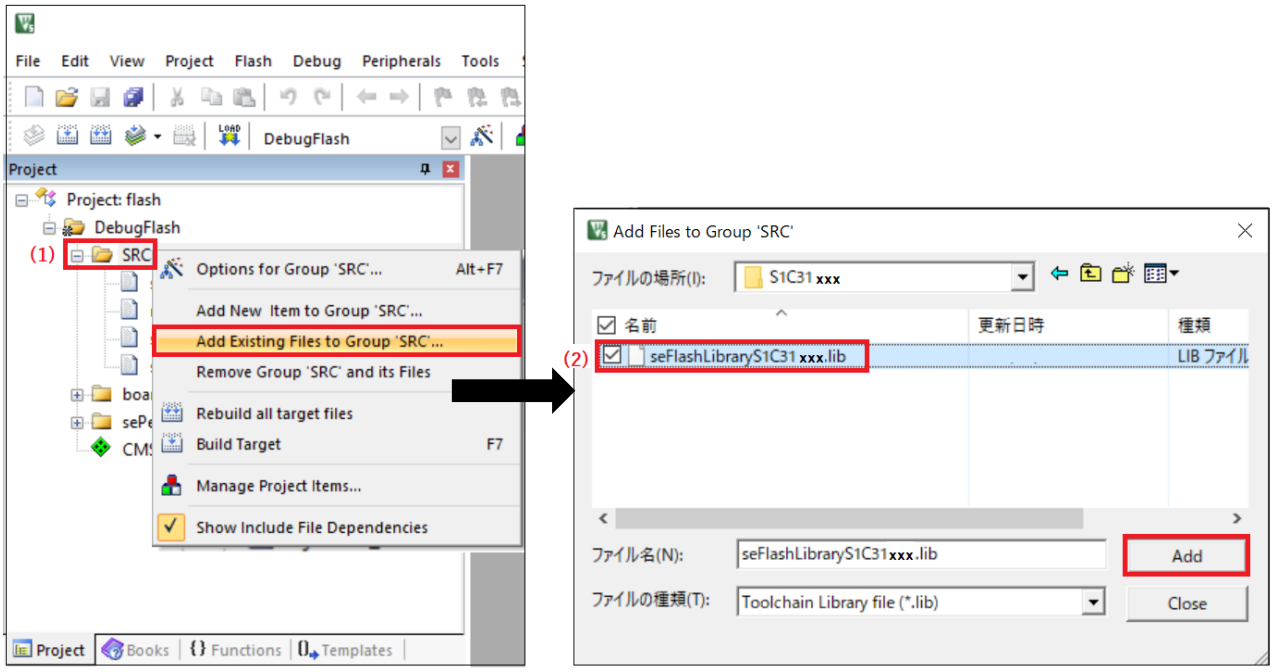

図 B.1　ライブラリの追加

2. インクルードパスの追加

(1) uVision メニューの［Project］＞［Options for Target ‘xxx’…］を選択します。

(2) 表示されたダイアログの［C/C++］＞‘Include Paths’からフォルダを参照します。

(3) ［New (Insert)］から、S1C31xxx周辺回路サンプルソフトウェアパッケージに同梱されているドライバ定義の以下のインクルードパスを追加します。
Drivers¥CMSIS¥Driver¥Include

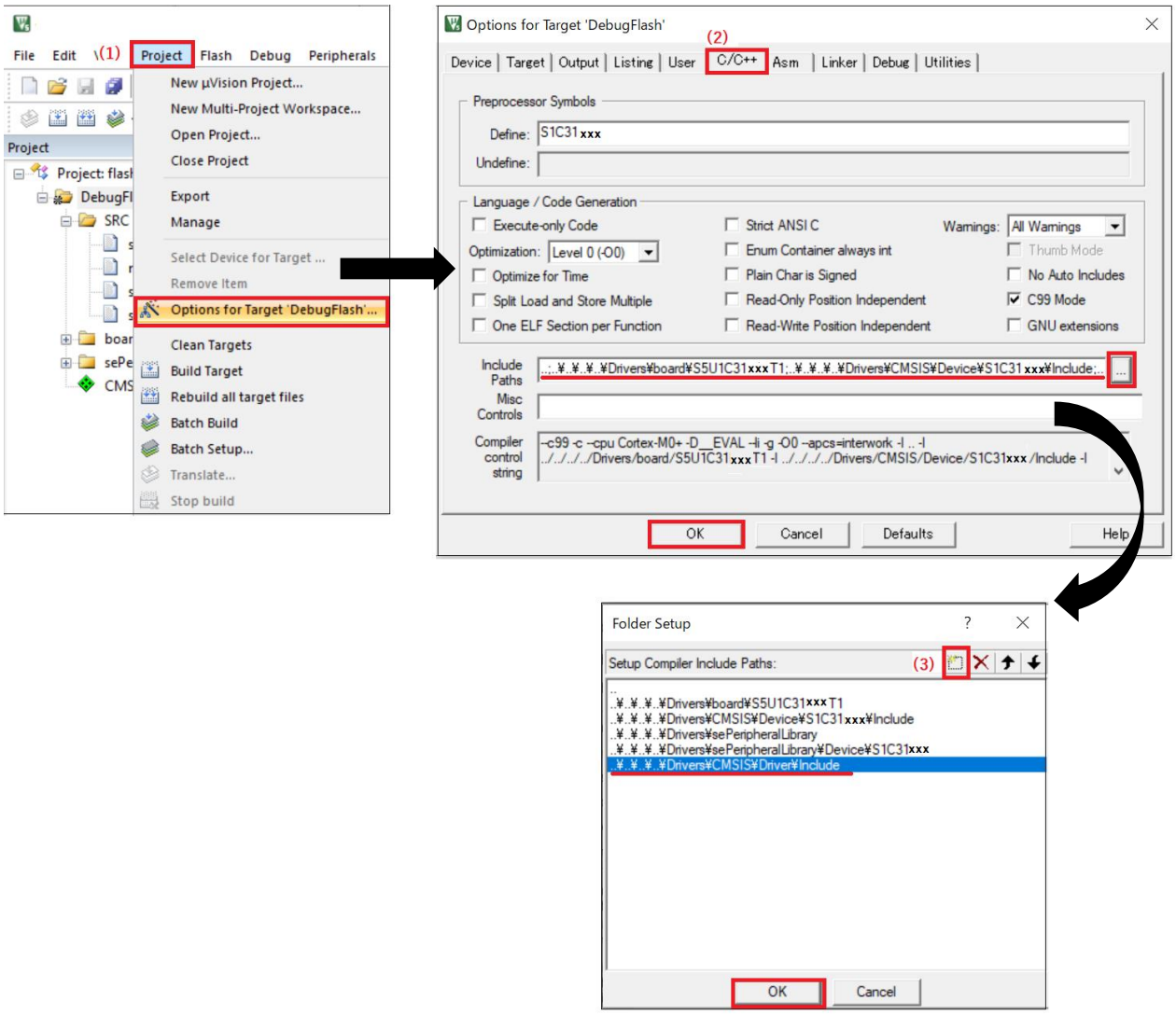

図 B.2　インクルードパスの追加

3. リンカスクリプトの設定

(1) プロジェクト内に同梱されているリンカスクリプトファイル（.sct）を編集します。

(2) S1C31xxx周辺回路サンプルソフトウェアパッケージ同梱のサンプルソフトウェアのリンカスクリプトファイル（flash_flash.sct）を参考に、以下のセクションを追加します。

```
; *************************************************************
; *** Scatter-Loading Description File generated by uVision ***
; *************************************************************
…
  RW_IRAM1 0x00150000 {    ; RW data
   .ANY (+RW +ZI)
  }
  RW_IRAM2 +0 {    ; RW data
   *(.flash_common_text)
  }
…
```

RAM 領域に flash_common_text セクションを生成

上記を追加することで、本ライブラリ内で、本ライブラリのコードを RAM 領域に配置します。

(3) uVisionメニューの［Project］＞［Options for Target ‘xxx’…］を選択します。
(4) 表示されたダイアログの［Linker］＞‘Scatter File’から編集したリンカスクリプトファイルを指定します。

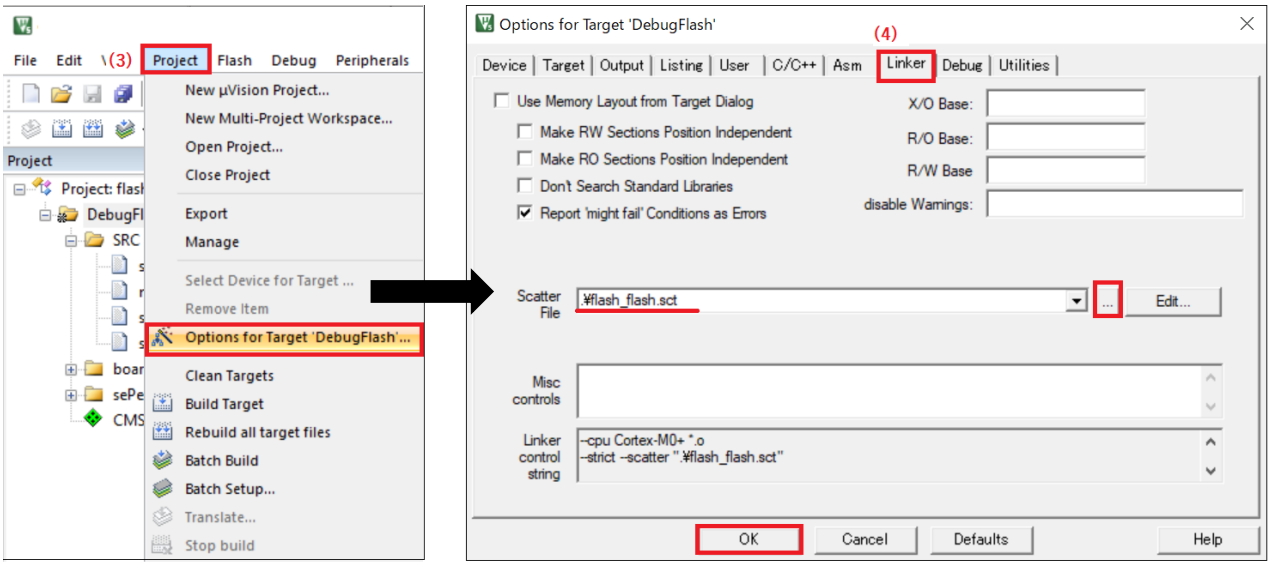

図 B.3　　リンカスクリプトファイルの設定

改訂履歴表

付－1

Rev. No.	日付	ページ	種別	改訂内容（旧内容を含む） および改訂理由
Rev 1.0	2021/4/30	全ページ	新規	新規制定

セイコーエプソン株式会社
営業本部　デバイス営業部

東京　〒160-8801 東京都新宿区新宿 4-1-6 JR 新宿ミライナタワー29F
大阪　〒530-6122　大阪市北区中之島 3-3-23　中之島ダイビル 22F

ドキュメントコード：414177500
2021 年 4 月 作成