User's Manual

RL78 ファミリ

Renesas Flash Driver RL78 Type02

ユーザーズマニュアル

ルネサスマイクロコンピュータ
RL78 / F24
RL78 / F23

本資料に記載の全ての情報は本資料発行時点のものであり、ルネサス エレクトロニクスは、
予告なしに、本資料に記載した製品または仕様を変更することがあります。
ルネサス エレクトロニクスのホームページなどにより公開される最新情報をご確認ください。

ルネサス エレクトロニクス
www.renesas.com

Rev.1.10　2022.12

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も、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、脆弱性問題に起因し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て、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いたします。
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

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/

商標について

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。

このマニュアルの使い方

対 象 者　このユーザーズマニュアルは、RL78/F23,F24 マイクロコントローラのフラッシュ・メモリを操作するシステムを開発するユーザを対象としています。

目　　的　このユーザーズマニュアルは、RL78/F23,F24 マイクロコントローラのフラッシュ・メモリの書き換えを行うために使用する Renesas Flash Driver (RFD) RL78 Type02 の機能を理解していただくことを目的としています。

構　　成　このマニュアルは、大きく分けて次の内容で構成しています。

1. 概要
2. システム構成
3. RFD RL78 Type02 API 関数
4. フラッシュ・メモリ・シーケンサ操作
5. サンプル・プログラム
6. RFD RL78 Type02 サンプル・プロジェクトの作成

読 み 方　このマニュアルを読むにあたっては、電気、論理回路、マイクロコントローラ、C 言語とアセンブラの一般的な知識を持つことを想定しています。
RL78/F23,F24 のハードウェア機能を理解するために、それぞれの RL78/F23,F24 製品のユーザーズマニュアルを参照してください。

凡　　例　データ表記の重み　：左が上位桁、右が下位桁
アクティブ・ロウの表記　：$\overline{\times\times\times}$（端子、信号名称に上線）
注　：本文中につけた注の説明
注意　：気をつけて読んでいただきたい内容
備考　：本文の補足説明
数の表記　：2 進数…××××または××××B
10 進数…xxxx
16 進数…XXXXH または 0xXXXX
２の累乗を示す単位の表記（アドレス空間、メモリ容量）
：K（キロ）2^{10} = 1024
M（メガ）2^{20} = 1024^2

関連資料　関連資料は暫定版の場合がありますが、この資料では「暫定」の表示をしておりません。あらかじめご了承ください。

No	Document Title	Document Number
1	RL78/F23,F24 ユーザーズマニュアル ハードウェア編	R01UH0944JJ
2	E1/E20/E2エミュレータ, E2エミュレータLite ユーザーズマニュアル別冊（RL78接続時の注意事項）	R20UT1994JJ
3	RL78用 Renesas Flash Driver, EEPROM Emulation Software 対象MCUリスト - Automotive	R20UT5229JJ

目　次

略語

用語	説明
RFD	Renesas Flash Driver（ルネサス・フラッシュ・ドライバ）
API	Application Program Interface.（アプリケーション・プログラム・インタフェース）
BGO	Background operation. データ・フラッシュ書き換え中に、プログラム・メモリ内の命令実行が可能。
FSW	Flash Shield Window（フラッシュ・シールド・ウインドウ） セルフ・プログラミング実行時に、指定したウインドウ範囲、またはウインドウ範囲以外のフラッシュ領域の書き込みおよび消去を禁止にする機能です。
RAM	Random Access Memory ランダムにアクセスできる揮発性メモリ。プログラム実行中に変更する値を保持するメモリです。
ROM	Read Only Memory 不揮発性メモリ。内容変更することができないメモリです。コード・フラッシュ・メモリを ROM と表現する場合があります。

専門用語

用語	説明
コード・フラッシュ・メモリ	アプリケーション・コードや定数データを格納するフラッシュ・メモリ ※本文中で、”CF”と略す場合があります。
データ・フラッシュ・メモリ	データを格納するフラッシュ・メモリ ※本文中で、”DF”と略す場合があります。
エクストラ領域	コンフィギュレーション設定領域、セキュリティ設定領域、ブロック保護設定領域、ブート・スワップ設定領域の総称
フラッシュ・メモリ・シーケンサ	RL78マイクロコントローラにはフラッシュ・メモリ制御用の専用回路が搭載されています。本書ではこの回路のことをフラッシュ・メモリ・シーケンサと呼びます。フラッシュ・メモリ・シーケンサに、コード・フラッシュ領域、またはデータ・フラッシュ領域を書き換える「コード/データ・フラッシュ領域シーケンサ」とエクストラ領域を書き換える「エクストラ領域シーケンサ」の総称です。
フラッシュ・メモリ制御モード	フラッシュ・メモリ・シーケンサの書き換え可否状態(モード)を示します。 - コード・フラッシュ・プログラミング・モード(Code flash programming mode) - データ・フラッシュ・プログラミング・モード(Data flash programming mode) - 非書き換えモード(Non-programmable mode)
コード・フラッシュ・プログラミング・モード	Code flash programming mode コード・フラッシュ・メモリ（および、エクストラ領域）を書き換え可能な状態（モード）を指します。
データ・フラッシュ・プログラミング・モード	Data flash programming mode データ・フラッシュ・メモリを書き換え可能な状態（モード）を指します。
非書き換えモード	Non-programmable mode フラッシュ・メモリ（および エクストラ領域） を書き換え不可の状態（モード）を指します。
セルフ・プログラミング	外部のフラッシュ・プログラミング・ツールを使用せず、ユーザ・プログラムを実行して、フラッシュ・メモリの書き換えを行うこと。
シリアル・プログラミング	外部のフラッシュ・プログラミング・ツールを使用し、フラッシュ・メモリの書き換えを行うこと。
ブート領域	リセット・ベクタを含む 00000H - 03FFFH の論理領域(16KB)。
ブート・クラスタ 0/1	ブート・クラスタは16Kバイトのブロック群で、0/1どちらか片方がブート領域に配置されます。 物理領域名： ブート・クラスタ0 [00000H - 03FFFH]（出荷時の論理アドレス） ブート・クラスタ1 [04000H - 07FFFH]（出荷時の論理アドレス）
ブート・スワップ	ブート・クラスタ0とブート・クラスタ1を置換します。

1 概 要

1.1 製品概要

Renesas Flash Driver(RFD) RL78 Type02 は、RL78/F23,F24 のフラッシュ・メモリ内のデータを書き換えるためのソフトウェアです。

1.1.1 目的

本書の目的は、RFD RL78 Type02 に関する情報を記述することです。

1.2 製品内容

RFD RL78 Type02 の API 関数をユーザ・プログラムから呼び出すことにより、コード・フラッシュ・メモリ、またはデータ・フラッシュ・メモリの内容を書き換えることができます。

RFD RL78 Type02 は、以下を含んでいます。

- 本ユーザーズマニュアル
- RL78/F23,F24 のデータ・フラッシュ・メモリとコード・フラッシュ・メモリを操作する RFD のソース・コード・ファイル。
- データ・フラッシュ・メモリ、コード・フラッシュ・メモリ、エクストラ領域の内容を消去、書き込みするサンプル・プログラム。

1.3 製品の特長

RFD RL78 Type02 は、フラッシュ・メモリ制御回路用コマンドの処理フローに基づいて、フラッシュの書き換え操作を行います。それぞれの API は、１つ、もしくは複数の関数で構成されており、各関数とユーザで行う処理を組み合わせて実現します。これは、ユーザ・アプリケーションに依存する処理、例えばタイムアウト処理のように、タイムアウト値がユーザ・アプリケーション・プログラムの実行条件によって異なるケースがあり、柔軟に対応できるよう、このような構成を採用しています。

ユーザ・アプリケーションが、RFD RL78 Type02 の API 関数を使ってフラッシュ・メモリを操作するときのイメージを図 1-1 に示します。

RFD RL78 Type02 では、複数の API 関数とユーザ・プログラムで行うべき処理を組み合わせた処理の例をサンプル・プログラムとして提供しています。フラッシュ操作処理をアプリケーションに組み込む際は、このサンプル・プログラムを参考にしてください。

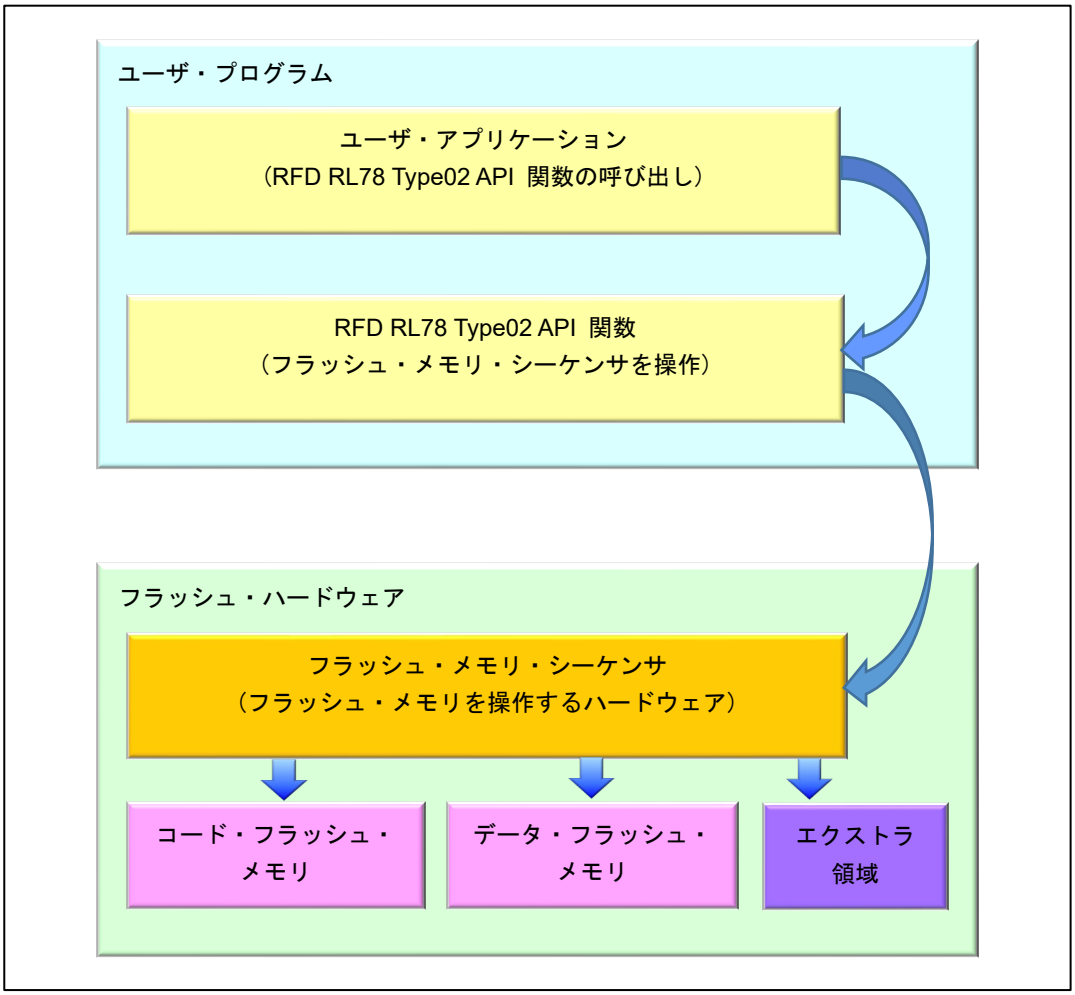


図 1-1 RFD RL78 Type02 の API 関数を使ったフラッシュ・メモリの操作イメージ

1.4 動作環境

- ホスト・マシン

動作環境は、ホスト・マシンには依存しませんが、C コンパイラ・パッケージ、デバッガおよび、エミュレータが動作する環境が必要となります。（RFD RL78 Type02 の開発は Windows10 Enterprise で実施）

- C コンパイラ・パッケージ

RFD RL78 Type02 の対象の C コンパイラ・パッケージを表 1-1 に示します。

表 1-1 対象の C コンパイラ・パッケージ

パッケージ	メーカー	バージョン
CC-RL（CS+, e^{2}studio 用）	Renesas Electronics	V1.11 以降
IAR Embedded Workbench®	IAR システムズ	V4.21 以降

- エミュレータ

動作確認したエミュレータを表 1-2 に示します。

表 1-2 動作確認したエミュレータ

エミュレータ	メーカー
E2 エミュレータ	Renesas Electronics
E2 エミュレータ Lite	Renesas Electronics

- ターゲット MCU

RL78/F24

RL78/F23

1.5 注意事項

(1) コード・フラッシュ/エクストラ領域の書き換え操作

コード・フラッシュ/エクストラ領域を書き換え操作する場合は、実行するプログラムを RAM に配置してください。

(2) データ・フラッシュ領域を操作する場合の前提条件

データ・フラッシュ領域を操作する前に、必ず、データ・フラッシュ・コントロール・レジスタ(DFLCTL レジスタ)の DFLEN ビット[bit0] = 1(データ・フラッシュのアクセス許可)に設定しておく必要があります。

(3) フラッシュ・メモリ書き換え操作中のプログラム実行

RL78/F23,F24 のセルフ・プログラミングは、フラッシュ・メモリ・シーケンサを使用し、フラッシュ・メモリの書き換えを制御します。フラッシュ・メモリの書き換えが可能な"フラッシュ・メモリ制御モード"では、操作対象のフラッシュ・メモリは参照できなくなります。

・コード・フラッシュ・プログラミング・モードでは、コード・フラッシュ・メモリを参照することができません。コード・フラッシュ・プログラミング・モード中に実行する ROM(コード・フラッシュ・メモリ)上の RFD RL78 Type02 の関数、ユーザ・プログラム、およびそれぞれの参照データは、事前に RAM へコピーして、RAM 上で実行、参照する必要があります。

・データ・フラッシュ・プログラミング・モードでは、データ・フラッシュ・メモリを参照することができません。データ・フラッシュ・プログラミング・モード中に参照するデータは、事前に RAM へコピーして、RAM 上で参照する必要があります。

(4) 内部ベリファイ・コマンドを使用する場合の注意点

書き込んだ直後に、対象の領域に内部ベリファイ・コマンドを 1 回だけ実行してください。

(5) オンチップ・デバッガでセルフ・プログラミングのデバッグをする場合の注意事項

オンチップ・デバッガでセルフ・プログラミングのデバッグをする場合、デバッグ実行時に RAM の先頭アドレスから 128byte の領域を使用するため、この領域を空けてください。それと同時に、ご使用の開発環境が CS+, e^{2}studio の場合は、デバッガでフラッシュのセルフ・プログラミングを行う設定をしておく必要があります。

・CS+の設定例:

プロジェクトの"RL78 E2 [Lite](デバッグ・ツール)"から"接続用設定"タブを選択、"フラッシュ"の「Flash のセルフ・プログラミングを行う」を「はい」に設定します。

・e^{2}studio の設定例:

プロジェクトの"プロパティ"から"実行/デバッグ設定"を選択し、対象の"HardwareDebug"設定を編集します。"Debugger"タブを選択後、"Connection Settings"タブを選択し、表示された「フラッシュのセルフ・プログラミングを行う」を「はい」に設定します。

1.6 C コンパイラ定義

RFD RL78 Type02 のヘッダ・ファイル(r_rfd_compiler.h)に記述する対象コンパイラの定義を示します。

コンパイラごとに異なる記述が必要なため、使用しているコンパイラを” r_rfd_compiler.h”ファイルで認識し、対象のコンパイラ用の定義を使用します。

・C コンパイラの定義

- CC-RL コンパイラの定義 :

”__CCRL__”が定義されている場合

#define COMPILER_CC (1)

- IAR コンパイラ(V2/V3/V4) :

” __IAR_SYSTEMS_ICC__”が定義されている場合

#define COMPILER_IAR (2)

<r_rfd_compiler.h ファイル内の記述>

```
/* Compiler definition */
#define COMPILER_CC      (1)
#define COMPILER_IAR     (2)

#if defined (__CCRL__)
      #define COMPILER COMPILER_CC
#elif defined (__IAR_SYSTEMS_ICC__)
      #define COMPILER COMPILER_IAR
#else
      /* Unknown compiler error */
      #error   "Non-supported compiler."
#endif

/* Compiler dependent definition */
#if (COMPILER_CC == COMPILER)
      #define R_RFD_FAR_FUNC                                    __far
      #define R_RFD_NO_OPERATION                                __nop
      #define R_RFD_DISABLE_INTERRUPT                           __DI
      #define R_RFD_ENABLE_INTERRUPT                            __EI
      #define R_RFD_GET_PSW_IE_STATE                            __get_psw
      #define R_RFD_IS_PSW_IE_ENABLE(u08_psw_ie_state) (0u != ((u08_psw_ie_state) & 0x80u))
#elif (COMPILER_IAR == COMPILER)
      #define R_RFD_FAR_FUNC                                    __far_func
      #define R_RFD_NO_OPERATION                                __no_operation
      #define R_RFD_DISABLE_INTERRUPT                           __disable_interrupt
      #define R_RFD_ENABLE_INTERRUPT                            __enable_interrupt
      #define R_RFD_GET_PSW_IE_STATE                            __get_interrupt_state
      #define R_RFD_IS_PSW_IE_ENABLE(u08_psw_ie_state) (0u != ((u08_psw_ie_state) & 0x80u))
#else
      /* Unknown compiler error */
      #error   "Non-supported compiler."
#endif
```

・C コンパイラ・オプション

以下に、動作確認した C コンパイラ・オプションを示します。

- [CC-RL(CS+)]

Major compile options:

-cpu=S3 -g -g_line -lang=c99

- [IAR(Embedded Workbench)]

Major compile options:

--core s3 --calling_convention v2 --code_model far --data_model near -e -Ol --no_cse --no_unroll --no_inline --no_code_motion --no_tbaa --no_cross_call --no_scheduling --no_clustering --debug

2 システム構成

2.1 ファイル構成

2.1.1 フォルダ構成

RFD RL78 Type02 のフォルダ構成を図 2-1 に示します。

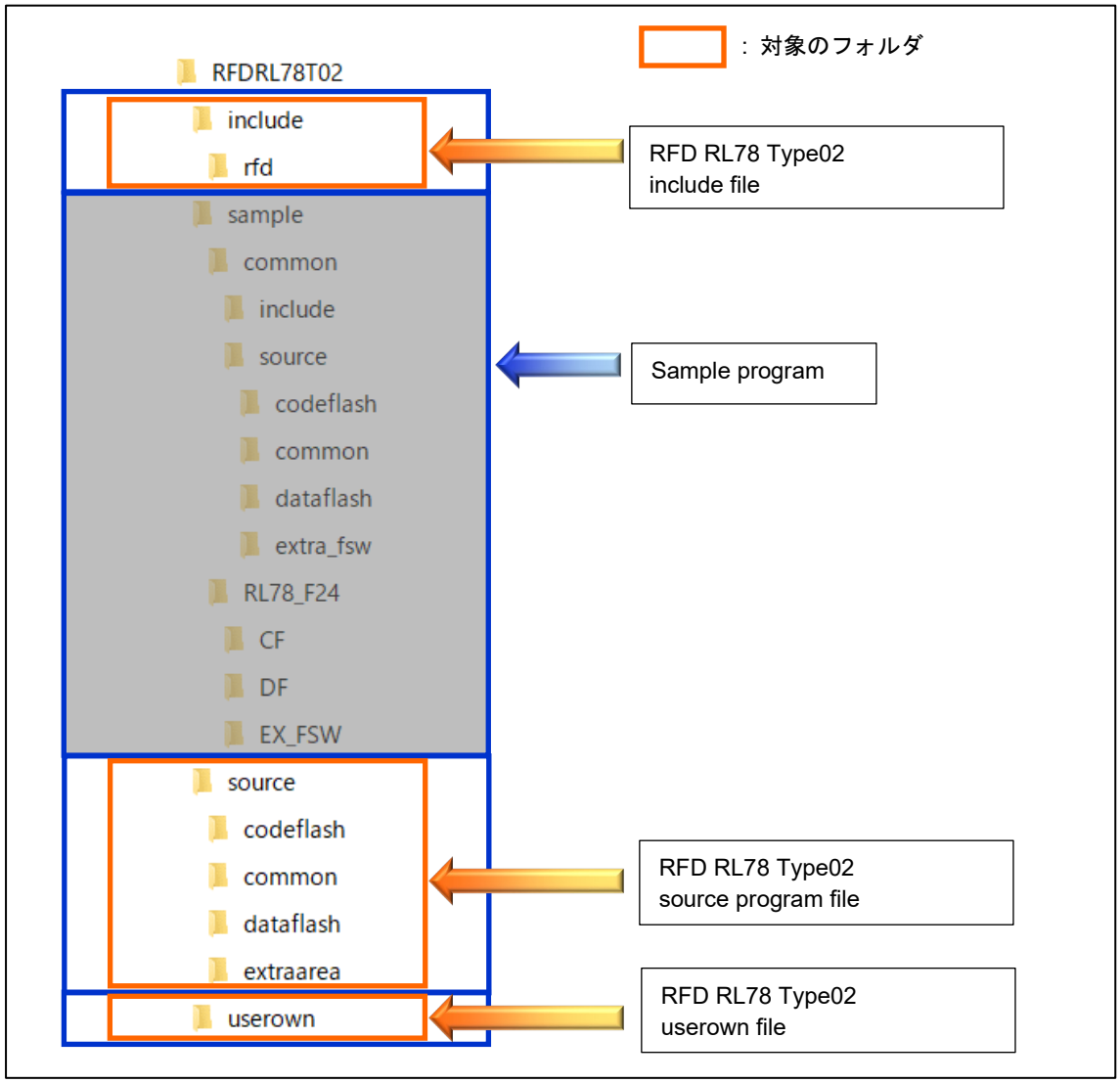


図 2-1 RFD RL78 Type02 のフォルダ構成(Folder Structure)

注）図 2-1 では RL78/F24 を使用する場合の例を記載しています
サンプル用フォルダについては、「5.1.1 フォルダ構成」をご確認ください。

2.1.2 ファイル・リスト

2.1.2.1 ソース・ファイル・リスト

“source\common\”フォルダ内のプログラム・ソース・ファイルを表 2-1 に示します。

表 2-1 “source\common\”フォルダ内プログラム・ソース・ファイル

No	ソース・ファイル名	概略(Summary)
1	r_rfd_common_api.c	フラッシュ・メモリ操作の共通設定 API 関数ファイル
2	r_rfd_common_control_api.c	フラッシュ・メモリ操作の共通コマンド制御 API 関数ファイル
3	r_rfd_common_get_api.c	フラッシュ・メモリ操作の共通情報取得 API 関数ファイル
4	r_rfd_common_extension_api.c	フラッシュ・メモリ操作の共通拡張機能 API 関数ファイル

“source\codeflash\”フォルダ内のプログラム・ソース・ファイルを表 2-2 に示します。

表 2-2 “source\codeflash\”フォルダ内プログラム・ソース・ファイル

No	ソース・ファイル名	概略(Summary)
1	r_rfd_code_flash_api.c	コード・フラッシュ・メモリ制御 API 関数ファイル

“source\dataflash\”フォルダ内のプログラム・ソース・ファイルを表 2-3 に示します。

表 2-3 “source\dataflash\”フォルダ内プログラム・ソース・ファイル

No	ソース・ファイル名	概略(Summary)
1	r_rfd_data_flash_api.c	データ・フラッシュ・メモリ制御 API 関数ファイル

“source\extraarea\”フォルダ内のプログラム・ソース・ファイルを表 2-4 に示します。

表 2-4 “source\extraarea\”フォルダ内プログラム・ソース・ファイル

No	ソース・ファイル名	概略(Summary)
1	r_rfd_extra_area_api.c	エクストラ領域制御 API 関数ファイル
2	r_rfd_extra_area_security_api.c	エクストラ領域のセキュリティ機能制御 API 関数ファイル

“userown\”フォルダ内のプログラム・ソース・ファイルを表 2-5 に示します。

表 2-5 “userown\“フォルダ内プログラム・ソース・ファイル

No	ソース・ファイル名	概略(Summary)
1	r_rfd_common_userown.c	RFD RL78 Type02 内でユーザ処理を実施するためのフック関数ファイル

2.1.2.2 ヘッダ・ファイル・リスト

“include\rfd”フォルダ内のプログラム・ヘッダ・ファイルを表 2-6 に示します。

表 2-6 “include\rfd”フォルダ内プログラム・ヘッダ・ファイル

No	ヘッダ・ファイル名	概略(Summary)
1	r_rfd.h	共通ヘッダ・ファイルを記述したファイル。 RFD RL78 Type02 使用時にインクルードする必要があるファイル
2	r_rfd_compiler.h	RFD RL78 Type02 で使用するコンパイラごとに異なる定義等を記述したファイル
3	r_rfd_memmap.h	RFD RL78 Type02 で使用するセクションを記述するためのマクロを定義したファイル
4	r_rfd_device.h	RFD RL78 Type02 で使用するハードウェア固有のマクロを定義したファイル
5	r_rfd_types.h	RFD RL78 Type02 で使用する変数の型を定義したファイル
6	r_typedefs.h	RFD RL78 Type02 で使用するデータの型を定義したファイル

“include\”フォルダ内のプログラム・ヘッダ・ファイルを表 2-7 に示します。

表 2-7 “include\”フォルダ内プログラム・ヘッダ・ファイル

No	ヘッダ・ファイル名	概略(Summary)
1	r_rfd_common_api.h	フラッシュ・メモリ操作の共通設定 API 関数のプロトタイプ宣言を定義したファイル
2	r_rfd_code_flash_api.h	コード・フラッシュ・メモリ制御 API 関数のプロトタイプ宣言を定義したファイル
3	r_rfd_common_control_api.h	フラッシュ・メモリ操作の共通コマンド制御 API 関数のプロトタイプ宣言を定義したファイル
4	r_rfd_common_get_api.h	フラッシュ・メモリ操作の共通情報取得 API 関数のプロトタイプ宣言を定義したファイル
5	r_rfd_common_extension_api.h	フラッシュ・メモリ操作の共通拡張機能 API 関数のプロトタイプ宣言を定義したファイル
6	r_rfd_common_userown.h	RFD RL78 Type02 内でユーザ処理を実施するためのフック関数のプロトタイプ宣言を定義したファイル
7	r_rfd_data_flash_api.h	データ・フラッシュ・メモリ制御 API 関数のプロトタイプ宣言を定義したファイル
8	r_rfd_extra_area_api.h	エクストラ領域制御 API 関数のプロトタイプ宣言を定義したファイル
9	r_rfd_extra_area_security_api.h	エクストラ領域のセキュリティ機能制御 API 関数プロトタイプ宣言を定義したファイル

2.2 RL78/F23,F24 リソース

2.2.1 メモリ・マップ

RL78/F23,F24 のコード・フラッシュ:CF (1 ブロック:1Kbyte)、データ・フラッシュ:DF（1 ブロック:1Kbyte）、RAM のメモリ・マップを表 2-8 に示します。

表 2-8 コード・フラッシュ、データ・フラッシュ、RAM のメモリ・マップ

RL78	デバイス型名	コード・フラッシュ:CF	RAM
F23	R7F123FxG (x=B,G,L,M)	128KB (00000H-3FFFFH)	12KB (FCF00H-FFEFFH)
	データ・フラッシュ:DF	8KB (F1000H-F2FFFH) RL78/F23 共通	
F24	R7F124FxJ (x=B,G,L,M,P)	256KB (00000H-3FFFFH)	24KB (F9F00H-FFEFFH)
	データ・フラッシュ:DF	16KB (F1000H-F4FFFH) RL78/F24 共通	

2.2.2 ブロックイメージ

RL78/F24 のコード・フラッシュ(CF)、データ・フラッシュ(DF)のブロックイメージの例を図 2-2、図 2-3 に示します。

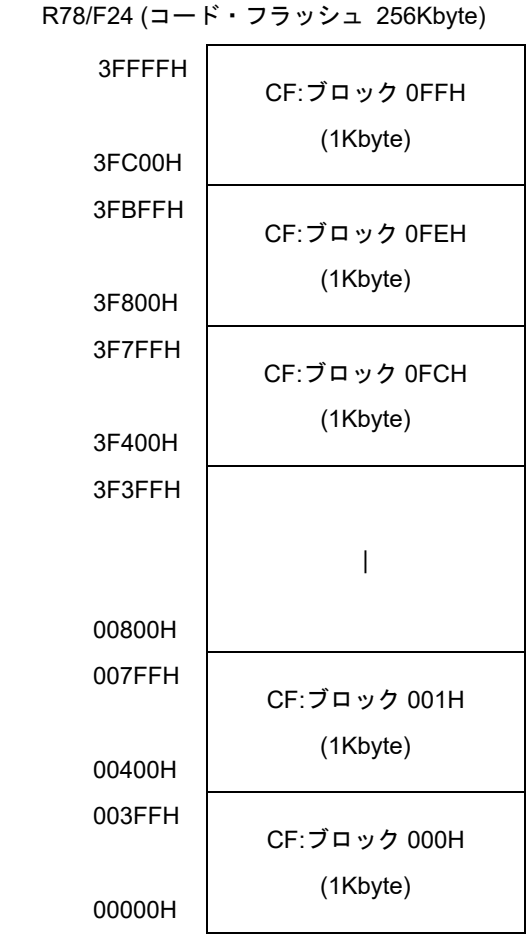


図 2-2 コード・フラッシュ のブロックイメージ

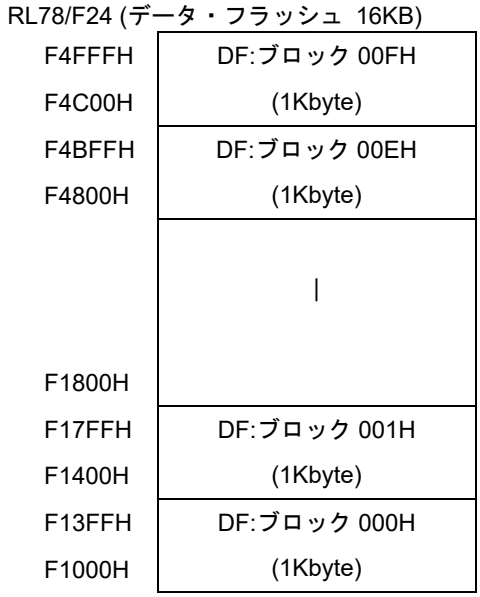


図 2-3 データ・フラッシュのブロックイメージ

2.2.3 レジスター覧(フラッシュ・メモリ・シーケンサ操作関連)

RFD RL78 Type02 が使用する RL78/F23,F24 内蔵レジスタの一覧を表 2-9 に示します。

表 2-9 RFD RL78 Type02 使用 RL78/F23,F24 内蔵レジスター覧

Base	Offset	Register Name	Size	Function name / Note
F0000H	90H	DFLCTL	1byte	データ・フラッシュ・コントロール・レジスタ
	C0H	FLPMC	1byte	フラッシュ・プログラミング・モード・コントロール・レジスタ
	C1H	FLARS	1byte	フラッシュ領域選択レジスタ
	C2H	FLAPL	2byte	フラッシュ・アドレス・ポインタ・レジスタ L
	C4H	FLAPH	1byte	フラッシュ・アドレス・ポインタ・レジスタ H
	C5H	FSSQ	1byte	フラッシュ・メモリ・シーケンサ制御レジスタ
	C6H	FLSEDL	2byte	フラッシュ・エンド・アドレス・ポインタ・レジスタ L
	C8H	FLSEDH	1byte	フラッシュ・エンド・アドレス・ポインタ・レジスタ H
	C9H	FLRST	1byte	フラッシュ・レジスタ初期化レジスタ
	CAH	FSASTL	1byte	フラッシュ・メモリ・シーケンサ・ステータス・レジスタ L
	CBH	FSASTH	1byte	フラッシュ・メモリ・シーケンサ・ステータス・レジスタ H
	CCH	FLWL	2byte	フラッシュ・ライト・バッファ・レジスタ L
	CEH	FLWH	2byte	フラッシュ・ライト・バッファ・レジスタ H
FFF00H	B0H	FLSEC	2byte	フラッシュ・セキュリティ・フラグ・モニタ・レジスタ
	B2H	FLFSWS	2byte	フラッシュ FSW モニタ・レジスタ S
	B4H	FLFSWE	2byte	フラッシュ FSW モニタ・レジスタ E
	B6H	FSSET	1byte	フラッシュ・メモリ・シーケンサ初期設定レジスタ
	B7H	FSSE	1byte	フラッシュ・エクストラ領域シーケンサ制御レジスタ
	C0H	PFCMD	1byte	フラッシュ・プロテクト・コマンド・レジスタ
	C1H	PFS	1byte	フラッシュ・ステータス・レジスタ
	C6H	FLWE	1byte	フラッシュ・ECC ライト・バッファ・レジスタ

2.3 RFD RL78 Type02 使用リソース

2.3.1 RFD RL78 Type02 使用時のセクション

2.3.1.1 コード・フラッシュ書き換え時のセクション

コード・フラッシュを書き換える”コード・フラッシュ・プログラミング・モード”では、コード・フラッシュが参照できなくなります。プログラム領域に該当するセクションは、予め ROM から RAM へ転送しておき、RAM でプログラムを実行する必要があります。また、RAM 配置の初期値付きグローバル変数(RFD_DATA)は、対象のコンパイラの指示に従い、初期値を ROM からコピーしておく必要があります。

コード・フラッシュ書き換え時に使用するセクションと配置の一覧を表 2-10 に示します。

表 2-10 コード・フラッシュ書き換え時のセクション

セクション名	内容	配置
RFD_CMN	共通フラッシュ制御 API 関数のプログラム・セクション	RAM
RFD_CF	コード・フラッシュ制御 API 関数のプログラム・セクション	RAM
RFD_DATA	初期値付きグローバル変数のデータ・セクション	RAM
SMP_CMN	共通フラッシュ制御 サンプル関数のプログラム・セクション	RAM
SMP_CF	コード・フラッシュ制御 サンプル関数のプログラム・セクション	RAM

2.3.1.2 データ・フラッシュ書き換え時のセクション

RAM 配置の初期値付きグローバル変数(RFD_DATA)は、対象のコンパイラの指示に従い、初期値を ROM からコピーしておく必要があります。

データ・フラッシュ書き換え時に使用するセクションと配置の一覧を表 2-11 に示します。

表 2-11 データ・フラッシュ書き換え時のセクション

セクション名	内容	配置
RFD_CMN	共通フラッシュ制御 API 関数のプログラム・セクション	ROM
RFD_DF	データ・フラッシュ制御 API 関数のプログラム・セクション	ROM
RFD_DATA	初期値付きグローバル変数のデータ・セクション	RAM
SMP_CMN	共通フラッシュ制御 サンプル関数のプログラム・セクション	ROM
SMP_DF	データ・フラッシュ制御 サンプル関数のプログラム・セクション	ROM

2.3.1.3 エクストラ領域書き換え時のセクション

エクストラ領域を書き換える”コード・フラッシュ・プログラミング・モード”では、コード・フラッシュが参照できなくなります。プログラム領域に該当するセクションは、予め ROM から RAM へ転送しておき、RAM でプログラムを実行する必要があります。また、RAM 配置の初期値付きグローバル変数(RFD_DATA)は、対象のコンパイラの指示に従い、初期値を ROM からコピーしておく必要があります。

エクストラ領域書き換え時に使用するセクションと配置の一覧を表 2-12 に示します。

表 2-12 エクストラ領域書き換え時のセクション

セクション名	内容	配置
RFD_CMN	共通フラッシュ制御 API 関数のプログラム・セクション	RAM
RFD_EX	エクストラ領域制御 API 関数のプログラム・セクション	RAM
RFD_DATA	初期値付きグローバル変数のデータ・セクション	RAM
SMP_CMN	共通フラッシュ制御 サンプル関数のプログラム・セクション	RAM
SMP_EX	エクストラ領域制御 サンプル関数のプログラム・セクション	RAM

2.3.2 API 関数のコード・サイズとスタック・サイズ

RFD RL78 Type02 の API 関数が使用するコード・サイズとスタック・サイズを表 2-13 に示します。

表 2-13 RFD RL78 Type02 の API 関数が使用するコード・サイズとスタック・サイズ

API 関数名	コード・サイズ(Bytes)		スタック・サイズ(Bytes)	
	CC-RL	IAR	CC-RL	IAR
R_RFD_Init	37	44	4	4
R_RFD_SetDataFlashAccessMode	36	20	10	10
R_RFD_SetFlashMemoryMode	264	284	14	16
R_RFD_CheckFlashMemoryMode	26	36	4	4
R_RFD_CheckCFDFSeqEndStep1	13	24	4	6
R_RFD_CheckExtraSeqEndStep1	13	23	4	6
R_RFD_CheckCFDFSeqEndStep2	8	19	4	6
R_RFD_CheckExtraSeqEndStep2	6	19	4	6
R_RFD_GetSeqErrorStatus	8	8	4	4
R_RFD_ClearSeqRegister	11	10	4	4
R_RFD_ForceStopSeq	6	5	4	4
R_RFD_ForceReset	2	2	4	4
R_RFD_SetBootAreaImmediately	15	19	4	4
R_RFD_GetSecurityAndBootFlags	5	5	4	4
R_RFD_GetFSW	22	24	8	6
r_rfd_wait_count	19	19	6	6
R_RFD_EraseCodeFlashReq	34	43	4	4
R_RFD_WriteCodeFlashReq	28	58	4	6
R_RFD_BlankCheckCodeFlashReq	34	43	4	4
R_RFD_IVerifyCodeFlashReq	34	43	4	4
R_RFD_EraseDataFlashReq	29	41	4	4
R_RFD_WriteDataFlashReq	20	27	4	6
R_RFD_BlankCheckDataFlashReq	34	76	6	12
R_RFD_IVerifyDataFlashReq	34	76	6	12
R_RFD_SetExtraEraseProtectReq	24	29	4	4
R_RFD_SetExtraWriteProtectReq	24	29	4	4
R_RFD_SetExtraBootAreaProtectReq	24	29	4	4
R_RFD_SetExtraBootAreaReq	48	77	4	6
R_RFD_SetExtraFSWReq	21	30	4	4
R_RFD_HOOK_EnterCriticalSection	9	9	4	4
R_RFD_HOOK_ExitCriticalSection	11	10	4	4

3 RFD RL78 Type02 API 関数

3.1 RFD RL78 Type02 API 関数 一覧

3.1.1 共通フラッシュ制御 API 関数

RFD RL78 Type02 の共通フラッシュ制御 API 関数一覧を表 3-1 に示します。

表 3-1 RFD RL78 Type02 共通フラッシュ制御 API 関数一覧

	API 関数名	概要
1	R_RFD_Init	引数で指定された周波数をフラッシュ・メモリ・シーケンサに設定し、RFD RL78 Type02 の初期化を行います。
2	R_RFD_SetDataFlashAccessMode	引数で指定されたデータ・フラッシュへのアクセスの許可、または、禁止を設定します。
3	R_RFD_SetFlashMemoryMode	フラッシュ・メモリ・シーケンサを引数で指定されたフラッシュ・メモリ制御モードへ変更後、CPU 動作周波数の値を設定します。
4	R_RFD_CheckFlashMemoryMode	引数で指定されたモードであるかどうかを確認します。
5	R_RFD_CheckCFDFSeqEndStep1	起動したコード/データ・フラッシュ領域シーケンサの動作終了を確認します。
6	R_RFD_CheckExtraSeqEndStep1	起動したエクストラ領域シーケンサの動作終了を確認します。
7	R_RFD_CheckCFDFSeqEndStep2	フラッシュ・メモリ・シーケンサ制御レジスタのクリアにより、コマンド動作が終了したかどうかを確認します。
8	R_RFD_CheckExtraSeqEndStep2	エクストラ領域シーケンサ制御レジスタのクリアにより、コマンド動作が終了したかどうかを確認します。
9	R_RFD_GetSeqErrorStatus	コード/データ・フラッシュ領域シーケンサ・コマンド、または、エクストラ領域シーケンサ・コマンドにより、発生したエラー情報を取得します。
10	R_RFD_ClearSeqRegister	コード/データ・フラッシュ領域シーケンサ、エクストラ領域シーケンサ制御を行うレジスタをクリアします。
11	R_RFD_ForceStopSeq	コード/データ・フラッシュ領域シーケンサ動作を強制停止します。
12	R_RFD_ForceReset	CPU の内部リセットを発生させます。
13	R_RFD_SetBootAreaImmediately	引数で指定されたブート・クラスタを、ブート領域(00000H~03FFFH)に即時設定します。
14	R_RFD_GetSecurityAndBootFlags	セキュリティ・フラグ(各プロテクト・フラグ)とブート領域切替フラグの情報を取得します。
15	R_RFD_GetFSW	フラッシュ・シールド・ウインドウの範囲を取得します。
16	r_rfd_wait_count	1 カウントで 1μsec として、入力されたパラメータ値の時間をソフトウェアループで Wait します。

3.1.2 コード・フラッシュ制御 API 関数

RFD RL78 Type02 のコード・フラッシュ制御 API 関数一覧を表 3-2 に示します。

表 3-2 RFD RL78 Type02 コード・フラッシュ制御 API 関数一覧

	API 関数名	概要
1	R_RFD_EraseCodeFlashReq	コード/データ・フラッシュ領域シーケンサを起動し、コード・フラッシュの消去(1 ブロック)を開始します。
2	R_RFD_WriteCodeFlashReq	コード/データ・フラッシュ領域シーケンサを起動し、コード・フラッシュの書き込み(4byte)を開始します。
3	R_RFD_BlankCheckCodeFlashReq	コード/データ・フラッシュ領域シーケンサを起動し、コード・フラッシュのブランク・チェック(1 ブロック)を開始します。
4	R_RFD_IVerifyCodeFlashReq	コード/データ・フラッシュ領域シーケンサを起動し、コード・フラッシュの内部ベリファイ(1 ブロック)を開始します。

3.1.3 データ・フラッシュ制御 API 関数

RFD RL78 Type02 のデータ・フラッシュ制御 API 関数一覧を表 3-3 に示します。

表 3-3 RFD RL78 Type02 データ・フラッシュ制御 API 関数一覧

	API 関数名	概要
1	R_RFD_EraseDataFlashReq	コード/データ・フラッシュ領域シーケンサを起動し、データ・フラッシュの消去(1 ブロック)を開始します。
2	R_RFD_WriteDataFlashReq	コード/データ・フラッシュ領域シーケンサを起動し、データ・フラッシュの書き込み(1byte)を開始します。
3	R_RFD_BlankCheckDataFlashReq	コード/データ・フラッシュ領域シーケンサを起動し、データ・フラッシュのブランク・チェック(指定バイト数)を開始します。
4	R_RFD_IVerifyDataFlashReq	コード/データ・フラッシュ領域シーケンサを起動し、データ・フラッシュの内部ベリファイ(書き込みバイト数)を開始します。

3.1.4 エクストラ領域制御 API 関数

RFD RL78 Type02 のエクストラ領域制御 API 関数一覧を表 3-4 に示します。

表 3-4 RFD RL78 Type02 エクストラ領域制御 API 関数一覧

	API 関数名	概要
1	R_RFD_SetExtraEraseProtectReq	エクストラ領域シーケンサを起動し、ブロック消去禁止フラグの書き込みを開始します。
2	R_RFD_SetExtraWriteProtectReq	エクストラ領域シーケンサを起動し、書き込み禁止フラグの書き込みを開始します。
3	R_RFD_SetExtraBootAreaProtectReq	エクストラ領域シーケンサを起動し、ブート領域書き換え禁止フラグの書き込みを開始します。
4	R_RFD_SetExtraBootAreaReq	エクストラ領域シーケンサを起動し、ブート領域切替フラグの書き込みを開始します。
5	R_RFD_SetExtraFSWReq	エクストラ領域シーケンサを起動し、引数で指定されたフラッシュ・シールド・ウインドウの範囲の書き込みを開始します。

3.1.5 フック関数

RFD RL78 Type02 のフック関数一覧を表 3-5 に示します。

表 3-5 RFD RL78 Type02 フック関数一覧

	API 関数名	概要
1	R_RFD_HOOK_EnterCriticalSection	割り込み禁止命令を実行します。
2	R_RFD_HOOK_ExitCriticalSection	割り込み許可命令を実行します。

3.2 データ型定義

3.2.1 データ型

RFD RL78 Type02 のデータ型定義一覧を表 3-6 に示します。

表 3-6 RFD RL78 Type02 データ型定義一覧

Macro value	Type	Description
int8_t	signed char	1byte signed integer
uint8_t	unsigned char	1byte unsigned integer
int16_t	signed short	2byte signed integer
uint16_t	unsigned short	2byte unsigned integer
int32_t	signed long	4byte signed integer
uint32_t	unsigned long	4byte unsigned integer
bool	unsigned char	Boolean (false:0 / true:1)

補足：これらのデータ型は C 言語規格 C99 以降では標準整数型として stdint.h と stdbool.h に定義されています。

3.2.2 グローバル変数

RFD RL78 Type02 で使用するグローバル変数を以下に示します。

(1) g_u08_cpu_frequency

型 / 名称	uint8_t g_u08_cpu_frequency
初期値	0x00 (R_RFD_VALUE_U08_INIT_VARIABLE)
説明	CPU 動作周波数[2~40(MHz)] -（CPU 動作周波数 - 1）の値 : 0x01u-0x27u (1-39)
定義ファイル	r_rfd_common_api.c

(2) g_u08_fset_cpu_frequency

型 / 名称	uint8_t g_u08_fset_cpu_frequency
初期値	0x00 (R_RFD_VALUE_U08_INIT_VARIABLE)
説明	FSSET レジスタの FSET ビットへ設定する値
定義ファイル	r_rfd_common_api.c

(3) sg_u08_psw_ie_state

型 / 名称	static uint8_t sg_u08_psw_ie_state
初期値	0x00 (R_RFD_VALUE_U08_INIT_VARIABLE)
説明	PSW の割り込み許可フラグ(IE)の状態を退避/復帰するための保存データ - 割り込み禁止 : 0x00u - 割り込み許可 : 0x80u
定義ファイル	r_rfd_common_userown.c

注）初期値ありグローバル変数へ代入する初期値を ROM 上の Data セクションから RAM へコピーする処理はユーザが行う必要があります。

3.2.3 列挙型

- e_rfd_flash_memory_mode（列挙変数名 : e_rfd_flash_memory_mode_t）

フラッシュ・メモリ制御モード

Symbol Name	Value	Description
R_RFD_ENUM_ FLASH_MODE_CODE_PROGRAMMING	0x01	コード・フラッシュ・プログラミング・モード(Code flash programming mode)
R_RFD_ENUM_ FLASH_MODE_DATA_PROGRAMMING	0x02	データ・フラッシュ・プログラミング・モード(Data flash programming mode)
R_RFD_ENUM_ FLASH_MODE_CODE_TO_NONPROGRAMMABLE	0x03	非書き換えモード(Non-programmable mode) [コード・フラッシュ・プログラミング・モードから遷移]
R_RFD_ENUM_ FLASH_MODE_DATA_TO_NONPROGRAMMABLE	0x04	非書き換えモード(Non-programmable mode) [データ・フラッシュ・プログラミング・モードから遷移]

- e_rfd_df_access（列挙変数名 : e_rfd_df_access_t）

データ・フラッシュのアクセス制御

Symbol Name	Value	Description
R_RFD_ENUM_DF_ACCESS_DISABLE	0x00	データ・フラッシュ・アクセス禁止
R_RFD_ENUM_DF_ACCESS_ENABLE	0x01	データ・フラッシュ・アクセス許可

- e_rfd_boot_cluster（列挙変数名 : e_rfd_boot_cluster_t）

ブート・クラスタ番号

Symbol Name	Value	Description
R_RFD_ENUM_BOOT_CLUSTER_1	0x00	ブート・クラスタ 1
R_RFD_ENUM_BOOT_CLUSTER_0	0x01	ブート・クラスタ 0

- e_rfd_ret（列挙変数名 : e_rfd_ret_t）

戻り値

Symbol Name	Value	Description
R_RFD_ENUM_RET_STS_OK	0x00	正常終了
R_RFD_ENUM_RET_STS_BUSY	0x01	実行中
R_RFD_ENUM_RET_ERR_PARAMETER	0x10	パラメータ・エラー
R_RFD_ENUM_RET_ERR_MODE_MISMATCHED	0x11	モード不一致エラー

3.2.4 マクロ定義

3.2.4.1 RFD グローバル・データ設定用マクロ

- 16bit/8bit データ・マスク用マクロ

データの指定サイズ外の bit を 0 で AND してマスクします。

Symbol Name	Value	Description
R_RFD_VALUE_U08_MASK1_8BIT	0xFFu	8bit マスク値
R_RFD_VALUE_U16_MASK1_16BIT	0xFFFFu	16bit マスク値

- データ 16bit/8bit シフト用マクロ

32bit のデータを 16bit/8bit シフト、16bit のデータを 8bit シフトします。

Symbol Name	Value	Description
R_RFD_VALUE_U08_SHIFT_8BIT	8u	8bit シフト値
R_RFD_VALUE_U08_SHIFT_16BIT	16u	16bit シフト値

- 初期値設定用マクロ

グローバル変数の初期値を定義。

Symbol Name	Value	Description
R_RFD_VALUE_U08_INIT_VARIABLE	0x00u	グローバル変数の初期値

3.2.4.2 RL78/F23,F24 内蔵レジスタ、エクストラ領域設定用マクロ

- DFLCTL（データ・フラッシュ・コントロール・レジスタ）用マクロ

データ・フラッシュへのアクセス許可/禁止を設定します。

対象レジスタ定義 : R_RFD_REG_U08_DFLCTL（対象ビット[DFLEN] : R_RFD_REG_U01_DFLCTL_DFLEN）

Symbol Name	Value	Description
R_RFD_VALUE_U01_ DFLEN_DATA_FLASH_ACCESS_DISABLE	0u	データ・フラッシュのアクセス禁止
R_RFD_VALUE_U01_ DFLEN_DATA_FLASH_ACCESS_ENABLE	1u	データ・フラッシュのアクセス許可

- FLARS（フラッシュ領域選択レジスタ）用マクロ

アクセスの対象領域を指定する。

対象レジスタ定義 : R_RFD_REG_U08_FLARS

Symbol Name	Value	Description
R_RFD_VALUE_U08_FLARS_USER_AREA	0x00u	ユーザ領域指定
R_RFD_VALUE_U08_FLARS_EXTRA_AREA	0x01u	エクストラ領域指定

- FSSQ（フラッシュ・メモリ・シーケンサ制御レジスタ）用マクロ 1

フラッシュ・メモリ・シーケンサ起動時の各コマンドを定義。

[bit7] SQST : シーケンサの動作開始/停止ビットです。SQST=1 でシーケンサは動作開始します。

[bit2-0] SQMD2-0 :フラッシュ・メモリ・シーケンサの各コマンド

対象レジスタ定義 : R_RFD_REG_U08_FSSQ

Symbol Name	Value	Description
R_RFD_VALUE_U08_FSSQ_WRITE	0x81u	フラッシュ・メモリの書き込みコマンド
R_RFD_VALUE_U08_FSSQ_IVERIFY_CF	0x82u	コード・フラッシュ・メモリの内部ベリファイ・コマンド
R_RFD_VALUE_U08_FSSQ_IVERIFY_DF	0x8Au	データ・フラッシュ・メモリの内部ベリファイ・コマンド
R_RFD_VALUE_U08_FSSQ_BLANKCHECK_CF	0x83u	コード・フラッシュ・メモリのブランク・チェック・コマンド
R_RFD_VALUE_U08_FSSQ_BLANKCHECK_DF	0x8Bu	データ・フラッシュ・メモリのブランク・チェック・コマンド
R_RFD_VALUE_U08_FSSQ_ERASE	0x84u	フラッシュ・メモリの消去コマンド
R_RFD_VALUE_U08_FSSQ_CLEAR	0x00u	フラッシュ・メモリ・シーケンサ動作設定のクリア用設定値

- FSSQ(フラッシュ・メモリ・シーケンサ制御レジスタ)用マクロ 2

フラッシュ・メモリ・シーケンサ強制停止ビットの設定値を定義。

[bit6] FSSTP : シーケンサの強制停止ビットです。FSSTP=1 でシーケンサを強制停止します。

対象レジスタ定義 : R_RFD_REG_U01_FSSQ_FSSTP

Symbol Name	Value	Description
R_RFD_VALUE_U01_FSSQ_FSSTP_ON	1u	フラッシュ・メモリ・シーケンサ強制停止用設定ビット値

- FSSE(フラッシュ・エクストラ領域シーケンサ制御レジスタ)用マクロ

エクストラ領域シーケンサ起動時の各コマンドを定義。

[bit7] ESQST : シーケンサの動作開始/停止ビットです。ESQST=1 でシーケンサは動作開始します。

[bit2-0] ESQMD2-0 : エクストラ領域シーケンサの各コマンド

対象レジスタ定義 : R_RFD_REG_U08_FSSE

Symbol Name	Value	Description
R_RFD_VALUE_U08_FSSE_FSW	0x82u	フラッシュ・シールド・ウインドウ機能の設定コマンド
R_RFD_VALUE_U08_FSSE_SECURITY_FLAG	0x81u	セキュリティ・フラグの設定コマンド
R_RFD_VALUE_U08_FSSE_CLEAR	0x00u	エクストラ領域シーケンサ動作設定のクリア用設定値

- PFCMD(フラッシュ・プロテクト・コマンド・レジスタ)用マクロ

特定のレジスタへの書き込み動作に対してプロテクションを施すために使用するレジスタへ入力する固定値。

対象レジスタ定義 : R_RFD_REG_U08_PFCMD

Symbol Name	Value	Description
R_RFD_VALUE_U08_ PFCMD_SPECIFIC_SEQUENCE_WRITE	0xA5u	フラッシュ・メモリ・シーケンサの特定シーケンスでのプロテクション解除値

- PFS(特定シーケンス処理時のエラー)用マクロ

[bit0] FPRERR: 特定シーケンス処理時のエラーです。FPRERR =1 でプロテクション・エラーです。

対象レジスタ定義 : R_RFD_REG_U08_PFS

Symbol Name	Value	Description
R_RFD_VALUE_U08_MASK1_PFS_FPRERR	0x01u	特定シーケンス実行中のプロテクション・エラー比較値

- FLPMC（フラッシュ・プログラミング・モード・コントロール・レジスタ）用マクロ

フラッシュ・プログラミング・モードと非書き換えモードの移行制御に必要な値を定義。

対象レジスタ定義 : R_RFD_REG_U08_FLPMC

Symbol Name	Value	Description
R_RFD_VALUE_U08_FLPMC_MODE_ NONPROGRAMMABLE	0x08u	非書き換えモード
R_RFD_VALUE_U08_FLPMC_MODE_ CODE_FLASH_PROGRAMMING	0x82u	コード・フラッシュ・プログラミング・モード
R_RFD_VALUE_U08_FLPMC_MODE_ DATA_FLASH_PROGRAMMING	0x10u	データ・フラッシュ・プログラミング・モード

Symbol Name	Value	Description
R_RFD_VALUE_U08_FLPMC_ TRANSFER_1ST_LAYER	0x12u	コード・フラッシュ・プログラミング・モードへの移行と、コード・フラッシュ・プログラミング・モードから非書き換えモードへの移行制御用設定値
R_RFD_VALUE_U08_FLPMC_ TRANSFER_1ST_LAYER_INVERSION	0xEDu	コード・フラッシュ・プログラミング・モードへの移行と、コード・フラッシュ・プログラミング・モードから非書き換えモードへの移行制御用設定値 0x12u を反転した値
R_RFD_VALUE_U08_FLPMC_ TRANSFER_2ND_LAYER	0x92u	コード・フラッシュ・プログラミング・モードへの移行と、コード・フラッシュ・プログラミング・モードから非書き換えモードへの移行制御用設定値
R_RFD_VALUE_U08_FLPMC_ TRANSFER_2ND_LAYER_INVERSION	0x6Du	コード・フラッシュ・プログラミング・モードへの移行と、コード・フラッシュ・プログラミング・モードから非書き換えモードへの移行制御用設定値 0x92u を反転した値

- FSASTH（フラッシュ・メモリ・シーケンサ・ステータス・レジスタ: High 8bit）用マクロ

フラッシュ・メモリ・シーケンサ（エクストラ領域シーケンサ、またはコード/データ・フラッシュ領域シーケンサ）の終了ステータスを定義。

[bit7] ESQEND : エクストラ領域シーケンサの終了ステータスです。ESQEND=1 でシーケンサは動作完了です。ESQST ビットのクリアでクリアされます。

[bit6] SQEND : コード/データ・フラッシュ領域シーケンサの終了ステータスです。SQEND=1 でシーケンサは動作完了です。SQST ビットのクリアでクリアされます。

対象レジスタ定義 : R_RFD_REG_U08_FSASTH

Symbol Name	Value	Description
R_RFD_VALUE_U08_MASK1_FSASTH_SQEND	0x40u	コード/データ・フラッシュ領域シーケンサの終了比較値
R_RFD_VALUE_U08_MASK1_FSASTH_ESQEND	0x80u	エクストラ領域シーケンサの終了比較値

- FSASTL（フラッシュ・メモリ・シーケンサ・ステータス・レジスタ: Low 8bit）用マクロ

フラッシュ・メモリ・シーケンサ（エクストラ領域シーケンサ、またはコード/データ・フラッシュ領域シーケンサ）終了時のエラー・ステータス・マスク値を定義。

[bit5] ESEQER : エクストラ領域シーケンサのエラーです。ESEQER =1 でシーケンサ・エラーです。

[bit4] SEQER : コード/データ・フラッシュ領域シーケンサのエラーです。SEQER =1 でシーケンサ・エラーです。

[bit3] BLER : ブランク・チェック・コマンドのエラーです。BLER =1 でブランク・エラーです。

[bit2] IVER : 内部ベリファイ・コマンドのエラーです。IVER =1 で内部ベリファイ・エラーです。

[bit1] WRER : 書き込みコマンドのエラーです。WRER =1 で書き込みエラーです。

[bit0] ERER : ブロック消去コマンドのエラーです。ERER =1 で消去エラーです。

対象レジスタ定義 : R_RFD_REG_U08_FSASTL

Symbol Name	Value	Description
R_RFD_VALUE_U08_ MASK1_FSASTL_ERROR_FLAG	0x3Fu	フラッシュ・メモリ・シーケンサ（エクストラ領域シーケンサ、またはコード/データ・フラッシュ領域シーケンサ）終了時のエラー・ステータス・マスク値

- FSSET（フラッシュ・メモリ・シーケンサ初期設定レジスタ）用マクロ 1

ブート・スワップ指定、テンポラリ・ブート・スワップ設定、もしくはそれ以外の bit を 0 で AND してマスクします。

[bit7] TMSPMD : ブート・スワップ指定。TMSPMD=0 でブート・スワップはエクストラ領域の情報に従います。TMSPMD=1 でブート・スワップは TMBTSEL ビットに従います。

[bit6] TMBTSEL : テンポラリ・ブート・スワップ設定。TMBTSEL=0 でブート領域にブート・クラスタ 0 を指定。TMBTSEL=1 でブート領域にブート・クラスタ 1 を指定。

対象レジスタ定義 : R_RFD_REG_U08_FSSET

Symbol Name	Value	Description
R_RFD_VALUE_U08_MASK1_ FSSET_TMSPMD_AND_TMBTSEL	0xC0u	ブート・スワップ指定、テンポラリ・ブート・スワップ設定以外をマスク
R_RFD_VALUE_U08_MASK0_ FSSET_TMSPMD_AND_TMBTSEL	0x3Fu	ブート・スワップ指定、テンポラリ・ブート・スワップ設定をマスク
R_RFD_VALUE_U08_MASK1_FSSET_TMSPMD	0x80u	ブート・スワップ指定以外をマスク
R_RFD_VALUE_U08_ FSSET_BOOT_CLUSTER_0	0x80u	テンポラリ・ブート・スワップ、ブート・クラスタ 0 の設定値
R_RFD_VALUE_U08_ FSSET_BOOT_CLUSTER_1	0xC0u	テンポラリ・ブート・スワップ、ブート・クラスタ 1 の設定値

- FSSET（フラッシュ・メモリ・シーケンサ初期設定レジスタ）用マクロ２

フラッシュ・メモリ・シーケンサの動作周波数範囲。FSSET レジスタ設定値変換用補正値(-1)、および FSSET レジスタ設定値変換用補正値シフト値。

[bit4-0] FSET4-0 : FSSET レジスタ用に変換した CPU 動作周波数を入力します。

対象レジスタ定義 : R_RFD_REG_U08_FSSET

Symbol Name	Value	Description
R_RFD_VALUE_U08_FREQUENCY_LOWER_LIMIT	2u	入力可能最小動作周波数(2MHz)
R_RFD_VALUE_U08_FREQUENCY_UPPER_LIMIT	40u	入力可能最大動作周波数(40MHz)
R_RFD_VALUE_U08_FREQUENCY_ADJUST	1u	FSSETレジスタ設定値変換用補正値(-1)
R_RFD_VALUE_U08_FREQUENCY_SHIFT_ADJUST	1u	FSSETレジスタ設定値変換用補正値シフト値
R_RFD_VALUE_U08_FREQUENCY_ CALC_THRESHOLD	23u	FSSETレジスタ設定値計算用しきい値

- FLRST（フラッシュ初期化レジスタ）用マクロ

エクストラ領域シーケンサ/フラッシュ・メモリ・シーケンサ・レジスタの初期化実行を設定する値を定義。

[bit0] FLRST : FLRST(bit)=1 で、エクストラ領域シーケンサ/フラッシュ・メモリ・シーケンサ・レジスタの初期化を実行します。

対象レジスタ定義 : R_RFD_REG_U08_FLRST

Symbol Name	Value	Description
R_RFD_VALUE_U08_FLRST_ON	0x01u	シーケンサ・レジスタの初期化実行設定値
R_RFD_VALUE_U08_FLRST_OFF	0x00u	シーケンサ・レジスタの初期化非実行設定値

- FLFSWS/FLFSWE（フラッシュ FSW モニタ・レジスタ START/END）用マクロ

FSW の設定状態取得、および設定用のマスク値を定義。

FLFSWE[bit9-0] : FSW エンド・ブロック番号+1

FLFSWS[bit9-0] : FSW スタート・ブロック番号

対象レジスタ定義 : R_RFD_REG_U16_FLFSWE / R_RFD_REG_U16_FLFSWS

(1) FSW の設定状態取得用のマスク値を定義。

Symbol Name	Value	Description
R_RFD_VALUE_U16_MASK1_FLFSW_BLOCK_NUMBER	0x03FFu	ブロック番号設定マスク値

(2) FSW 設定用のマスク値を定義。

Symbol Name	Value	Description
R_RFD_VALUE_U16_MASK1_FSW_BLOCK_INFO	0x03FFu	FSW ブロック指定用マスク値

- FLAPH/FLAPL, FLSEDH/FLSEDL（フラッシュ・アドレス・ポインタ・レジスタ HIGH/LOW）用マクロ

(1) データ・フラッシュ・メモリ消去（1 ブロック:1Kbyte）、ブランク・チェック用の先頭/終了アドレスを定義。

FLAPH[bit3-0] : FLAP19-16 は、データ・フラッシュ・メモリ領域の先頭上位アドレス設定値。0x0F 固定値。

FLAPL[bit15-0] : FLAP15-0 は、データ・フラッシュ・メモリ領域の先頭下位アドレス設定値。

FLSEDH[bit3-0] : EWA19-16 は、データ・フラッシュ・メモリ領域の終了上位アドレス設定値。0x0F 固定値。

FLSEDL[bit15-0] : EWA15-0 は、データ・フラッシュ・メモリ領域の終了下位アドレス設定値。

対象レジスタ定義 : R_RFD_REG_U08_FLAPH / R_RFD_REG_U16_FLAPL
R_RFD_REG_U08_FLSEDH / R_RFD_REG_U16_FLSEDL

Symbol Name	Value	Description
R_RFD_VALUE_U16_ DATA_FLASH_ADDR_LOW	0x1000u	データ・フラッシュ領域先頭の下位アドレス設定値(16bit)
R_RFD_VALUE_U08_ DATA_FLASH_ADDR_HIGH	0x0Fu	データ・フラッシュ領域先頭の上位アドレス設定値(8bit)
R_RFD_VALUE_U08_ DATA_FLASH_BLOCK_ADDR_LOW	0x3Fu	データ・フラッシュ・ブロック先頭の下位アドレス・マスク値(8bit)
R_RFD_VALUE_U16_ DATA_FLASH_BLOCK_ADDR_END	0x03FFu	データ・フラッシュ・ブロック終了の下位アドレス設定値(16bit)
R_RFD_VALUE_U08_ DATA_FLASH_SHIFT_LOW_ADDR	10u	ブロック番号からデータ・フラッシュ領域オフセット算出用下位アドレス・シフト値

(2) コード・フラッシュ・メモリ消去、ブランク・チェック（1 ブロック:1Kbyte）用の先頭/終了アドレスを定義。

FLAPH[bit3-0] : FLAP19-16 は、コード・フラッシュ・メモリ領域の先頭上位アドレス設定値。

FLAPL[bit15-0] : FLAP15-0 は、コード・フラッシュ・メモリ領域の先頭下位アドレス設定値。

FLSEDH[bit3-0] : EWA19-16 は、コード・フラッシュ・メモリ領域の終了上位アドレス設定値。

FLSEDL[bit15-0] : EWA15-0 は、コード・フラッシュ・メモリ領域の終了下位アドレス設定値。

対象レジスタ定義 : R_RFD_REG_U08_FLAPH / R_RFD_REG_U16_FLAPL
R_RFD_REG_U08_FLSEDH / R_RFD_REG_U16_FLSEDL

Symbol Name	Value	Description
R_RFD_VALUE_U16_ CODE_FLASH_BLOCK_ADDR_LOW	0x003Fu	コード・フラッシュ・ブロック先頭の下位アドレス・マスク値(16bit)
R_RFD_VALUE_U16_ CODE_FLASH_BLOCK_ADDR_HIGH	0x03C0u	コード・フラッシュ・ブロック先頭の上位アドレス・マスク値(16bit -> シフト後下位 8bit のみ使用)
R_RFD_VALUE_U16_ CODE_FLASH_BLOCK_ADDR_END	0x03FFu	コード・フラッシュ・ブロック終了の下位 1KB 単位のアドレス設定値(16bit)
R_RFD_VALUE_U08_ CODE_FLASH_SHIFT_LOW_ADDR	10u	ブロック番号からコード・フラッシュ領域オフセット算出用下位アドレス・シフト値
R_RFD_VALUE_U08_ CODE_FLASH_SHIFT_HIGH_ADDR	6u	ブロック番号からコード・フラッシュ領域オフセット算出用上位アドレス・シフト値

(例) ブロック番号: 107 -> 0x006B

R_RFD_VALUE_U16_CODE_FLASH_BLOCK_ADDR_LOW : 0x002B -> 0xAC00（10 ビット左へシフト）

R_RFD_VALUE_U16_CODE_FLASH_BLOCK_ADDR_HIGH : 0x0040 -> 0x0001（6 ビット右へシフト）

ブロック先頭アドレス : 0x0001_AC00

- FLSEC(フラッシュ・セキュリティ・フラグ・モニタ・レジスタ)用マクロ

エクストラ領域設定データ、セキュリティモニタ用マスク・データを定義。

[bit12] WRPR : 書き込み禁止フラグ。WRPR=0 で書き込み禁止。

[bit10] SEPR : ブロック消去禁止フラグ。SEPR=0 でブロック消去禁止。

[bit9] BTPR : ブート領域書き換え禁止制御フラグ。BTPR=0 でブート領域書き換え禁止。

[bit8] BTFLG : ブート領域切替フラグ。

BTFLG = 0 : ブート領域は、ブート・クラスタ 1。

BTFLG = 1 : ブート領域は、ブート・クラスタ 0。

対象レジスタ定義 : R_RFD_REG_U16_FLWH, R_RFD_REG_U16_FLWL, R_RFD_REG_U16_FLSEC

Symbol Name	Value	Description
R_RFD_VALUE_U16_MASK0_ERASE_PROTECT_FLAG	0xFBFFu	ブロック消去禁止設定用マスク値
R_RFD_VALUE_U16_MASK0_WRITE_PROTECT_FLAG	0xEFFFu	書き込み禁止設定用マスク値
R_RFD_VALUE_U16_MASK0_ BOOT_CLUSTER_PROTECT_FLAG	0xFDFFu	ブート領域書き換え禁止設定用マスク値
R_RFD_VALUE_U16_MASK0_BOOT_FLAG	0xFEFFu	ブート領域切替フラグ設定、モニタ用マスク値
R_RFD_VALUE_U16_MASK1_BOOT_FLAG	0x0100u	ブート領域切替フラグ・モニタ用マスク値

3.3 API 関数仕様

この章では、Renesas Flash Driver (RFD) RL78 Type02 の API 関数の詳細仕様について説明します。
RFD RL78 Type02 の API 関数を使用して、フラッシュ・メモリの書き換えを実施する上での前提条件があります。この前提条件と異なる条件で RFD RL78 Type02 の API 関数を使用した場合、各関数の動作が不定となる可能性がありますので、ご注意ください。

≪前提条件≫

- R_RFD_Init()関数は、全ての RFD 関数を使用する前に、1 回実行してください。
- セルフ・プログラミング実行中は、高速オンチップ・オシレータを起動しておく必要があります。RFD RL78 Type02 の全ての API 関数は、高速オンチップ・オシレータが起動している状態で実行してください。
- データ・フラッシュを操作する場合、データ・フラッシュへのアクセスを許可した状態で RFD RL78 Type02 の API を実行してください。データ・フラッシュへのアクセス許可方法については、対象となる RL78 マイクロコントローラのユーザーズマニュアルを参照してください。

以下に API 関数仕様の記述例を示します。
≪API 関数仕様の記述例≫
Information

Syntax	この関数を C 言語で記述されたプログラムから呼び出す際の書式を示します。	
Reentrancy	再帰可否 ：Reentrant（再帰可能）、または Non-Reentrant（再起不可）。	
Parameters (IN)	この関数の引数（入力）。	引数 [値、範囲、引数の意味等]
Parameters (IN/OUT)	この関数の引数（入出力）。	引数 [値、範囲、引数の意味等]
Parameters (OUT)	この関数の引数（出力）。	引数 [値、範囲、引数の意味等]
Return Value	この関数からの戻り値の型 （列挙型、ポインタ等）	戻り値の列挙子（定数）：値 [定数の意味：詳細説明]
		戻り値の列挙子（定数）：値 [定数の意味：詳細説明]
Description	機能概要	
Preconditions	事前条件の概要	
Remarks	特記事項	

動作概要：

この関数の機能概要を示します。

備考：

この関数の使用条件や制限事項を示します。

3.3.1 共通フラッシュ制御 API 関数仕様

RFD RL78 Type02 の共通フラッシュ制御関数を示します。

3.3.1.1 R_RFD_Init

Information

Syntax	R_RFD_FAR_FUNC e_rfd_ret_t R_RFD_Init(unit8_t i_u08_cpu_frequency);	
Reentrancy	Non-Reentrant	
Parameters (IN)	unit8_t i_u08_cpu_frequency	CPU 動作周波数 [2~40(MHz)]
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	e_rfd_ret_t	R_RFD_ENUM_RET_STS_OK : 0x00 [正常終了：周波数が範囲内]
		R_RFD_ENUM_RET_ERR_PARAMETER : 0x10 [パラメータ・エラー：周波数が範囲外]
Description	引数で指定された周波数をフラッシュ・メモリ・シーケンサに設定し、RFD RL78 Type02 の初期化を行います。	
Preconditions	非書き換えモードで実行してください。高速オンチップ・オシレータを起動している状態で実行してください。	
Remarks	全ての RFD 関数を使用する前に、1 回実行してください。	

動作概要：

・引数（i_u08_cpu_frequency）が 2~40(MHz)の範囲内であるかを確認し、範囲内であれば、(i_u08_cpu_frequency -1)を(g_u08_cpu_frequency)に設定します 。

・FSSET レジスタ用に変換した CPU 動作周波数の値を(g_u08_fset_cpu_frequency)に設定します。

- 引数（i_u08_cpu_frequency）が、しきい値(23MHz)以下の場合
  (i_u08_cpu_frequency -1)を(g_u08_fset_cpu_frequency)に設定します。
- 引数（i_u08_cpu_frequency）が、しきい値(23MHz)を超える場合
  (i_u08_cpu_frequency + R_RFD_VALUE_U08_FREQUENCY_CALC_THRESHOLD)を 1bit 右シフトした値を(g_u08_fset_cpu_frequency)に設定します。

備考：

・セルフ・プログラミング実行中は、高速オンチップ・オシレータを起動しておく必要があります。高速オンチップ・オシレータが起動している状態で、本関数を実行してください。
※RFD Type02 for RL78 では、高速オンチップ・オシレータの起動やチェックは行っていません。

・引数(i_u08_cpu_frequency)には、実際に CPU が動作する周波数の値の小数点以下を切り上げた整数値を設定します。（例：CPU が動作する周波数が 4.5MHz の場合は、初期化関数で 5 を設定してください）
CPU の動作周波数を 4 MHz 未満で使用する場合は、2 MHz, 3 MHz を使用することができます。その際、整数値でない周波数（2.5MHz など）は使用できません。
引数(i_u08_cpu_frequency)に設定する周波数は、フラッシュ書き換え時、実際に CPU が動作する周波数であり、必ずしも高速オンチップ・オシレータの周波数を設定するということではありません。

- CPU 動作周波数と異なる値を指定した場合、その後の動作は不定となります。その際、フラッシュの書き換えが完了した場合でも、データの値、及びその後の保持期間を満たすことができない可能性があります。
  ※CPU 動作周波数の範囲については、対象となる RL78 マイクロコントローラのユーザーズマニュアルを参照してください。

・非書き換えモード以外で本関数を実行した場合、その後の動作は不定となります。

3.3.1.2 R_RFD_SetDataFlashAccessMode

Information

Syntax	R_RFD_FAR_FUNC void R_RFD_SetDataFlashAccessMode (e_rfd_df_access_t i_e_df_access);	
Reentrancy	Non-Reentrant	
Parameters (IN)	e_rfd_df_access_t i_e_df_access	データ・フラッシュのアクセス制御 R_RFD_ENUM_DF_ACCESS_ENABLE : 0x01 [データ・フラッシュ・アクセス許可] R_RFD_ENUM_DF_ACCESS_DISABLE : 0x00 [データ・フラッシュ・アクセス禁止]
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	N/A	
Description	引数で指定されたデータ・フラッシュへのアクセスの許可、または、禁止を設定します。	
Preconditions	非書き換えモードで実行してください。	
Remarks	-	

動作概要：

・引数(i_e_df_access)が R_RFD_ENUM_DF_ACCESS_DISABLE の場合、DFLEN(DFLCTL の bit0)=’0’ (R_RFD_VALUE_U01_DFLEN_DATA_FLASH_ACCESS_DISABLE) データ・フラッシュ・アクセス禁止状態を設定します。

・引数(i_e_df_access)が R_RFD_ENUM_DF_ACCESS_ENABLE の場合、DFLEN(DFLCTL の bit0)=’1’ (R_RFD_VALUE_U01_DFLEN_DATA_FLASH_ACCESS_ENABLE) データ・フラッシュ・アクセス許可状態を設定します。

・セットアップ時間をウエイトします。（セットアップ時間: 4μsec）セットアップ時間のウエイト完了後、データ・フラッシュへのアクセスが可能となります。

備考：

・引数(i_e_df_access)に R_RFD_ENUM_DF_ACCESS_DISABLE、R_RFD_ENUM_DF_ACCESS_ENABLE の値以外を指定した場合、DFLEN(DFLCTL の bit0)=’0’ (R_RFD_VALUE_U01_DFLEN_DATA_FLASH_ACCESS_DISABLE)データ・フラッシュ・アクセス禁止状態を設定します。

・非書き換えモード以外で本関数を実行した場合、その後の動作は不定となります。

3.3.1.3 R_RFD_SetFlashMemoryMode

Information

Syntax	R_RFD_FAR_FUNC e_rfd_ret_t R_RFD_SetFlashMemoryMode (e_rfd_flash_memory_mode_t i_e_flash_mode);	
Reentrancy	Non-Reentrant	
Parameters (IN)	e_rfd_flash_memory_mode_t i_e_flash_mode	フラッシュ・メモリ制御モード
		R_RFD_ENUM_FLASH_MODE_CODE_PROGRAMMING : 0x01 [コード・フラッシュ・プログラミング・モード] R_RFD_ENUM_FLASH_MODE_DATA_PROGRAMMING : 0x02 [データ・フラッシュ・プログラミング・モード] R_RFD_ENUM_FLASH_MODE_CODE_TO_NONPROGRAMMABLE : 0x03 [非書き換えモード(コード・フラッシュ・プログラミング・モードから遷移)] R_RFD_ENUM_FLASH_MODE_DATA_TO_NONPROGRAMMABLE : 0x04 [非書き換えモード(データ・フラッシュ・プログラミング・モードから遷移)]
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	e_rfd_ret_t	R_RFD_ENUM_RET_STS_OK : 0x00 [正常終了]
		R_RFD_ENUM_RET_ERR_MODE_MISMATCHED : 0x11 [モード不一致エラー](指定モードへ設定されなかった)
Description	フラッシュ・メモリ・シーケンサを引数で指定されたフラッシュ・メモリ制御モードへ変更後、CPU 動作周波数の値を設定します。	
Preconditions	コード/データ・フラッシュ領域シーケンサ、エクストラ領域シーケンサのコマンドを実行していない状態で、本関数を実行してください。	
Remarks	-	

動作概要：

・フック関数[R_RFD_HOOK_EnterCriticalSection()]を呼び出し、それまでの割り込み設定(禁止[DI]/許可[EI])を退避すると共に、割り込みを禁止に設定します。

・引数(i_e_flash_mode)の値に応じて FLPMC レジスタ値を設定、指定されたフラッシュ・メモリ制御モードに移行します。

・モード移行時 Wait(tMS)を実施する。Wait 時間(tMS)は、デバイスのハードウェアマニュアルでご確認ください。

・フック関数[R_RFD_HOOK_ExitCriticalSection()]を呼び出し、割り込み設定(禁止[DI]/許可[EI])を復帰します。

・R_RFD_Init 関数実行時に設定された、”g_u08_fset_cpu_frequency”を FSSET レジスタに設定します。

備考：

・フック関数[R_RFD_HOOK_EnterCriticalSection()]の呼び出しから[R_RFD_HOOK_ExitCriticalSection()]の呼び出しまでは、本関数の割り込み禁止区間です。割り込みを許可してこの区間で割り込みを発生させた場合、その後の動作は不定となります。

・引数へフラッシュ・メモリ制御モード以外の値を指定した場合は、非書き換えモード（データ・フラッシュ・プログラミング・モードからの遷移）を設定した場合と同一の処理を行います。

・R_RFD_Init 関数を実行せずに本関数を実行した場合、RFD の各書き換え処理が正常に実施されても、そのデータの値は保証の対象外となります。RFD RL78 Type02 を使用する場合、全ての RFD 関数を使用する前に、必ず、R_RFD_Init()関数を 1 回実行してください。

・コード・フラッシュ・プログラミング・モード、およびデータ・フラッシュ・プログラミング・モードへは、非書き換えモードから遷移してください。

3.3.1.4 R_RFD_CheckFlashMemoryMode

Information

Syntax	R_RFD_FAR_FUNC e_rfd_ret_t R_RFD_CheckFlashMemoryMode (e_rfd_flash_memory_mode_t i_e_flash_mode);	
Reentrancy	Non-Reentrant	
Parameters (IN)	e_rfd_flash_memory_mode_t i_e_flash_mode	フラッシュ・メモリ制御モード
		R_RFD_ENUM_FLASH_MODE_CODE_PROGRAMMING : 0x01 [コード・フラッシュ・プログラミング・モード] R_RFD_ENUM_FLASH_MODE_DATA_PROGRAMMING : 0x02 [データ・フラッシュ・プログラミング・モード] R_RFD_ENUM_FLASH_MODE_CODE_TO_NONPROGRAMMABLE : 0x03 [非書き換えモード] R_RFD_ENUM_FLASH_MODE_DATA_TO_NONPROGRAMMABLE : 0x04 [非書き換えモード]
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	e_rfd_ret_t	R_RFD_ENUM_RET_STS_OK : 0x00 [正常終了]
		R_RFD_ENUM_RET_ERR_MODE_MISMATCHED : 0x11 [モード不一致エラー]
Description	引数で指定されたモードであるかどうかを確認します。	
Preconditions	コード/データ・フラッシュ領域シーケンサ、エクストラ領域シーケンサのコマンドを実行していない状態で、本関数を実行してください。	
Remarks	-	

動作概要：

・FLPMC レジスタの値を読み出し、その値が引数(i_e_flash_mode)で指定したモードのレジスタ値と等しいかを確認します。
- 非書き換えモード：0x08
- コード・フラッシュ・プログラミング・モード：0x82
- データ・フラッシュ・プログラミング・モード：0x10

備考：

・R_RFD_SetFlashMemoryMode()以外でフラッシュ・メモリ・シーケンサのモードを設定した場合、本関数は正しく動作しない可能性があります。

・コード/データ・フラッシュ領域シーケンサ、エクストラ領域シーケンサのコマンド実行中に、本関数を実行した場合、その後の動作は不定となります。

・引数へフラッシュ・メモリ制御モード以外の値を指定した場合は、非書き換えモードを設定した場合と同一の処理を行います。

3.3.1.5 R_RFD_CheckCFDFSeqEndStep1

Information

Syntax	R_RFD_FAR_FUNC e_rfd_ret_t R_RFD_CheckCFDFSeqEndStep1(void);	
Reentrancy	Non-Reentrant	
Parameters (IN)	N/A	
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	e_rfd_ret_t	R_RFD_ENUM_RET_STS_OK : 0x00 [正常終了]
		R_RFD_ENUM_RET_STS_BUSY : 0x01 [シーケンサ・コマンド動作中]
Description	起動したコード/データ・フラッシュ領域シーケンサの動作終了を確認します。	
Preconditions	コード/データ・フラッシュ領域シーケンサを起動するコマンド開始後に実行してください。	
Remarks	戻り値が R_RFD_STS_BUSY である間は、本関数を再度実行してください。 本関数で R_RFD_ENUM_RET_STS_OK を確認後、関数 R_RFD_CheckCFDFSeqEndStep2()を実行してください。	

動作概要：

・起動したコード/データ・フラッシュ領域シーケンサの動作が終了[SQEND(FSASTH の bit6) = 1]したかどうかを確認します。

・コード/データ・フラッシュ領域シーケンサの動作が終了していた場合、フラッシュ・メモリ・シーケンサ制御レジスタをクリア(FSSQ = 0x00)し、R_RFD_ENUM_RET_STS_OK を返します。
終了していなかった場合は、R_RFD_ENUM_RET_STS_BUSY を返します。

備考：

・戻り値が R_RFD_STS_BUSY である間は、本関数を再度実行してください。

・コード/データ・フラッシュ領域シーケンサを起動するコマンド開始後以外で、本関数を実行した場合、正常に動作しません。

・本関数で R_RFD_ENUM_RET_STS_OK を確認後、関数 R_RFD_CheckCFDFSeqEndStep2()を実行してください。

3.3.1.6 R_RFD_CheckExtraSeqEndStep1

Information

Syntax	R_RFD_FAR_FUNC e_rfd_ret_t R_RFD_CheckExtraSeqEndStep1(void);	
Reentrancy	Non-Reentrant	
Parameters (IN)	N/A	
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	e_rfd_ret_t	R_RFD_ENUM_RET_STS_OK : 0x00 [正常終了]
		R_RFD_ENUM_RET_STS_BUSY : 0x01 [シーケンサ・コマンド動作中]
Description	起動したエクストラ領域シーケンサの動作終了を確認します。	
Preconditions	エクストラ領域シーケンサを起動するコマンド開始後に実行してください。	
Remarks	戻り値が R_RFD_STS_BUSY である間は、本関数を再度実行してください。 本関数で R_RFD_ENUM_RET_STS_OK を確認後、関数 R_RFD_CheckExtraSeqEndStep2()を実行してください。	

動作概要：

・起動したエクストラ領域シーケンサの動作が終了[ESQEND(FSASTH の bit7) = 1]したかどうかを確認します。

・エクストラ領域シーケンサの動作が終了していた場合、エクストラ領域シーケンサ制御レジスタをクリア(FSSE = 0x00)し、R_RFD_ENUM_RET_STS_OK を返します。

終了していなかった場合は、R_RFD_ENUM_RET_STS_BUSY を返します。

備考：

・戻り値が R_RFD_STS_BUSY である間は、本関数を再度実行してください。

・Extra 領域シーケンサを起動するコマンド開始後以外で、本関数を実行した場合、正常に動作しません。

・本関数で R_RFD_ENUM_RET_STS_OK を確認後、関数 R_RFD_CheckExtraSeqEndStep2()を実行してください。

3.3.1.7 R_RFD_CheckCFDFSeqEndStep2

Information

Syntax	R_RFD_FAR_FUNC e_rfd_ret_t R_RFD_CheckCFDFSeqEndStep2(void);	
Reentrancy	Non-Reentrant	
Parameters (IN)	N/A	
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	e_rfd_ret_t	R_RFD_ENUM_RET_STS_OK : 0x00 [正常終了 ： シーケンサ動作終了]
		R_RFD_ENUM_RET_STS_BUSY : 0x01 [シーケンサ動作中]
Description	フラッシュ・メモリ・シーケンサ制御レジスタのクリアにより、コマンド動作が終了したかどうかを確認します。	
Preconditions	関数 R_RFD_CheckCFDFSeqEndStep1()で、R_RFD_ENUM_RET_STS_OK 確認後に実行してください。	
Remarks	戻り値が R_RFD_STS_BUSY である間は、本関数を再度実行してください。	

動作概要：

・フラッシュ・メモリ・シーケンサ制御レジスタをクリア(FSSQ = 0x00)により、コード/データ・フラッシュ領域シーケンサ・コマンドの動作が全て終了[SQEND(FSASTH の bit6) = 0]したかどうかを確認します。

・コード/データ・フラッシュ領域シーケンサ・コマンドの動作が終了していた場合、
R_RFD_ENUM_RET_STS_OK を返します。
終了していなかった場合、R_RFD_ENUM_RET_STS_BUSY を返します。

備考：

・戻り値が R_RFD_STS_BUSY である間は、本関数を再度実行してください。

・R_RFD_CheckCFDFSeqEndStep1()で、R_RFD_ENUM_RET_STS_OK 確認後以外で、本関数を実行した場合、正常に動作しません。

3.3.1.8 R_RFD_CheckExtraSeqEndStep2

Information

Syntax	R_RFD_FAR_FUNC e_rfd_ret_t R_RFD_CheckExtraSeqEndStep2(void);	
Reentrancy	Non-Reentrant	
Parameters (IN)	N/A	
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	e_rfd_ret_t	R_RFD_ENUM_RET_STS_OK : 0x00 [正常終了 ： シーケンサ動作終了]
		R_RFD_ENUM_RET_STS_BUSY : 0x01 [シーケンサ動作中]
Description	エクストラ領域シーケンサ制御レジスタのクリアにより、コマンド動作が終了したかどうかを確認します。	
Preconditions	関数 R_RFD_CheckExtraSeqEndStep1()で、R_RFD_ENUM_RET_STS_OK 確認後に使用してください。	
Remarks	戻り値が R_RFD_STS_BUSY である間は、本関数を再度実行してください。	

動作概要：

・エクストラ領域シーケンサ制御レジスタをクリア(FSSE = 0x00)により、エクストラ領域シーケンサ・コマンドの動作が全て終了[ESQEND(FSASTH の bit7) = 0]したかどうかを確認します。

・エクストラ領域シーケンサ・コマンドの動作が終了していた場合、R_RFD_ENUM_RET_STS_OK を返します。終了していなかった場合、R_RFD_ENUM_RET_STS_BUSY を返します。

備考：

・戻り値が R_RFD_STS_BUSY である間は、本関数を再度実行してください。

・R_RFD_CheckExtraSeqEndStep1()で、R_RFD_ENUM_RET_STS_OK 確認後以外で、本関数を実行した場合、正常に動作しません。

3.3.1.9 R_RFD_GetSeqErrorStatus

Information

Syntax	R_RFD_FAR_FUNC void R_RFD_GetSeqErrorStatus (uint8_t __near * onp_u08_error_status);	
Reentrancy	Non-Reentrant	
Parameters (IN)	N/A	
Parameters (IN/OUT)	N/A	
Parameters (OUT)	uint8_t __near * onp_u08_error_status	発生したエラー情報を格納する変数へのポインタ
Return Value	N/A	
Description	コード/データ・フラッシュ領域シーケンサ・コマンド、または、エクストラ領域シーケンサ・コマンドにより、発生したエラー情報を取得します。	
Preconditions	コード/データ・フラッシュ領域シーケンサ、エクストラ領域シーケンサのコマンドを実行していない状態で、本関数を実行してください。	
Remarks	-	

動作概要：

・FSASTL レジスタの値(8bit)を読み出し、bit5-0 の値を引数ポインタ(onp_u08_error_status)が示す変数に格納します。

※bit7,6 は固定値(0)を設定する。

取得するエラー情報（FSASTL レジスタ bit 5-0 の 6bit）

[bit5：エクストラ領域シーケンサ・エラー]

[bit4：コード/データ・フラッシュ領域シーケンサ・エラー]

[bit3：ブランク・チェック・コマンド・エラー]

[bit2：内部ベリファイ・コマンド・エラー]

[bit1：書き込みコマンド・エラー]

[bit0：消去コマンド・エラー]

備考：

・コード/データ・フラッシュ領域シーケンサ、エクストラ領域シーケンサのコマンド実行中に、本関数を実行した場合、正しい値を取得できません。

3.3.1.10 R_RFD_ClearSeqRegister

Information

Syntax	R_RFD_FAR_FUNC void R_RFD_ClearSeqRegister(void);	
Reentrancy	Non-Reentrant	
Parameters (IN)	N/A	
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	N/A	
Description	コード/データ・フラッシュ領域シーケンサ、エクストラ領域シーケンサ制御を行うレジスタをクリアします。	
Preconditions	コード・フラッシュ・プログラミング・モード、または、データ・フラッシュ・プログラミング・モードで使用してください。 コード/データ・フラッシュ領域シーケンサ、エクストラ領域シーケンサのコマンドを実行していない状態で使用してください。	
Remarks	R_RFD_CheckCFDFSeqEndStep2()、または、 R_RFD_CheckExtraSeqEndStep2()実行後に、本関数を実行してください。	

動作概要：

・フラッシュ初期化レジスタ[FLRST]に 0x01 を設定した後、FLRST レジスタに 0x00 を設定します。対象レジスタがクリアされます。
- 対象となるコード/データ・フラッシュ領域シーケンサ、エクストラ領域シーケンサ制御を行うレジスタ：
FLAPH/L, FLSEDH/L, FLWH/L, FLARS, FSSQ, FSSE

備考：

・本関数を実行しても、フラッシュ・メモリ・シーケンサ・コマンドにより発生したエラー情報（FSASTL レジスタ）はクリアされません。
・コード/データ・フラッシュ領域シーケンサ、エクストラ領域シーケンサ実行中に、本関数を実行した場合、その後の動作は不定となります。
・コード・フラッシュ・プログラミング・モード、または、データ・フラッシュ・プログラミング・モード以外で、本関数を実行した場合、その後の動作は不定となります。

3.3.1.11 R_RFD_ForceStopSeq

Information

Syntax	R_RFD_FAR_FUNC void R_RFD_ForceStopSeq(void);	
Reentrancy	Non-Reentrant	
Parameters (IN)	N/A	
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	N/A	
Description	コード/データ・フラッシュ領域シーケンサ動作を強制停止します。	
Preconditions	コード/データ・フラッシュ領域シーケンサを起動するコマンド開始後（コマンド実行中、シーケンサ動作中）に使用してください。 R_RFD_CheckCFDFSeqEndStep1()により、R_RFD_ENUM_RET_STS_OK が返る前（シーケンサ動作終了前）に使用してください。	
Remarks	本関数実行後 R_RFD_CheckCFDFSeqEndStep1()を使用してください。	

動作概要：

・ブランク・チェック、内部ベリファイ、消去コマンドのコード/データ・フラッシュ領域シーケンサ動作中にFSSQ レジスタの FSSTP[bit6]に’1’を設定し、コード/データ・フラッシュ領域シーケンサ動作を強制停止します。

備考：

・本関数は、コマンドを**緊急時に**強制停止させたい場合のみ使用してください。
・ブランク・チェック、内部ベリファイ、消去コマンドのコード/データ・フラッシュ領域シーケンサ動作中にのみ使用してください。
・消去コマンド中に本関数を実行した場合は、対象領域をもう一度消去してください。
・ブランク・チェック、内部ベリファイ、消去以外のコマンドのコード/データ・フラッシュ領域シーケンサ動作中、及びエクストラ領域シーケンサ動作中は本関数を実行しないでください。使用した場合、その後の動作は不定になります。（書き込みコマンド中に本関数を実行した場合は、書き込みデータが不定となります。）
・本関数は、コマンド実行状態が不定のときは実行できません。
・本関数使用後、強制停止したコマンドによりエラーが発生する場合がありますが、コマンドが終了していない可能性があるため、エラー・フラグを参照しないでください。

3.3.1.12 R_RFD_ForceReset

Information

Syntax	R_RFD_FAR_FUNC void R_RFD_ForceReset(void);	
Reentrancy	Non-Reentrant	
Parameters (IN)	N/A	
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	N/A	
Description	CPU の内部リセットを発生させます。	
Preconditions	-	
Remarks	-	

動作概要：

・意図的に不正命令(0xFF)の命令コードを実行し、CPU の内部リセットを発生させます。

備考：

・CPU の内部リセットが発生するため、この関数以降の処理は実行されません。

・FFH の命令コードによる内部リセット（不正命令の実行による内部リセット）については、対象となる RL78 マイクロコントローラのユーザーズマニュアルを参照してください。

・本関数によるリセットは、オンチップ・デバッグ・エミュレータによるエミュレーションでは発生しません。

3.3.1.13 R_RFD_SetBootAreaImmediately

Information

Syntax	R_RFD_FAR_FUNC void R_RFD_SetBootAreaImmediately (e_rfd_boot_cluster_t i_e_boot_cluster);	
Reentrancy	Non-Reentrant	
Parameters (IN)	e_rfd_boot_cluster_t i_e_boot_cluster	ブート・クラスタ番号
		R_RFD_ENUM_BOOT_CLUSTER_0 : 0x01 [ブート・クラスタ 0] R_RFD_ENUM_BOOT_CLUSTER_1 : 0x00 [ブート・クラスタ 1]
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	N/A	
Description	引数で指定されたブート・クラスタを、ブート領域(00000H~03FFFH)に即時設定します。	
Preconditions	コード・フラッシュ・プログラミング・モード、または、データ・フラッシュ・プログラミング・モードで使用してください。 コード/データ・フラッシュ領域シーケンサ、エクストラ領域シーケンサのコマンドを実行していない状態で使用してください。	
Remarks	-	

動作概要：

・ユーザが引数(i_e_boot_cluster)で指定したブート・クラスタ番号の指定値を FSSET レジスタの TMBTSEL [bit6] に設定すると共に TMSPMD[bit7]に'1'を設定し、即時に対象のブート・クラスタをブート領域に設定します。

- 引数(i_e_boot_cluster)に"R_RFD_ENUM_BOOT_CLUSTER_0"を指定:
  FSSET に"R_RFD_VALUE_U08_FSSET_BOOT_CLUSTER_0(0x80u) | (g_u08_fset_cpu_frequency)"を設定します。
- 引数(i_e_boot_cluster)に"R_RFD_ENUM_BOOT_CLUSTER_1"を指定:
  FSSET に"R_RFD_VALUE_U08_FSSET_BOOT_CLUSTER_1(0xC0u) | (g_u08_fset_cpu_frequency)"を設定します。

備考：

・引数(i_e_boot_cluster)に範囲外の値を設定した場合、ブート・クラスタ 0 をブート領域に設定します。

・ブート領域に設定しなかったブート・クラスタは、ブート領域(0x00000~0x03FFF)の直後(0x04000~0x07FFF)に配置されます。

・CPU リセットを行うと、本関数の実行に依存せず、FLSEC レジスタのブート領域切替フラグ(BTFLG)[bit0]の値に対するクラスタがブート領域に設定されます。

・コード/データ・フラッシュ領域シーケンサ、エクストラ領域シーケンサのコマンド実行中に、本関数を実行した場合、その後の動作は不定となります。

3.3.1.14 R_RFD_GetSecurityAndBootFlags

Information

Syntax	R_RFD_FAR_FUNC void R_RFD_GetSecurityAndBootFlags (uint16_t __near * onp_u16_security_and_boot_flags);	
Reentrancy	Non-Reentrant	
Parameters (IN)	N/A	
Parameters (IN/OUT)	N/A	
Parameters (OUT)	uint16_t __near * onp_u16_security_and_boot_flags	セキュリティ・フラグ(各プロテクト・フラグ)とブート領域切替フラグの情報を格納する変数へのポインタ
Return Value	N/A	
Description	セキュリティ・フラグ(各プロテクト・フラグ)とブート領域切替フラグの情報を取得します。	
Preconditions	コード/データ・フラッシュ領域シーケンサ、エクストラ領域シーケンサのコマンドを実行していない状態で使用してください。	
Remarks	-	

動作概要：

・セキュリティ・フラグ(各プロテクト・フラグ)とブート領域切替フラグの情報を示す FLSEC レジスタの値(16bit)を読み出し、引数ポインタ(onp_u16_security_and_boot_flags)が示す変数に格納します。

備考：

・取得するセキュリティ・フラグとブート領域切替フラグ情報：FLSEC レジスタ bit15-0

[bit15-13] : -

[bit12] WRPR： 書き込み禁止フラグ

[bit11] : -

[bit10] SEPR： ブロック消去禁止フラグ

[bit9] BTPR： ブート領域書き換え禁止フラグ

[bit8] BTFLG： ブート領域切替フラグ

[bit7-0] : -

・本関数で取得できる(BTFLG)[bit8]に関して、(0)はブート・クラスタ 1、(1)はブート・クラスタ 0 を示します。

・コード/データ・フラッシュ領域シーケンサ、エクストラ領域シーケンサのコマンド実行中に、本関数を実行した場合、正しい値を取得できない可能性があります。

3.3.1.15 R_RFD_GetFSW

Information

Syntax	R_RFD_FAR_FUNC void R_RFD_GetFSW (uint16_t __near * onp_u16_start_block_number, uint16_t __near * onp_u16_end_block_number);	
Reentrancy	Non-Reentrant	
Parameters (IN)	N/A	
Parameters (IN/OUT)	N/A	
Parameters (OUT)	uint16_t __near * onp_u16_start_block_number	スタート・ブロック番号を格納する変数へのポインタ
	uint16_t __near * onp_u16_end_block_number	エンド・ブロック番号+1 格納する変数へのポインタ
Return Value	N/A	
Description	フラッシュ・シールド・ウインドウの範囲を取得します。	
Preconditions	コード/データ・フラッシュ領域シーケンサ、エクストラ領域シーケンサのコマンドを実行していない状態で使用してください。	
Remarks	-	

動作概要：

・フラッシュ・シールド・ウインドウのスタート・ブロック、エンド・ブロック+1 を示す FLFSWS(16bit), FLFSWE(16bit)レジスタの値を読み出し、各引数ポインタが指す変数に格納します。

- 各引数ポインタが指す変数の値(出力)：

*onp_u16_start_block_number： スタート・ブロック(FLFSWS[bit9-0]へ設定、[bit15-10]は 0 でマスク)

*onp_u16_end_block_number： エンド・ブロック+1 (FLFSWE[bit9-0] へ設定、[bit15-10]は 0 でマスク)

備考：

・デバイスの初期状態で、本関数を実行すると、onp_u16_start_block_number = 1023, onp_u16_end_block_number = 1023 を取得します。

・コード/データ・フラッシュ領域シーケンサ、エクストラ領域シーケンサのコマンド実行中に、本関数を実行した場合、正しい値を取得できない可能性があります。

3.3.1.16 r_rfd_wait_count

Information

Syntax	R_RFD_FAR_FUNC void r_rfd_wait_count(uint8_t i_u08_count);	
Reentrancy	Non-Reentrant	
Parameters (IN)	uint8_t i_u08_count	Wait する時間（1 カウントで 1μsec:1~255）
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	N/A	
Description	1 カウントで 1μsec として、入力されたパラメータ値の時間をソフトウェアループで Wait します。	
Preconditions	-	
Remarks	-	

動作概要：

・g_u08_cpu_frequency（CPU 動作周波数 - 1）の値に 1 を足して、CPU 動作周波数値に戻します。

・指定した Wait する時間（カウント[μs]）のソフトウェアループ回数を算出、ソフトウェアループを実行します。

[指定した Wait する時間（カウント[μs]）のソフトウェアループ回数]

= [（周波数値[MHz]） ×（カウント[μs]）] ／（ループの実行サイクル:8[サイクル]）+ 1

例）周波数値:32[MHz]、カウント:10[μs]の場合

Wait する時間（カウント[μs]）のソフトウェアループ回数 =（32[MHz] ×10[μs] /8[サイクル] ）+ 1

（切り捨て計算で Wait 時間未満にならないよう、1 回加算されています。）

= 41[回]

本関数実行時間 = 1/32[MHz] × 8[サイクル] ×41[回] = 10.25[μs]

備考：

・Wait 範囲は 1~255μs までで、ループ以外の処理のオーバヘッドは Wait 時間に含まれていません。

3.3.2 コード・フラッシュ制御 API 関数仕様

RFD RL78 Type02 のコード・フラッシュ操作関数を示します。

3.3.2.1 R_RFD_EraseCodeFlashReq

Information

Syntax	R_RFD_FAR_FUNC void R_RFD_EraseCodeFlashReq(uint16_t i_u16_block_number);	
Reentrancy	Non-Reentrant	
Parameters (IN)	uint16_t i_u16_block_number	消去対象ブロック番号[0~511] 例:RL78/F24 では、MAX 256KB[0~255] 例:RL78/F23 では、MAX 128KB[0~127]
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	N/A	
Description	コード/データ・フラッシュ領域シーケンサを起動し、コード・フラッシュの消去(1 ブロック)を開始します。	
Preconditions	コード・フラッシュ・プログラミング・モードで使用してください。 コード/データ・フラッシュ領域シーケンサ、エクストラ領域シーケンサのコマンドを実行していない状態で使用してください。	
Remarks	本関数実行後 R_RFD_CheckCFDFSeqEndStep1()を実行してください。	

動作概要：

・フラッシュの書き換え領域をコード/データ・フラッシュ領域に設定します。
FLARS レジスタ = R_RFD_VALUE_U08_FLARS_USER_AREA:0x00 (EXA [bit0] = 0)

・コード/データ・フラッシュ領域シーケンサを起動し、コード・フラッシュ・メモリの消去する 1 ブロック(1Kbyte)のアドレスを設定します。
- 引数(i_u16_block_number)の消去対象ブロック番号から、コード・フラッシュ・メモリのスタート・アドレスとエンド・アドレス(1 ブロック分：1Kbyte)を計算し、FLAPL/H と FLSEDL/H へ設定します。

・FSSQ レジスタに R_RFD_VALUE_U08_FSSQ_ERASE :0x84 を設定し、消去を開始します。
(SQST[bit7] = 1, SQMD[bit2-0] = 4[0b100], 他の bit は 0)

備考：

・引数(i_u16_block_number)は 16bit の上位 6bit を無効とした下位 10bit を使用します。デバイスに実装されているコード・フラッシュ・ブロック数の範囲内の値であることが前提条件です。範囲外の値を指定した場合、その後の動作は不定となります。

・コード・フラッシュ・プログラミング・モード以外で本関数を実行した場合、その後の動作は不定となります。

・コード/データ・フラッシュ領域シーケンサ、エクストラ領域シーケンサのコマンド実行中に本関数を実行した場合、その後の動作は不定となります。

3.3.2.2 R_RFD_WriteCodeFlashReq

Information

Syntax	R_RFD_FAR_FUNC void R_RFD_WriteCodeFlashReq (uint32_t i_u32_start_addr, uint8_t __near * inp_u08_write_data);	
Reentrancy	Non-Reentrant	
Parameters (IN)	uint32_t i_u32_start_addr	書き込み対象スタート・アドレス(4byte 境界) [コード・フラッシュ領域のアドレス]
	uint8_t __near * inp_u08_write_data	書き込みデータ変数へのポインタ [ポインタが指す書き込みデータは 4byte]
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	N/A	
Description	コード/データ・フラッシュ領域シーケンサを起動し、コード・フラッシュの書き込み(4byte)を開始します。	
Preconditions	コード・フラッシュ・プログラミング・モードで使用してください。 コード/データ・フラッシュ領域シーケンサ、エクストラ領域シーケンサのコマンドを実行していない状態で使用してください。	
Remarks	本関数実行後 R_RFD_CheckCFDFSeqEndStep1()を実行してください。	

動作概要：

・フラッシュの書き換え領域をコード/データ・フラッシュ領域に設定します。
FLARS レジスタ = R_RFD_VALUE_U08_FLARS_USER_AREA:0x00 (EXA [bit0] = 0)

・コード/データ・フラッシュ領域シーケンサを起動し、コード・フラッシュ・メモリの書き込みアドレスと書き込みデータ(4byte)を設定します。
- 引数(i_u32_start_addr)の書き込み対象のコード・フラッシュ・スタート・アドレスを FLAPL/H レジスタに設定します。
- 引数ポインタ(inp_u08_write_data)が指す変数(コード・フラッシュの書き込みデータ)の値(4byte)を FLWL/H レジスタに設定します。

・FSSQ レジスタに R_RFD_VALUE_U08_FSSQ_WRITE:0x81 を設定し、書き込みを開始します。
(SQST[bit7] = 1, SQMD[bit2-0] = 1[0b001], 他の bit は 0)

備考：

・引数(i_u32_start_addr)は 32bit の上位 8bit を 0x00 でマスクした下位 24bit を使用します。デバイスに実装されているコード・フラッシュ領域の範囲内の 4byte 境界のアドレスであることが前提条件です。範囲外の領域や 4byte 境界以外のアドレスを指定した場合、その後の動作は不定となります。

・引数(inp_u08_write_data)は、入力が 8bit データへのポインタです。このポインタを更新しながら継続して使用する場合、コード・フラッシュの書き込み単位 4byte ずつ更新する必要があるので、ご注意ください。

・コード・フラッシュ・プログラミング・モード以外で本関数を実行した場合、その後の動作は不定となります。

・コード/データ・フラッシュ領域シーケンサ、エクストラ領域シーケンサのコマンド実行中に本関数を実行した場合、その後の動作は不定となります。

3.3.2.3 R_RFD_BlankCheckCodeFlashReq

Information

Syntax	R_RFD_FAR_FUNC void R_RFD_BlankCheckCodeFlashReq (uint16_t i_u16_block_number);	
Reentrancy	Non-Reentrant	
Parameters (IN)	uint16_t i_u16_block_number	ブランク・チェック対象ブロック番号[0~511] 例:RL78/F24 では、MAX 256KB[0~255] 例:RL78/F23 では、MAX 128KB[0~127]
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	N/A	
Description	コード/データ・フラッシュ領域シーケンサを起動し、コード・フラッシュのブランク・チェック(1 ブロック)を開始します。	
Preconditions	コード・フラッシュ・プログラミング・モードで使用してください。 コード/データ・フラッシュ領域シーケンサ、エクストラ領域シーケンサのコマンドを実行していない状態で使用してください。	
Remarks	本関数実行後 R_RFD_CheckCFDFSeqEndStep1()を実行してください。	

動作概要：

・フラッシュの書き換え領域をコード/データ・フラッシュ領域に設定します。
FLARS レジスタ = R_RFD_VALUE_U08_FLARS_USER_AREA:0x00 (EXA [bit0] = 0)

・コード/データ・フラッシュ領域シーケンサを起動し、コード・フラッシュ・メモリのブランク・チェックする 1 ブロック(1024byte)のアドレスを設定します。
- 引数(i_u16_block_number)のブランク・チェック対象ブロック番号から、コード・フラッシュ・メモリのスタート・アドレスとエンド・アドレス(1 ブロック分：1024byte)を計算し、FLAPL/H と FLSEDL/H へ設定します。

・FSSQ レジスタに R_RFD_VALUE_U08_FSSQ_BLANKCHECK_CF:0x83 を設定し、ブランク・チェックを開始します。
（SQST[bit7] = 1, MDCH[bit3] = 0, SQMD[bit2-0] = 3[0b011], 他の bit は 0）

備考：

・引数(i_u16_block_number)は 16bit の上位 6bit を無効とした下位 10bit を使用します。デバイスに実装されているコード・フラッシュ・ブロック数の範囲内の値であることが前提条件です。範囲外の値を指定した場合、その後の動作は不定となります。

・コード・フラッシュ・プログラミング・モード以外で本関数を実行した場合、その後の動作は不定となります。

・コード/データ・フラッシュ領域シーケンサ、エクストラ領域シーケンサのコマンド実行中に本関数を実行した場合、その後の動作は不定となります。

3.3.2.4 R_RFD_IVerifyCodeFlashReq

Information

Syntax	R_RFD_FAR_FUNC void R_RFD_FAR_FUNC void R_RFD_IVerifyCodeFlashReq (uint16_t i_u16_block_number);	
Reentrancy	Non-Reentrant	
Parameters (IN)	uint16_t i_u16_block_number	内部ベリファイ対象ブロック番号[0~511] 例:RL78/F24 では、MAX 256KB[0~255] 例:RL78/F23 では、MAX 128KB[0~127]
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	N/A	
Description	コード/データ・フラッシュ領域シーケンサを起動し、コード・フラッシュの内部ベリファイ(1 ブロック)を開始します。	
Preconditions	コード・フラッシュ・プログラミング・モードで使用してください。 コード/データ・フラッシュ領域シーケンサ、エクストラ領域シーケンサのコマンドを実行していない状態で使用してください。	
Remarks	本関数実行後 R_RFD_CheckCFDFSeqEndStep1()を実行してください。	

動作概要：

・フラッシュの書き換え領域をコード/データ・フラッシュ領域に設定します。
FLARS レジスタ = R_RFD_VALUE_U08_FLARS_USER_AREA:0x00 (EXA [bit0] = 0)

・コード/データ・フラッシュ領域シーケンサを起動し、コード・フラッシュ・メモリの内部ベリファイをする 1 ブロック(1024byte)のアドレスを設定します。
- 引数(i_u16_block_number)の内部ベリファイ対象ブロック番号から、コード・フラッシュ・メモリのスタート・アドレスとエンド・アドレス(1 ブロック分：1024byte)を計算し、FLAPL/H と FLSEDL/H へ設定します。

・FSSQ レジスタに R_RFD_VALUE_U08_FSSQ_IVERIFY_CF:0x82 を設定し、内部ベリファイを開始します。
(SQST[bit7] = 1, MDCH[bit3] = 0, SQMD[bit2-0] = 2[0b010], 他の bit は 0)

備考：

・引数(i_u16_block_number)は 16bit の上位 6bit を無効とした下位 10bit を使用します。デバイスに実装されているコード・フラッシュ・ブロック数の範囲内の値であることが前提条件です。範囲外の値を指定した場合、その後の動作は不定となります。

・コード・フラッシュ・プログラミング・モード以外で本関数を実行した場合、その後の動作は不定となります。

・コード/データ・フラッシュ領域シーケンサ、エクストラ領域シーケンサのコマンド実行中に本関数を実行した場合、その後の動作は不定となります。

・書き込んだ直後に、対象ブロックの内部ベリファイを 1 回だけ実行することができます。内部ベリファイを複数回実行しないでください。

3.3.3 データ・フラッシュ API 制御関数仕様

RFD RL78 Type02 のデータ・フラッシュ制御関数を示します。

3.3.3.1 R_RFD_EraseDataFlashReq

Information

Syntax	R_RFD_FAR_FUNC void R_RFD_EraseDataFlashReq(uint8_t i_u08_block_number);	
Reentrancy	Non-Reentrant	
Parameters (IN)	uint8_t i_u08_block_number	消去対象ブロック番号[0~63] 例:RL78/F24 では、MAX 16KB[0~15] 例:RL78/F23 では、MAX 8KB[0~7]
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	N/A	
Description	コード/データ・フラッシュ領域シーケンサを起動し、データ・フラッシュの消去(1 ブロック)を開始します。	
Preconditions	データ・フラッシュ・アクセス許可状態(DFLEN = 1)で使用してください。 データ・フラッシュ・プログラミング・モードで使用してください。 コード/データ・フラッシュ領域シーケンサ、エクストラ領域シーケンサのコマンドを実行していない状態で使用してください。	
Remarks	本関数実行後 R_RFD_CheckCFDFSeqEndStep1()を実行してください。	

動作概要：

・フラッシュの書き換え領域をコード/データ・フラッシュ領域に設定します。

FLARS レジスタ = R_RFD_VALUE_U08_FLARS_USER_AREA:0x00 (EXA [bit0] = 0)

・コード/データ・フラッシュ領域シーケンサを起動し、データ・フラッシュ・メモリの消去する 1 ブロック(1024byte)のアドレスを設定します。

- 引数(i_u08_block_number)の消去対象ブロック番号から、データ・フラッシュ・メモリのスタート・アドレスとエンド・アドレス(1 ブロック分：1024byte)を計算し、FLAPL/H と FLSEDL/H へ設定します。

・FSSQ レジスタに R_RFD_VALUE_U08_FSSQ_ERASE :0x84 を設定し、消去を開始します。

(SQST[bit7] = 1, SQMD[bit2-0] = 4[0b100], 他の bit は 0)

備考：

・引数(i_u08_block_number)は 8bit の上位 2bit を無効とした下位 6bit を使用します。デバイスに実装されているデータ・フラッシュ・ブロック数の範囲内の値であることが前提条件です。範囲外の値を指定した場合、その後の動作は不定となります。

・データ・フラッシュ・アクセス禁止状態で本関数を実行した場合、その後の動作は不定となります。

・データ・フラッシュ・プログラミング・モード以外で本関数を実行した場合、その後の動作は不定となります。

・コード/データ・フラッシュ領域シーケンサ、エクストラ領域シーケンサのコマンド実行中に本関数を実行した場合、その後の動作は不定となります。

3.3.3.2 R_RFD_WriteDataFlashReq

Information

Syntax	R_RFD_FAR_FUNC void R_RFD_WriteDataFlashReq (uint32_t i_u32_start_addr, uint8_t __near * inp_u08_write_data);	
Reentrancy	Non-Reentrant	
Parameters (IN)	uint32_t i_u32_start_addr	書き込み対象スタート・アドレス [データ・フラッシュ領域のアドレス]
	uint8_t __near * inp_u08_write_data	書き込みデータ変数へのポインタ [ポインタが指す書き込みデータは 1byte]
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	N/A	
Description	コード/データ・フラッシュ領域シーケンサを起動し、データ・フラッシュの書き込み(1byte)を開始します。	
Preconditions	データ・フラッシュ・アクセス許可状態(DFLEN = 1)で使用してください。 データ・フラッシュ・プログラミング・モードで使用してください。 コード/データ・フラッシュ領域シーケンサ、エクストラ領域シーケンサのコマンドを実行していない状態で使用してください。	
Remarks	本関数実行後 R_RFD_CheckCFDFSeqEndStep1()を実行してください。	

動作概要：

・フラッシュの書き換え領域をコード/データ・フラッシュ領域に設定します。
FLARS レジスタ = R_RFD_VALUE_U08_FLARS_USER_AREA:0x00 (EXA [bit0] = 0)

・コード/データ・フラッシュ領域シーケンサを起動し、データ・フラッシュ・メモリの書き込みアドレスと書き込みデータ(1byte)を設定します。
- 引数(i_u32_start_addr)の書き込み対象のデータ・フラッシュ・スタート・アドレスを FLAPL/H レジスタに設定します。
- 引数ポインタ(inp_u08_write_data)が指す変数（データ・フラッシュの書き込みデータ）の値(1byte)を FLWL レジスタの下位 8bit に設定します。

・FSSQ レジスタに R_RFD_VALUE_U08_FSSQ_WRITE:0x81 を設定し、書き込みを開始します。
（SQST[bit7] = 1, SQMD[bit2-0] = 1[0b001], 他の bit は 0）

備考：

・引数(i_u32_start_addr)は 32bit の上位 8bit を 0x00 でマスクした下位 24bit を使用します。デバイスに実装されているデータ・フラッシュ領域の範囲内であることが前提条件です。範囲外の領域を指定した場合、その後の動作は不定となります。

・データ・フラッシュ・アクセス禁止状態で本関数を実行した場合、その後の動作は不定となります。

・データ・フラッシュ・プログラミング・モード以外で本関数を実行した場合、その後の動作は不定となります。

・コード/データ・フラッシュ領域シーケンサ、エクストラ領域シーケンサのコマンド実行中に本関数を実行した場合、その後の動作は不定となります。

3.3.3.3 R_RFD_BlankCheckDataFlashReq

Information

Syntax	R_RFD_FAR_FUNC void R_RFD_BlankCheckDataFlashReq (uint32_t i_u32_start_addr, uint16_t i_u16_blankcheck_length);	
Reentrancy	Non-Reentrant	
Parameters (IN)	uint32_t i_u32_start_addr	ブランク・チェック対象スタート・アドレス [データ・フラッシュ領域のアドレス]
	uint16_t i_u16_blankcheck_length	ブランク・チェック対象データ数
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	N/A	
Description	コード/データ・フラッシュ領域シーケンサを起動し、データ・フラッシュのブランク・チェック(指定バイト数)を開始します。	
Preconditions	データ・フラッシュ・アクセス許可状態(DFLEN = 1)で使用してください。 データ・フラッシュ・プログラミング・モードで使用してください。 コード/データ・フラッシュ領域シーケンサ、エクストラ領域シーケンサのコマンドを実行していない状態で使用してください。	
Remarks	本関数実行後 R_RFD_CheckCFDFSeqEndStep1()を実行してください。	

動作概要：

・フラッシュの書き換え領域をコード/データ・フラッシュ領域に設定します。
FLARS レジスタ = R_RFD_VALUE_U08_FLARS_USER_AREA:0x00 (EXA [bit0] = 0)

・コード/データ・フラッシュ領域シーケンサを起動し、データ・フラッシュ・メモリのブランク・チェックするスタート・アドレスとエンド・アドレスを設定します。
- 引数(u32_start_addr)のブランク・チェック対象のデータ・フラッシュ・スタート・アドレスを FLAPL/H レジスタに設定します。
- エンド・アドレス(開始アドレス＋指定バイト数)を計算し、FLSEDL/H に設定します。

・FSSQ レジスタに R_RFD_VALUE_U08_FSSQ_BLANKCHECK_DF:0x8B を設定し、ブランク・チェックを開始します。
（SQST[bit7] = 1, MDCH[bit3] = 1, SQMD[bit2-0] = 3[0b011], 他の bit は 0）

備考：

・ブロックをまたがって指定することはできません。1 ブロックの範囲内で指定してください。

・引数(i_u32_start_addr)は 32bit の上位 8bit を 0x00 でマスクした下位 24bit を使用します。デバイスに実装されているデータ・フラッシュ領域の範囲内であることが前提条件です。範囲外の領域を指定した場合、その後の動作は不定となります。

・データ・フラッシュ・アクセス禁止状態で本関数を実行した場合、その後の動作は不定となります。

・データ・フラッシュ・プログラミング・モード以外で本関数を実行した場合、その後の動作は不定となります。

・コード/データ・フラッシュ領域シーケンサ、エクストラ領域シーケンサのコマンド実行中に本関数を実行した場合、その後の動作は不定となります。

3.3.3.4 R_RFD_IVerifyDataFlashReq

Information

Syntax	R_RFD_FAR_FUNC void R_RFD_IVerifyDataFlashReq (uint32_t i_u32_start_addr, uint16_t i_u16_iverify_length);	
Reentrancy	Non-Reentrant	
Parameters (IN)	uint32_t i_u32_start_addr	内部ベリファイ対象スタート・アドレス [データ・フラッシュ領域のアドレス] 例:RL78/F24 では、0xF1000 – 0xF4FFF [16KB]
	uint16_t i_u16_iverify_length	書き込みバイト数
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	N/A	
Description	コード/データ・フラッシュ領域シーケンサを起動し、データ・フラッシュの内部ベリファイ(書き込みバイト数)を開始します。	
Preconditions	データ・フラッシュ・アクセス許可状態(DFLEN = 1)で使用してください。 データ・フラッシュ・プログラミング・モードで使用してください。 コード/データ・フラッシュ領域シーケンサ、エクストラ領域シーケンサのコマンドを実行していない状態で使用してください。	
Remarks	本関数実行後 R_RFD_CheckCFDFSeqEndStep1()を実行してください。	

動作概要：

・フラッシュの書き換え領域をコード/データ・フラッシュ領域に設定します。
FLARS レジスタ = R_RFD_VALUE_U08_FLARS_USER_AREA:0x00 (EXA [bit0] = 0)

・コード/データ・フラッシュ領域シーケンサを起動し、データ・フラッシュ・メモリの内部ベリファイを開始するアドレスと対象のバイト数を設定します。
- 引数(i_u32_start_addr)の内部ベリファイ対象のデータ・フラッシュ・スタート・アドレスを FLAPL/H レジスタに設定します。
- エンド・アドレス(開始アドレス＋書き込みバイト数)を計算し、FLSEDL/H に設定します。

・FSSQ レジスタに R_RFD_VALUE_U08_FSSQ_IVERIFY_DF:0x8A を設定し、内部ベリファイを開始します。
(SQST[bit7] = 1, MDCH[bit3] = 1, SQMD[bit2-0] = 2[0b010], 他の bit は 0)

備考：

・ブロックをまたがって指定することはできません。1 ブロックの範囲内で指定してください。

・引数(i_u32_start_addr)は 32bit の上位 8bit を 0x00 でマスクした下位 24bit を使用します。デバイスに実装されているデータ・フラッシュ領域の範囲内であることが前提条件です。範囲外の領域を指定した場合、その後の動作は不定となります。

・データ・フラッシュ・アクセス禁止状態で本関数を実行した場合、その後の動作は不定となります。

・データ・フラッシュ・プログラミング・モード以外で本関数を実行した場合、その後の動作は不定となります。

・コード/データ・フラッシュ領域シーケンサ、エクストラ領域シーケンサのコマンド実行中に本関数を実行した場合、その後の動作は不定となります。

・書き込んだ直後に、対象アドレスの内部ベリファイを 1 回だけ実行することができます。複数回実行しないでください。

3.3.4 エクストラ領域制御 API 関数仕様

RFD RL78 Type02 のエクストラ領域制御関数を示します。

3.3.4.1 R_RFD_SetExtraEraseProtectReq

Information

Syntax	R_RFD_FAR_FUNC void R_RFD_SetExtraEraseProtectReq(void);	
Reentrancy	Non-Reentrant	
Parameters (IN)	N/A	
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	N/A	
Description	エクストラ領域シーケンサを起動し、ブロック消去禁止フラグの書き込みを開始します。	
Preconditions	コード・フラッシュ・プログラミング・モードで使用してください。 コード/データ・フラッシュ領域シーケンサ、エクストラ領域シーケンサのコマンドを実行していない状態で使用してください。	
Remarks	本関数実行後 R_RFD_CheckExtraSeqEndStep1()を実行してください。	

動作概要：

・フラッシュの書き換え領域をエクストラ領域に設定します。

FLARS レジスタ = R_RFD_VALUE_U08_FLARS_EXTRA_AREA:0x01 (EXA [bit0] = 1)

・エクストラ領域シーケンサを起動し、ブロック消去禁止フラグの書き込みを開始します。

- FLSEC レジスタを読み出し、現在設定されている BTFLG[bit8]の値を反映、SEPR[bit10]を'0'（ブロック消去禁止）にして FLWL レジスタに設定するとともに、0xFFFF を FLWH レジスタに設定します。

・FSSE レジスタに R_RFD_VALUE_U08_FSSE_SECURITY_FLAG :0x81 を設定し、書き込みを開始します。

（ESQST[bit7] = 1, ESQMD[bit2-0] = 1[0b001], 他の bit は 0）

備考：

・コード・フラッシュ・プログラミング・モード以外で本関数を実行した場合、その後の動作は不定となります。

・コード/データ・フラッシュ領域シーケンサ、エクストラ領域シーケンサのコマンド実行中に本関数を実行した場合、その後の動作は不定となります。

3.3.4.2 R_RFD_SetExtraWriteProtectReq

Information

Syntax	R_RFD_FAR_FUNC void R_RFD_SetExtraWriteProtectReq(void);	
Reentrancy	Non-Reentrant	
Parameters (IN)	N/A	
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	N/A	
Description	エクストラ領域シーケンサを起動し、書き込み禁止フラグの書き込みを開始します。	
Preconditions	コード・フラッシュ・プログラミング・モードで使用してください。 コード/データ・フラッシュ領域シーケンサ、エクストラ領域シーケンサのコマンドを実行していない状態で使用してください。	
Remarks	本関数実行後 R_RFD_CheckExtraSeqEndStep1()を実行してください。	

動作概要：

・フラッシュの書き換え領域をエクストラ領域に設定します。

FLARS レジスタ = R_RFD_VALUE_U08_FLARS_EXTRA_AREA:0x01 (EXA [bit0] = 1)

・エクストラ領域シーケンサを起動し、書き込み禁止フラグの書き込みを開始します。

- FLSEC レジスタを読み出し、現在設定されている BTFLG[bit8]の値を反映、WRPR[bit12]を'0'（書き込み禁止）にして FLWL レジスタに設定するとともに、0xFFFF を FLWH レジスタに設定します。

・FSSE レジスタに R_RFD_VALUE_U08_FSSE_SECURITY_FLAG :0x81 を設定し、書き込みを開始します。

(ESQST[bit7] = 1, ESQMD[bit2-0] = 1[0b001], 他の bit は 0)

備考：

・コード・フラッシュ・プログラミング・モード以外で本関数を実行した場合、その後の動作は不定となります。

・コード/データ・フラッシュ領域シーケンサ、エクストラ領域シーケンサのコマンド実行中に本関数を実行した場合、その後の動作は不定となります。

3.3.4.3 R_RFD_SetExtraBootAreaProtectReq

Information

Syntax	R_RFD_FAR_FUNC void R_RFD_SetExtraBootAreaProtectReq(void);	
Reentrancy	Non-Reentrant	
Parameters (IN)	N/A	
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	N/A	
Description	エクストラ領域シーケンサを起動し、ブート領域書き換え禁止フラグの書き込みを開始します。	
Preconditions	コード・フラッシュ・プログラミング・モードで使用してください。 コード/データ・フラッシュ領域シーケンサ、エクストラ領域シーケンサのコマンドを実行していない状態で使用してください。	
Remarks	本関数実行後 R_RFD_CheckExtraSeqEndStep1()を実行してください。	

動作概要：

・フラッシュの書き換え領域をエクストラ領域に設定します。

FLARS レジスタ = R_RFD_VALUE_U08_FLARS_EXTRA_AREA:0x01 (EXA [bit0] = 1)

・エクストラ領域シーケンサを起動し、ブート領域書き換え禁止フラグの書き込みを開始します。

- FLSEC レジスタを読み出し、現在設定されている BTFLG[bit8]の値を反映、BTPR[bit9]を'0'(書き込み禁止)にして FLWL レジスタに設定するとともに、0xFFFF を FLWH レジスタに設定します。

・FSSE レジスタに R_RFD_VALUE_U08_FSSE_SECURITY_FLAG :0x81 を設定し、書き込みを開始します。

（ESQST[bit7] = 1, ESQMD[bit2-0] = 1[0b001], 他の bit は 0）

備考：

・コード・フラッシュ・プログラミング・モード以外で本関数を実行した場合、その後の動作は不定となります。

・コード/データ・フラッシュ領域シーケンサ、エクストラ領域シーケンサのコマンド実行中に本関数を実行した場合、その後の動作は不定となります。

3.3.4.4 R_RFD_SetExtraBootAreaReq

Information

Syntax	R_RFD_FAR_FUNC void R_RFD_SetExtraBootAreaReq (e_rfd_boot_cluster_t i_e_boot_cluster);	
Reentrancy	Non-Reentrant	
Parameters (IN)	e_rfd_boot_cluster_t i_e_boot_cluster	ブート・クラスタ番号 R_RFD_ENUM_BOOT_CLUSTER_0 : 0x01 [ブート・クラスタ 0] R_RFD_ENUM_BOOT_CLUSTER_1 : 0x00 [ブート・クラスタ 1]
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	N/A	
Description	エクストラ領域シーケンサを起動し、ブート領域切替フラグの書き込みを開始します。	
Preconditions	コード・フラッシュ・プログラミング・モードで使用してください。 コード/データ・フラッシュ領域シーケンサ、エクストラ領域シーケンサのコマンドを実行していない状態で使用してください。	
Remarks	本関数実行後 R_RFD_CheckExtraSeqEndStep1()を実行してください。	

動作概要：

・本関数実行で BTFLG が書かれた直後にブート・スワップせず、リセット実行後にブート・スワップするように設定します。
- FSSET レジスタと FLSEC レジスタを読み出します。
- FSSET レジスタの TMSPMD[bit7]が'0'の時のみ、TMSPMD[bit7]に'1'を設定すると同時に現在設定されている FLSEC レジスタの BTFLG[bit8]で指定されているブート・クラスタの状態を、FSSET レジスタの TMBTSEL[bit6]に反映します。

TMSPMD = 0 / 1： ブート・スワップはエクストラ領域の情報(BTFLG)に従う / ブート・スワップは TMBTSEL に従う

BTFLG = 0 / 1： ブート領域はブート・クラスタ 1 / ブート領域はブート・クラスタ 0

TMBTSEL = 0 / 1： ブート領域はブート・クラスタ 0 / ブート領域はブート・クラスタ 1

・フラッシュの書き換え領域をエクストラ領域に設定します。
FLARS レジスタ = R_RFD_VALUE_U08_FLARS_EXTRA_AREA:0x01 (EXA [bit0] = 1)

・エクストラ領域シーケンサを起動し、ブート領域切替フラグの書き込みを開始します。
引数(i_e_boot_cluster)により指定されるブート・クラスタを FLSEC レジスタの BTFLG[bit8]に該当するビットに設定した値を FLWL レジスタに設定するとともに、R_RFD_VALUE_U08_MASK1_16BIT (0xFFFF)を FLWH レジスタに設定します。
- R_RFD_ENUM_BOOT_CLUSTER_1 が指定された場合:
  FLWL レジスタに R_RFD_VALUE_U16_MASK0_BOOT_FLAG (0xFEFF)を設定します。
- R_RFD_ENUM_BOOT_CLUSTER_0 が指定された場合:
  FLWL レジスタに R_RFD_VALUE_U08_MASK1_16BIT (0xFFFF)を設定します。

・FSSE レジスタに R_RFD_VALUE_U08_FSSE_SECURITY_FLAG :0x81 を設定し、書き込みを開始します。
（ESQST[bit7] = 1, ESQMD[bit2-0] = 1[0b001], 他の bit は 0）

備考：

・引数(i_e_boot_cluster)は、正しい値（列挙型：e_rfd_boot_cluster_t）であることが前提条件です。引数(i_e_boot_cluster)に R_RFD_ENUM_BOOT_CLUSTER_0、R_RFD_ENUM_BOOT_CLUSTER_1 の値以外を指定した場合、R_RFD_ENUM_BOOT_CLUSTER_0 を設定します。
ブート領域に設定したブート・クラスタ ：00000H~03FFFH（ブート領域）に配置されます。
ブート領域に設定しなかったブート・クラスタ ：04000H~07FFFH（ブート領域の直後）に配置されます。

・コード・フラッシュ・プログラミング・モード以外で本関数を実行した場合、その後の動作は不定となります。

・コード/データ・フラッシュ領域シーケンサ、エクストラ領域シーケンサのコマンド実行中に本関数を実行した場合、その後の動作は不定となります。

3.3.4.5 R_RFD_SetExtraFSWReq

Information

Syntax	R_RFD_FAR_FUNC void R_RFD_SetExtraFSWReq (uint16_t i_u16_start_block_number, uint16_t i_u16_end_block_number);	
Reentrancy	Non-Reentrant	
Parameters (IN)	uint16_t i_u16_start_block_number	スタート・ブロック番号 例:RL78/F24 では、Max. 256Kbyte[0~255] 例:RL78/F23 では、Max. 128Kbyte[0~127]
	uint16_t i_u16_end_block_number	エンド・ブロック番号+1 例:RL78/F24 では、Max. 256Kbyte[1~256] 例:RL78/F23 では、Max. 128Kbyte[1~128]
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	N/A	
Description	エクストラ領域シーケンサを起動し、引数で指定されたフラッシュ・シールド・ウインドウの範囲の書き込みを開始します。	
Preconditions	コード・フラッシュ・プログラミング・モードで使用してください。 コード/データ・フラッシュ領域シーケンサ、エクストラ領域シーケンサのコマンドを実行していない状態で使用してください。	
Remarks	本関数実行後 R_RFD_CheckExtraSeqEndStep1()を実行してください。	

動作概要：

・フラッシュの書き換え領域をエクストラ領域に設定します。
FLARS レジスタ = R_RFD_VALUE_U08_FLARS_EXTRA_AREA:0x01 (EXA [bit0] = 1)

・エクストラ領域シーケンサを起動し、フラッシュ・シールド・ウインドウのスタート・ブロック番号、エンド・ブロック番号+1 の書き込みを開始します。
- 引数(i_u16_start_block_number)の FSWS（フラッシュ・シールド・ウインドウ・スタート・ブロック・アドレス）レジスタに該当するブロック番号を FLWL レジスタへ設定します。この時、ブロックアドレス以外のビット[bit15-10]は'0'に設定されています。
- 引数(i_u16_end_block_number)の FSWE（フラッシュ・シールド・ウインドウ・エンド・ブロック・アドレス）レジスタに該当するブロック番号を FLWH レジスタへ設定します。この時、ブロックアドレス以外のビット[bit15-10]は'0'に設定されています。

・FSSE レジスタに R_RFD_VALUE_U08_FSSE_FSW : 0x82 を設定し、書き込みを開始します。
（ESQST[bit7] = 1, ESQMD[bit2-0] = 2[0b010], 他の bit は 0）

備考：

・設定されるブロック番号は、引数の 16bit のうち、[bit15-10]を無効とした[bit9-0]が設定されます。
（最大 1023）

・引数は(i_u16_start_block_number) ＜ (i_u16_end_block_number) となるように指定してください。

・引数 i_u16_end_block_number には、設定したいウインドウ範囲のエンド・ブロック番号+1 を指定してください。
（例）

- ブロック 12 からブロック 15 の 4 ブロックの範囲外をシールドする場合:
  i_u16_start_block_number = 12, i_u16_end_block_number = 16

・コード・フラッシュ・プログラミング・モード以外で本関数を実行した場合、その後の動作は不定となります。

・コード/データ・フラッシュ領域シーケンサ、エクストラ領域シーケンサのコマンド実行中に本関数を実行した場合、その後の動作は不定となります。

3.3.5 フック関数仕様

RFD RL78 Type02 のフック関数を示します。

3.3.5.1 R_RFD_HOOK_EnterCriticalSection

Information

Syntax	R_RFD_FAR_FUNC void R_RFD_HOOK_EnterCriticalSection (void);	
Reentrancy	Non-Reentrant	
Parameters (IN)	N/A	
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	N/A	
Description	割り込み禁止命令を実行します。	
Preconditions	割り込み禁止状態で実行する処理の前に、割り込みを禁止する場合に実行します。	
Remarks	-	

動作概要：

・割り込みの禁止/許可状態を取得し、PSW の割り込み許可フラグ(IE)の保存データ(sg_u08_psw_ie_state)へ退避します。

・割り込み禁止マクロ命令[R_RFD_DISABLE_INTERRUPT]を実行します。

備考：

・割り込み禁止状態で実行する必要がある処理（クリティカル・セクション）の前に実行し、クリティカル・セクション終了後、R_RFD_HOOK_ExitCriticalSection 関数を実行してください。

3.3.5.2 R_RFD_HOOK_ExitCriticalSection

Information

Syntax	R_RFD_FAR_FUNC void R_RFD_HOOK_ExitCriticalSection_ (void);	
Reentrancy	Non-Reentrant	
Parameters (IN)	N/A	
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	N/A	
Description	割り込み許可命令を実行します。	
Preconditions	割り込み禁止状態で実行する処理終了後、割り込みを許可する場合に実行します。	
Remarks	-	

動作概要：

・PSW の割り込み許可フラグ(IE)の保存データ(sg_u08_psw_ie_state)の状態により、割り込み許可マクロ命令を実行します。

sg_u08_psw_ie_state の値:

- 0x00[bit7 = 0：割り込み禁止]の場合 ：何も実行しません。
- 0x80[bit7 = 1：割り込み許可]の場合 ：割り込み許可マクロ命令[R_RFD_ENABLE_INTERRUPT]を実行して、割り込み許可状態(EI)へ復帰します。

備考：

・R_RFD_HOOK_EnterCriticalSection 関数実行後、割り込み禁止状態で実行する必要がある処理（クリティカル・セクション）の終了後に実行します。

4 フラッシュ・メモリ・シーケンサ操作

4.1 フラッシュ・メモリ制御モードの設定

フラッシュ・メモリ・シーケンサの特定シーケンスを実行することで、フラッシュ・メモリ制御モードをコード・フラッシュ、およびデータ・フラッシュを書き換え可能な状態へ設定することが可能です。

- コード・フラッシュ（および、エクストラ領域）書き換え可能状態 :
  コード・フラッシュ・プログラミング・モード(Code flash programming mode)
- データ・フラッシュ 書き換え可能状態 :
  データ・フラッシュ・プログラミング・モード(Data flash programming mode)
- フラッシュ・メモリ(および エクストラ領域) 書き換え不可状態 :
  非書き換えモード(Non-programmable mode)

本操作の対象関数：R_RFD_SetFlashMemoryMode

・コード・フラッシュ・プログラミング・モード、およびデータ・フラッシュ・プログラミング・モードへは、非書き換えモードから移行してください。

・コード・フラッシュ・プログラミング・モードからデータ・フラッシュ・プログラミング・モードへ直接モードを移行しないでください。また、データ・フラッシュ・プログラミング・モードからコード・フラッシュ・プログラミング・モードへ直接モードを移行しないでください。

・非書き換えモードへは、移行しているフラッシュ・メモリ制御モードに対応した移行手順で移行してください。

注）データ・フラッシュ領域を操作する場合の前提条件は、既にデータ・フラッシュ・コントロール・レジスタ（DFLCTL レジスタ）の DFLEN ビット[bit0] = 1（データ・フラッシュのアクセス許可）に設定されていることです。

4.1.1 特定シーケンス実行手順

以下の特定シーケンスで書き込み動作を行った場合のみ、フラッシュ・プログラミング・モード・コントロール・レジスタ（FLPMC レジスタ）への書き込みが有効になり、各モードへの移行が可能になります。

手順	**特定シーケンス**（プログラム処理）
ステップ 1:	PFCMD レジスタへ特定値(=0xA5)を書き込む。
ステップ 2:	FLPMC レジスタへ設定したい値を書き込む。
ステップ 3:	FLPMC レジスタへ設定したい値の反転値を書き込む。
ステップ 4:	FLPMC レジスタへ設定したい値を書き込む。

・特定シーケンスは FLRST レジスタの FLRST[bit0] = 0、かつフラッシュ・メモリ・シーケンサが停止中の場合に実行可能です。

・特定シーケンスでは、ステップ 1、2、3、4 の間で他のメモリやレジスタへの書き込み動作を行った場合、特定レジスタ（FLPMC レジスタ）への書き込みは行われず、プロテクション・エラーが発生し、フラッシュ・ステータス・レジスタ（PFS レジスタ）のステータス・フラグ(FPRERR[bit0])が'1'にセットされます。PFS レジスタの FPRERR[bit0]は、リセット、または次の特定シーケンス開始時にクリアされます。

PFCMD レジスタ（リセット時:不定）：

7	6	5	4	3	2	1	0
REG7	REG6	REG5	REG4	REG3	REG2	REG1	REG0
W	W	W	W	W	W	W	W

- フラッシュ・プロテクト・コマンド・レジスタ（PFCMD レジスタ）は、書き込み専用のレジスタで、読み出し値は不定となります。

FLPMC レジスタ（リセット時:0x08）：

7	6	5	4	3	2	1	0
FLPMC[7 : 0]							
R/W							

コード/データ・フラッシュ・モード	FLPMC レジスタ設定
非書き換えモード （リード・モード）	リセット後の状態です。 コード・フラッシュ・プログラミング・モード、またはデータ・フラッシュ・プログラミング・モードから特定シーケンスで移行します。 「4.1.4 非書き換えモード移行手順」を参照してください。
コード・フラッシュ・プログラミング・モード	非書き換えモードから移行します。 「4.1.2 非書き換えモードからコード・フラッシュ・プログラミング・モードに移行する手順」を参照してください。
データ・フラッシュ・プログラミング・モード	非書き換えモードから移行します。 「4.1.3 非書き換えモードからデータ・フラッシュ・プログラミング・モードに移行する手順」を参照してください。

PFS レジスタ（リセット時:0x00）：

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	FPRERR
R	R	R	R	R	R	R	R

4.1.2 非書き換えモードからコード・フラッシュ・プログラミング・モードに移行する手順

ステップ 1:	PFCMD レジスタ = 0xA5
ステップ 2:	FLPMC レジスタ = 0x12
ステップ 3:	FLPMC レジスタ = 0xED
ステップ 4:	FLPMC レジスタ = 0x12
ステップ 5:	3us Wait
ステップ 6:	PFCMD レジスタ = 0xA5
ステップ 7:	FLPMC レジスタ = 0x92
ステップ 8:	FLPMC レジスタ = 0x6D
ステップ 9:	FLPMC レジスタ = 0x92
ステップ 10:	PFCMD レジスタ = 0xA5
ステップ 11:	FLPMC レジスタ = 0x82
ステップ 12:	FLPMC レジスタ = 0x7D
ステップ 13:	FLPMC レジスタ = 0x82
ステップ 14:	10μs Wait

・ステップ 2, 4
FLPMC レジスタ設定値(0x12)
・ステップ 3
FLPMC レジスタ反転値(0xED)

・ステップ 7, 9
FLPMC レジスタ設定値(0x92)
・ステップ 8
FLPMC レジスタ反転値(0x6D)

・ステップ 11, 13
FLPMC レジスタ設定値(0x82)
・ステップ 12
FLPMC レジスタ反転値(0x7D)

4.1.3 非書き換えモードからデータ・フラッシュ・プログラミング・モードに移行する手順

ステップ 1:	PFCMD レジスタ = 0xA5
ステップ 2:	FLPMC レジスタ = 0x10
ステップ 3:	FLPMC レジスタ = 0xEF
ステップ 4:	FLPMC レジスタ = 0x10
ステップ 5:	10μs Wait

・ステップ 2, 4
FLPMC レジスタ設定値(0x10)
・ステップ 3
FLPMC レジスタ反転値(0xEF)

4.1.4 非書き換えモード移行手順

コード・フラッシュ・プログラミング・モード、またはデータ・フラッシュ・プログラミング・モードから非書き換えモード移行手順実施後、10μs の Wait 時間を経過してからプログラミング・モード対象となっていたフラッシュ・メモリを読み出すことができるようになります。

(1) データ・フラッシュ・プログラミング・モードから非書き換えモードに移行する場合

ステップ 1:	PFCMD レジスタ = 0xA5
ステップ 2:	FLPMC レジスタ = 0x08
ステップ 3:	FLPMC レジスタ = 0xF7
ステップ 4:	FLPMC レジスタ = 0x08
ステップ 5:	10μs Wait 後、データ・フラッシュ・メモリの読み出しが可能となります。

・ステップ 2, 4
FLPMC レジスタ設定値(0x08)
・ステップ 3
FLPMC レジスタ反転値(0xF7)

(2) コード・フラッシュ・プログラミング・モードから非書き換えモードに移行する場合

ステップ 1:	PFCMD レジスタ = 0xA5	・ステップ 2, 4 FLPMC レジスタ設定値(0x92) ・ステップ 3 FLPMC レジスタ反転値(0x6D)
ステップ 2:	FLPMC レジスタ = 0x92	
ステップ 3:	FLPMC レジスタ = 0x6D	
ステップ 4:	FLPMC レジスタ = 0x92	
ステップ 5:	3µs Wait	
ステップ 6:	PFCMD レジスタ = 0xA5	・ステップ 7, 9 FLPMC レジスタ設定値(0x12) ・ステップ 8 FLPMC レジスタ反転値(0xED)
ステップ 7:	FLPMC レジスタ = 0x12	
ステップ 8:	FLPMC レジスタ = 0xED	
ステップ 9:	FLPMC レジスタ = 0x12	
ステップ 10:	PFCMD レジスタ = 0xA5	・ステップ 11, 13 FLPMC レジスタ設定値(0x08) ・ステップ 12 FLPMC レジスタ反転値(0xF7)
ステップ 11:	FLPMC レジスタ = 0x08	
ステップ 12:	FLPMC レジスタ = 0xF7	
ステップ 13:	FLPMC レジスタ = 0x08	

ステップ 14:　10µs Wait 後、コード・フラッシュ・メモリの読み出しが可能となります。

4.2 フラッシュ・メモリ・シーケンサ用レジスタのクリア

フラッシュ初期化レジスタ(FLRST レジスタ)の FLRST ビットをセットすることで、対象レジスタをクリアします。

クリア対象レジスタ：FLAPH, FLAPL, FLSEDH, FLSEDL, FLWH, FLWL, FLARS, FSSQ, FSSE

本操作の対象関数：R_RFD_ClearSeqRegister

≪操作方法≫

・FLRST ビットを'1'にします。(FLRST レジスタに 0x01 を書き込みます。)

・1 サイクル以上待ちます。(NOP 命令等)

・FLRST ビットを'0'にします。(FLRST レジスタに 0x00 を書き込みます。)

注)FSSQ レジスタの SQST ビットが'0'、かつ FSSE レジスタの ESQST ビットが'0'の場合(フラッシュ・メモリ・シーケンサが停止中)に限り FLRST ビットの操作が可能です。それ以外では FLRST ビットを操作できません(書き込みが無効です)。

FLRST レジスタ(リセット時:0x00):

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	FLRST
R	R	R	R	R	R	R	R/W

4.3 フラッシュ・メモリ・シーケンサの動作周波数設定

R_RFD_Init 関数で設定された FSSET レジスタ用の値(g_u08_fset_cpu_frequency)をフラッシュ・メモリ・シーケンサ初期設定レジスタ(FSSET)の FSET[bit4-0]へ設定します。

CPU が動作する周波数の値の小数点以下を切り上げた整数値を設定します。(例：CPU が動作する周波数が 4.5MHz の場合は、初期化関数で 5 を設定してください)

CPU の動作周波数を 4 MHz 未満で使用する場合は、2 MHz, 3 MHz を使用することができます。その際、整数値でない周波数(2.5MHz など)は使用できません。

本操作の対象関数：R_RFD_Init, R_RFD_SetFlashMemoryMode

≪操作方法≫

・フラッシュ・メモリ制御モードを「コード・フラッシュ・プログラミング・モード」、または「データ・フラッシュ・プログラミング・モード」へ移行します。移行手順は「4.1.1 特定シーケンス実行手順」、「4.1.2 非書き換えモードからコード・フラッシュ・プログラミング・モードに移行する手順」、「4.1.3 非書き換えモードからデータ・フラッシュ・プログラミング・モードに移行する手順」を参照してください。

・フラッシュ・メモリ・シーケンサ初期設定レジスタ(FSSET)を読み出し、TMSPMD[bit7],TMBTSEL[bit6]の値を保持、[bit5]を'0'、CPU の動作周波数[2~40(MHz)]の値を FSET にあたる[bit4-0]へ設定し、その値を FSSET レジスタへ書き込みます。

注）コード・フラッシュ・プログラミング・モード、またはデータ・フラッシュ・プログラミング・モードのとき、FSSET レジスタの FSET[bit4-0]は書き込みが有効です。それ以外では FSSET レジスタの FSET[bit4-0]を操作できません(書き込みが無効です)。

フラッシュ・メモリ・シーケンサを使用して、コード/データ・フラッシュ・メモリ、またはエクストラ領域へ、書き換え等の操作を実行する場合、FSSET レジスタの FSET へ CPU の動作周波数を設定しておく必要があります。

CPU の動作周波数が正しく設定されていない状態での書き換え動作は不定となり、書かれたデータは保証されませんので、ご注意ください。(書き込み直後のフラッシュ・メモリのデータ値が期待値通りであっても、その値の保持期間を保証できません。)

FSSET レジスタ(リセット時:0x00):

7	6	5	4	3	2	1	0
TMSPMD	**TMBTSEL**	0	FSET4	FSET3	FSET2	FSET1	FSET0
R/W	R/W	R	R/W	R/W	R/W	R/W	R/W

4.4 フラッシュ・メモリ・シーケンサ・コマンド

4.4.1 概要

RL78/F23,F24 のフラッシュ・メモリ・シーケンサは、コード・フラッシュ領域、またはデータ・フラッシュ領域を書き換えるコード/データ・フラッシュ領域シーケンサとエクストラ領域を書き換えるエクストラ領域シーケンサがあります。それぞれの領域を書き換えるには、各シーケンサのコマンドを実行する必要があります。また、「1.5 注意事項」の「(3)フラッシュ・メモリ書き換え操作中のプログラム実行」を参照、ご理解いただいた上で、フラッシュ・メモリ・シーケンサのコマンドを実行してください。

4.4.1.1 書き換え領域の選択

フラッシュ領域選択レジスタ(FLARS レジスタ)により、コード/データ・フラッシュ領域の書き換え時はユーザ領域を選択、エクストラ領域の書き換え時はエクストラ領域を選択する必要があり、FLARS レジスタの EXA[bit0]を操作して、書き込む領域を選択します。FLRST レジスタの FLRST[bit0] = 1 の期間中は、EXA ビットへの書き込みができません。

FLARS レジスタ(リセット時:0x00):

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	EXA
R	R	R	R	R	R	R	R/W

EXA = 0(リセット時)/1: ユーザ領域選択/エクストラ領域選択

4.4.2 コード/データ・フラッシュ領域シーケンサ・コマンド

コード/データ・フラッシュ領域の書き換えは、コード/データ・フラッシュ領域シーケンサの専用コマンドを使用します。コマンドの発行は、フラッシュ・メモリ・シーケンサ制御レジスタ(FSSQ)の SQMD2-0[bit2-0]へ対象のコマンド番号を入力するとともに、SQST[bit7]を'1'に設定します。

FSSQ レジスタ（リセット時:0x00）：

7	6	5	4	3	2	1	0
SQST	FSSTP	DCLR	0	MDCH	SQMD2	SQMD1	SQMD 0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

コード/データ・フラッシュ領域シーケンサの専用コマンドを表 4-1 に示します。

表 4-1 コード/データ・フラッシュ領域シーケンサの専用コマンド

SQMD2-0	MDCH 設定値	機能（専用コマンド） コマンドの説明
0H	CF:0 DF:0	コマンド未実行
1H	CF:0 DF:0	書き込み FLAPH/FLAPLレジスタで指定されるフラッシュ・メモリのアドレスに、FLWH/FLWLレジスタで指定したデータを書き込みます。 ・コード・フラッシュ（1ワード[4byte]）書き込み:FLWH/FLWLレジスタへ設定。 ・データ・フラッシュ(1byte)書き込み:FLWLレジスタのFLW7-0[bit7-0]へ設定。
2H	CF:0 DF:1	内部ベリファイ FLAPH/FLAPLレジスタで指定されるアドレスからFLSEDH/FLSEDLレジスタで指定されるアドレスまでの内部ベリファイを行います。内部ベリファイする対象のフラッシュ・メモリにより、FSSQレジスタのMDCH[bit3]の設定値が異なります。コード・フラッシュではMDCH[bit3] = 0、データ・フラッシュではMDCH[bit3] = 1を設定しておく必要があります。
3H	CF:0 DF:1	ブランク・チェック FLAPH/FLAPLレジスタで指定されるアドレスからFLSEDH/FLSEDLレジスタで指定されるアドレスまでのブランク・チェックを行います。ブランク・チェックする対象のフラッシュ・メモリにより、FSSQレジスタのMDCH[bit3]の設定値が異なります。コード・フラッシュではMDCH[bit3] = 0、データ・フラッシュではMDCH[bit3] = 1を設定しておく必要があります。
4H	CF:0 DF:0	ブロック消去 FLAPH/FLAPLレジスタで指定されるブロック先頭アドレスからFLSEDH/FLSEDLレジスタで指定されるブロック終了アドレスまでのブロック消去を行います。
その他の値	-	設定禁止

※CF：コード・フラッシュ・メモリ・アクセス時

DF：データ・フラッシュ・メモリ・アクセス時

・FLAPH/FLAPL レジスタ（フラッシュ・アドレス・ポインタ・レジスタ）

FLAPH レジスタ（リセット時:0x00）：

7	6	5	4	3	2	1	0
0	0	0	0	FLAP 19	FLAP 18	FLAP 17	FLAP 16
R	R	R	R	R/W	R/W	R/W	R/W

FLAPL レジスタ(リセット時:0x0000)：

15	14	13	12	11	10	9	8
FLAP 15	FLAP 14	FLAP 13	FLAP 12	FLAP 11	FLAP 10	FLAP 9	FLAP 8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

7	6	5	4	3	2	1	0
FLAP 7	FLAP 6	FLAP 5	FLAP 4	FLAP 3	FLAP 2	FLAP 1	FLAP 0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

・FLWH/FLWL レジスタ（フラッシュ・ライト・バッファ・レジスタ）

FLWH レジスタ（リセット時:0x0000）：

15	14	13	12	11	10	9	8
FLW 31	FLW 30	FLW 29	FLW 28	FLW 27	FLW 26	FLW 25	FLW 24
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

7	6	5	4	3	2	1	0
FLW 23	FLW 22	FLW 21	FLW 20	FLW 19	FLW 18	FLW 17	FLW 16
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

FLWL レジスタ（リセット時:0x0000）：

15	14	13	12	11	10	9	8
FLW 15	FLW 14	FLW 13	FLW 12	FLW 11	FLW 10	FLW 9	FLW 8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

7	6	5	4	3	2	1	0
FLW 7	FLW 6	FLW 5	FLW 4	FLW 3	FLW 2	FLW 1	FLW 0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

※データ指定時は、各ビットの機能ごとに設定方法が異なるため、注意が必要です。

・FLSEDH/FLSEDL レジスタ（フラッシュ・エンド・アドレス指定レジスタ）

FLSEDH レジスタ（リセット時:0x00）：

7	6	5	4	3	2	1	0
0	0	0	0	EWA 19	EWA 18	EWA 17	EWA 16
R	R	R	R	R/W	R/W	R/W	R/W

FLSEDL レジスタ（リセット時:0x0000）：

15	14	13	12	11	10	9	8
EWA 15	EWA 14	EWA 13	EWA 12	EWA 11	EWA 10	EWA 9	EWA 8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

7	6	5	4	3	2	1	0
EWA 7	EWA 6	EWA 5	EWA 4	EWA 3	EWA 2	EWA 1	EWA 0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

4.4.2.1 コード・フラッシュ領域書き換えの操作

コード・フラッシュ領域の書き換えは、フラッシュ・メモリ制御モードをコード・フラッシュ・プログラミング・モードに移行後、コード/データ・フラッシュ領域シーケンサ・コマンドを実行します。各コマンド実行に必要な指定アドレスやデータをあらかじめ該当レジスタに設定してから、コマンドを開始する必要があります。

コード・フラッシュ領域書き換え時の消去ブロック単位/書き込み単位

- 消去ブロック単位：1**Kbyte**
- 書き込み単位：**1 ワード[4byte]**

本操作の対象関数：R_RFD_EraseCodeFlashReq, R_RFD_WriteCodeFlashReq,
R_RFD_BlankCheckCodeFlashReq, R_RFD_IVerifyCodeFlashReq

≪操作方法≫

対象コマンドは、コード・フラッシュのブロック消去、書き込み、ブランク・チェック、内部ベリファイです。

・**コード・フラッシュ・プログラミング・モード**に移行します。移行手順は「4.1.1 特定シーケンス実行手順」、および「4.1.2 非書き換えモードからコード・フラッシュ・プログラミング・モードに移行する手順」を参照してください。

・FLARS レジスタ = 0x00(EXA[bit0] = 0)：**「ユーザ領域選択」**を設定します。

・各コマンド実行前に、対象レジスタへ指定データを設定します。

(1) ブロック消去

FLAPH/FLAPL レジスタ：コード・フラッシュ・メモリのブロック先頭アドレス（例:0x002000）

FLSEDH/FLSEDL レジスタ：コード・フラッシュ・メモリのブロック終了アドレス（例:0x0023FF）

(2) 書き込み:1 ワード[4byte]単位のため、アドレスは、[bit1-0]が'0'の 4 の倍数を設定する必要があります。

FLAPH/FLAPL レジスタ：対象のフラッシュ・メモリの先頭アドレス（例:0x002000）

FLSEDH/FLSEDL レジスタ： ALL'0'、または未設定（例:0x000000）

FLWH/FLWL レジスタ：書き込むデータ（1 ワード[4byte]）

(3) ブランク・チェック: 1 ワード[4byte]単位のため、アドレスは、[bit1-0]が'0'の 4 の倍数を設定する必要があります。

FLAPH/FLAPL レジスタ：対象のフラッシュ・メモリの先頭アドレス（例:0x002000）

FLSEDH/FLSEDL レジスタ：対象のフラッシュ・メモリの終了アドレス（例:0x0023FF）

※1 ワード[4byte]のみブランク・チェックする場合は、FLAPH/FLAPL = FLSEDH/FLSEDL を設定します。

(4) 内部ベリファイ

FLAPH/FLAPL レジスタ：コード・フラッシュ・メモリのブロック先頭アドレス（例:0x002000）

FLSEDH/FLSEDL レジスタ：コード・フラッシュ・メモリのブロック終了アドレス（例:0x0023FF）

※書き込み直後の領域に 1 回だけ実行することができます。複数回実行しないでください。

・FSSQ の SQMD2-0[bit2-0]へ対象コマンド番号を入力するとともに、SQST[bit7]を'1'に設定します。

ブロック消去：0x84、書き込み：0x81、ブランク・チェック：0x83、内部ベリファイ：0x82

・コード/データ・フラッシュ領域シーケンサ・コマンドの完了を待ちます。コマンドの完了待ち手順は、「4.4.4.1 コード/データ・フラッシュ領域シーケンサ・コマンドの終了判定手順」を参照してください。

・コマンド実行後の処理

コマンド処理を継続する場合：

コード・フラッシュ・プログラミング・モードに移行したまま、対象レジスタのデータを更新して同一コマンド実行や、他のコード・フラッシュ領域書き換えコマンドを実行することも可能です。

コマンド処理を完了する場合：

非書き換え・モードに移行します。移行手順は「4.1.1 特定シーケンス実行手順」、および「4.1.4 非書き換えモード移行手順」を参照してください。

4.4.2.2 データ・フラッシュ領域書き換えの操作

データ・フラッシュ領域の書き換えは、フラッシュ・メモリ制御モードをデータ・フラッシュ・プログラミング・モードに移行後、コード/データ・フラッシュ領域シーケンサ・コマンドを実行します。各コマンド実行に必要な指定アドレスやデータをあらかじめ該当レジスタに設定してから、コマンドを開始する必要があります。

データ・フラッシュ領域書き換え時の消去ブロック単位/書き込み単位

- 消去ブロック単位：**1Kbyte**
- 書き込み単位：**1byte**

本操作の対象関数：R_RFD_EraseDataFlashReq, R_RFD_WriteDataFlashReq,
R_RFD_BlankCheckDataFlashReq, R_RFD_IVerifyDataFlashReq

≪操作方法≫

対象コマンドは、データ・フラッシュのブロック消去、書き込み、ブランク・チェック、内部ベリファイです。

・**データ・フラッシュ・プログラミング・モード**に移行します。移行手順は「4.1.1 特定シーケンス実行手順」、および「4.1.3 非書き換えモードからデータ・フラッシュ・プログラミング・モードに移行する手順」を参照してください。

・FLARS レジスタ = 0x00(EXA[bit0] = 0)：**「ユーザ領域選択」**を設定します。

・各コマンド実行前に、対象レジスタへ指定データを設定します。

(1) ブロック消去

FLAPH/FLAPL レジスタ：データ・フラッシュ・メモリのブロック先頭アドレス（例:0x0F1400）

FLSEDH/FLSEDL レジスタ：データ・フラッシュ・メモリのブロック終了アドレス（例:0x0F17FF）

(2) 書き込み:1byte

FLAPH/FLAPL レジスタ：対象のフラッシュ・メモリの先頭アドレス（例:0x0F1101）

FLSEDH/FLSEDL レジスタ： ALL'0'、または未設定（例:0x000000）

FLWH/FLWL レジスタ：書き込むデータを設定(0x00000000-0x000000FF) FLW7-0[bit7-0]のみ有効です。

(3) ブランク・チェック

FLAPH/FLAPL レジスタ：対象のフラッシュ・メモリの先頭アドレス（例:0x0F1400）

FLSEDH/FLSEDL レジスタ：対象のフラッシュ・メモリの終了アドレス（例:0x0F17FF）

※1byte のみブランク・チェックする場合は、FLAPH/FLAPL = FLSEDH/FLSEDL を設定します。

(4) 内部ベリファイ

FLAPH/FLAPL レジスタ：データ・フラッシュ・メモリの先頭アドレス(例:0x0F1400)

FLSEDH/FLSEDL レジスタ：データ・フラッシュ・メモリの終了アドレス(例:0x0F15FF)

※書き込み直後の領域に 1 回だけ実行することができます。複数回実行しないでください。

・FSSQ の SQMD2-0[bit2-0]へ対象コマンド番号を入力するとともに、SQST[bit7]を'1'に設定します。
ブロック消去：0x84、書き込み：0x81、ブランク・チェック：**0x8B（MDCH[bit3] = 1:DF の場合のみ）、**
内部ベリファイ：**0x8A（MDCH[bit3] = 1:DF の場合のみ）**

・コード/データ・フラッシュ領域シーケンサ・コマンドの完了を待ちます。コマンドの完了待ち手順は、「4.4.4.1 コード/データ・フラッシュ領域シーケンサ・コマンドの終了判定手順」を参照してください。

・コマンド実行後の処理
コマンド処理を継続する場合：
データ・フラッシュ・プログラミング・モードに移行したまま、対象レジスタのデータを更新して同一コマンド実行や、他のデータ・フラッシュ領域書き換えコマンドを実行することも可能です。
コマンド処理を完了する場合：
非書き換え・モードに移行します。移行手順は「4.1.1 特定シーケンス実行手順」、および「4.1.4 非書き換えモード移行手順」を参照してください。

4.4.3 エクストラ領域シーケンサ・コマンド

エクストラ領域の書き換えは、フラッシュ・メモリ制御モードをコード・フラッシュ・プログラミング・モードに移行後、エクストラ領域シーケンサの専用コマンドを使用します。コマンドの発行は、フラッシュ・エクストラ領域シーケンサ制御レジスタ(FSSE)の ESQMD2-0[bit2-0]へ対象のコマンド番号を入力するとともに、ESQST[bit7]を'1'に設定します。

FSSE レジスタ（リセット時:0x00）:

7	6	5	4	3	2	1	0
ESQST	0	0	0	0	ESQMD2	ESQMD1	ESQMD0
R/W	R	R	R	R	R/W	R/W	R/W

エクストラ領域シーケンサの専用コマンドを表 4-2 に示します。

表 4-2 エクストラ領域シーケンサの専用コマンド

ESQMD3-0	機能（専用コマンド）
	コマンドの説明
0H	コマンド未実行
1H	エクストラ領域書き込み（セキュリティ・フラグとブート領域切替フラグのプログラミング）
	FLWH/FLWLレジスタで指定したデータを書き込みます。セキュリティ・フラグとブート領域切替フラグの設定を行います。セキュリティ・フラグは、現フラグに対して禁止の設定のみ可能です。ブート・プロテクトが設定されている場合(BTPR=0)、ブート領域切替フラグは設定できません。
2H	エクストラ領域書き込み（FSW範囲設定のプログラミング）
	FLWH/FLWLレジスタで指定したデータを書き込みます。FSWの設定を行います。
その他の値	設定禁止

4.4.3.1 エクストラ領域の書き換えの操作

エクストラ領域の書き換えは、フラッシュ・メモリ制御モードをコード・フラッシュ・プログラミング・モードに移行後、エクストラ領域シーケンサ・コマンドを実行します。各コマンド実行に必要なデータをあらかじめ該当レジスタに設定してから、コマンドを開始する必要があります。

エクストラ領域書き換え時の書き込み単位

- 書き込み単位：1 ワード[4byte]

※消去コマンド、消去単位はありません。

本操作の対象関数：R_RFD_SetExtraEraseProtectReq, R_RFD_SetExtraWriteProtectReq,
R_RFD_SetExtraBootAreaProtectReq, R_RFD_SetExtraBootAreaReq, ,
R_RFD_SetExtraFSWReq

≪操作方法≫

対象コマンドは、エクストラ領域データの書き込みです。

・**コード・フラッシュ・プログラミング・モード**に移行します。移行手順は「4.1.1 特定シーケンス実行手順」、および「4.1.2 コード・フラッシュ・プログラミング・モード移行手順」を参照してください。

・FLARS レジスタ = 0x01(EXA[bit0] = 1)：**「エクストラ領域選択」**を設定します。

・コマンド実行前に FLWH/FLWL レジスタへ 1 ワード[4byte]のデータを指定します。FLWH/FLWL レジスタの各ビット(FLW31-FLW0)は、対象のエクストラ領域データの[EX bit31-0]に対応します。

FLWH レジスタ（リセット時:0x0000）：

15	14	13	12	11	10	9	8
FLW 31	FLW 30	FLW 29	FLW 28	FLW 27	FLW 26	FLW 25	FLW 24
7	6	5	4	3	2	1	0
FLW 23	FLW 22	FLW 21	FLW 20	FLW 19	FLW 18	FLW 17	FLW 16

FLWL レジスタ（リセット時:0x0000）：

15	14	13	12	11	10	9	8
FLW 15	FLW 14	FLW 13	FLW 12	FLW 11	FLW 10	FLW 9	FLW 8
7	6	5	4	3	2	1	0
FLW 7	FLW 6	FLW 5	FLW 4	FLW 3	FLW 2	FLW 1	FLW 0

※データ指定時は、各ビットの機能ごとに設定方法が異なるため、注意が必要です。

・書き込む領域はコマンドで指定し、FSSE の ESQMD2-0[bit2-0]へ対象コマンド番号を入力するとともに、ESQST[bit7]を'1'に設定します。

(1) セキュリティ・フラグとブート領域切替フラグのプログラミング:0x81

エクストラ領域シーケンサ・コマンドの完了を待ちます。コマンドの完了待ち手順は、「4.4.4.2 エクストラ領域シーケンサ・コマンドの終了判定手順」を参照してください。

(2) FSW 範囲設定のプログラミング:0x82

・コマンド実行後の処理

コマンド処理を継続する場合：

コード・フラッシュ・プログラミング・モードに移行したまま、対象レジスタのデータを更新して同一コマンド実行や、他のエクストラ領域書き換えコマンドを実行することも可能です。

コマンド処理を完了する場合：

非書き換え・モードに移行します。移行手順は「4.1.1 特定シーケンス実行手順」、および「4.1.4 非書き換えモード移行手順」を参照してください。

4.4.3.2 エクストラ領域シーケンサ・コマンドの設定データ

エクストラ領域の書き込みは、変更しないデータも含めて 1 ワード[4byte]単位で行います。対象コマンドごとに示すエクストラ領域データの[EX bit31-0]を FLWH/FLWL レジスタの[FLW31-FLW0]へ設定してからコマンドを実行します。

(1) FSW 関連データのプログラミング

以下のエクストラ領域データの[EX bit31-0]を FLWH/FLWL レジスタの[FLW31-FLW0]へ設定します。

EX bit 31	EX bit 30	EX bit 29	EX bit 28	EX bit 27	EX bit 26	EX bit 25	EX bit 24
-	-	-	-	-	-	FSWE9	FSWE8
EX bit 23	EX bit 22	EX bit 21	EX bit 20	EX bit 19	EX bit 18	EX bit 17	EX bit 16
FSWE7	FSWE6	FSWE5	FSWE4	FSWE3	FSWE2	FSWE1	FSWE0
EX bit 15	EX bit 14	EX bit 13	EX bit 12	EX bit 11	EX bit 10	EX bit 9	EX bit 8
-	-	-	-	-	-	FSWS9	FSWS8
EX bit 7	EX bit 6	EX bit 5	EX bit 4	EX bit 3	EX bit 2	EX bit 1	EX bit 0
FSWS7	FSWS6	FSWS5	FSWS4	FSWS3	FSWS2	FSWS1	FSWS0

- FSWE9-0[bit25-16]は、ウインドウ範囲の[エンド・ブロック] +1 です。
- FSWS9-0[bit9-0]は、ウインドウ範囲の[スタート・ブロック]です。

(2) セキュリティ・フラグとブート領域切替フラグのプログラミング

以下のエクストラ領域データの[EX bit31-0]を FLWH/FLWL レジスタの[FLW31-FLW0]へ設定します。

EX bit 31	EX bit 30	EX bit 29	EX bit 28	EX bit 27	EX bit 26	EX bit 25	EX bit 24
1	1	1	1	1	1	1	1
EX bit 23	EX bit 22	EX bit 21	EX bit 20	EX bit 19	EX bit 18	EX bit 17	EX bit 16
1	1	1	1	1	1	1	1
EX bit 15	EX bit 14	EX bit 13	EX bit 12	EX bit 11	EX bit 10	EX bit 9	EX bit 8
1	1	1	WRPR	1	SEPR	BTPR	BTFLG
EX bit 7	EX bit 6	EX bit 5	EX bit 4	EX bit 3	EX bit 2	EX bit 1	EX bit 0
1	1	1	1	1	1	1	1

- WRPR[bit12]は、シリアル・プログラミング・モードでの書き込み禁止を設定します。

  WRPR = 0 / 1(出荷時)：シリアル・プログラミング・モードでの書き込みを**禁止**/書き込みを**許可**

- SEPR[bit10]は、シリアル・プログラミング・モードでのブロック消去禁止を設定します。

  SEPR = 0 / 1(出荷時) ：シリアル・プログラミング・モードでのブロック消去を**禁止**/ブロック消去を**許可**

- BTPR[bit9]は、シリアル/セルフ・プログラミング両方でのブート領域の書き換え禁止を設定します。

  BTPR = 0 / 1(出荷時)：ブート領域の書き換え**禁止**/ブート領域の書き換えを**許可**

- BTFLG[bit8]は、TMSPMD = 0[ブート・スワップはエクストラ領域のブート領域切替フラグ(BTFLG)に従う]の場合のブート領域に設定するブート・クラスタを制御します。

  BTFLG = 0 / 1(出荷時)：ブート領域はブート・クラスタ 1 /ブート領域はブート・クラスタ 0

注意　1. BTFLG を書き換える場合、その他の全てのビットは' 1'を設定してください。

2. BTFLG 以外のセキュリティ・フラグを'0'(禁止)に書き換える場合、BTFLG(読み込んだ値と同じ値を設定)を除き、その他の全てのビットは' 1'を設定してください。

3. WRPR=0(禁止)に設定した場合、シリアル・プログラミング・モードのチップ消去コマンドを実行した場合のみ、WRPR=1(許可)にすることができます。

※但し、以下のいずれかの禁止が設定されている場合は、シリアル・プログラミング・モードのチップ消去コマンドを実行できません。

・SEPR = 0（ブロック消去禁止）

・BTPR = 0（ブート領域書き換え禁止）

4.4.4 フラッシュ・メモリ・シーケンサ・コマンドの終了判定手順

RL78/F23,F24 で起動したフラッシュ・メモリ・シーケンサ・コマンドを終了する場合、特定の終了判定手順を実行する必要があります。

フラッシュ・メモリ・シーケンサ・コマンドの終了判定では、FSASTH レジスタの ESQEND[bit7]、または SQEND[bit6]を参照、”1”にセットされたことを確認して、コード/データ・フラッシュ領域シーケンサ・コマンド、及びエクストラ領域シーケンサ・コマンドの終了判定手順を実施します。終了判定実施後、コマンドごとのエラーの発生有無を FSASTL レジスタのエラービット(BLER[bit3], IVER[bit2], WRER[bit1], ERER[bit0])で確認します。

FSASTH レジスタ（リセット時:0x00）：

7	6	5	4	3	2	1	0
ESQEND	SQEND	0	0	0	0	0	0
R	R	R	R	R	R	R	R

FSASTL レジスタ（リセット時:0x00）：

7	6	5	4	3	2	1	0
MBTSEL	0	ESEQER	SEQER	BLER	IVER	WRER	ERER
R	R	R	R	R	R	R	R

注) ブート・フラグ・モニタ・ビット(MBTSEL[bit7])は、エクストラ領域のブート領域切替フラグ(BTFLG[bit8])の反転値が反映されます。

4.4.4.1 コード/データ・フラッシュ領域シーケンサ・コマンドの終了判定手順

≪終了判定手順≫

(1) コード/データ・フラッシュ領域シーケンサ・コマンド起動後、FSASTH レジスタの SQEND[bit6]が自動的にセットされるまで待ちます。
(2) FSASTH レジスタの SQEND[bit6]のセット確認後、FSSQ レジスタの SQST[bit7] をクリアします。
(3) FSASTH レジスタの SQEND[bit6]が自動的にクリアされるまで待ち、クリアされたら終了します。

4.4.4.2 エクストラ領域シーケンサ・コマンドの終了判定手順

≪終了判定手順≫

(1) エクストラ領域シーケンサ・コマンド起動後、FSASTH レジスタの ESQEND[bit7]が自動的にセットされるまで待ちます。
(2) FSASTH レジスタの ESQEND[bit7]のセット確認後、FSSE レジスタの ESQST[bit7] をクリアします。
(3) FSASTH レジスタの ESQEND[bit7]が自動的にクリアされるまで待ち、クリアされたら終了します。

4.4.5 コード/データ・フラッシュ領域シーケンサ・コマンドの強制終了手順

コード/データ・フラッシュ領域シーケンサ・コマンド実行中、コマンドを強制的に完了しなければならない非常事態が発生した場合、コマンドを強制停止することができます。

※エクストラ領域シーケンサ・コマンド実行中は、コマンドを強制終了することができません。

≪強制終了手順≫

(1) 「4.4.4.1 コード/データ・フラッシュ領域シーケンサ・コマンドの終了判定手順」”(1)”のコマンド起動後から”(2)”の FSSQ レジスタの SQST[bit7] をクリアする前までに、FSSQ レジスタの FSSTP[bit6]=1 に設定することで、起動したコード/データ・フラッシュ領域シーケンサ・コマンドを強制停止します。

(2) FSASTH レジスタの SQEND[bit6]のセット確認後、FSSQ レジスタの SQST[bit7] と FSSTP[bit6]をクリアします。

(3) FSASTH レジスタの SQEND[bit6]が自動的にクリアされるまで待ち、クリアされたら終了します。

FSSQ レジスタ(リセット時:0x00) :

7	6	5	4	3	2	1	0
SQST	FSSTP	DCLR	0	MDCH	SQMD2	SQMD1	SQMD 0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

4.5 ブート・スワップ機能

4.5.1 概要

ベクタ・テーブル・データ、プログラムの基本機能、およびセルフ・プログラミングを実行するブート・プログラムを配置しているブート領域[0x00000 - 0x03FFF]の書き換え中に、電源の瞬断、外部要因によるリセットの発生などにより書き換えが失敗した場合、書き換え中のデータが破壊され、その後のリセットによるユーザ・プログラムの再スタートや、再書き込みができなくなります。この問題を回避するための機能がブート・スワップ機能です。

4.5.2 ブート・スワップ機能の動作

ブート・スワップ機能では、ブート領域であるブート・クラスタ 0 [0x00000 - 0x03FFF]とブート・スワップ対象領域であるブート・クラスタ 1 [0x04000 - 0x07FFF]を置換します。書き換え処理を行う前に、あらかじめ新しいブート・プログラムをブート・クラスタ 1 [0x04000 - 0x07FFF]に書き込んでおきます。このブート・クラスタ 1 [0x04000 - 0x07FFF]とブート・クラスタ 0 [0x00000 - 0x03FFF]をスワップし、ブート・クラスタ 1 をブート領域[0x00000 - 0x03FFF]にします。これによって、ブート領域の書き換え中に電源の瞬断が発生しても、次のリセット・スタートは新しいブート・プログラムが書き込まれているブート・クラスタ 1 [0x00000 - 0x03FFF]から実行されるため、正常にプログラムが動作します。

ブート領域	リセット・ベクタを含む 0x00000 – 0x03FFF の論理領域。
ブート・クラスタ 0/1	ブート・クラスタは16Kバイトのブロック群で、0/1どちらか片方がブート領域に配置されます。 物理領域名： ブート・クラスタ0 [0x00000 – 0x03FFF]（出荷時の論理アドレス） ブート・クラスタ1 [0x04000 – 0x07FFF]（出荷時の論理アドレス）

注) ブート・スワップを実行後、ブート・クラスタ 0 とブート・クラスタ 1 の論理アドレスが入れ替わります。

BTPR=1、かつコード・フラッシュ・プログラミング・モード、またはデータ・フラッシュ・プログラミング・モードのとき、FSSET レジスタの TMSPMD[bit7]と TMBTSEL[bit6]は書き込みが有効です。それ以外では FSSET レジスタの TMSPMD[bit7]と TMBTSEL[bit6]を操作できません（書き込みが無効です）。

ブート・スワップ機能の動作は、フラッシュ・メモリ・シーケンサ初期設定レジスタ(FSSET)の TMSPMD[bit7]の値によりエクストラ領域のブート領域切替フラグ(BTFLG)、または FSSET レジスタの TMBTSEL[bit6]に従います。

- TMSPMD[bit7]が'0'の時（リセット時）、ブート領域はエクストラ領域の BTFLG に従います。
  BTFLG = 0 / 1（出荷時）：ブート領域はブート・クラスタ 1 /ブート領域はブート・クラスタ 0
- TMSPMD[bit7]が'1'の時、ブート領域は FSSET レジスタの TMBTSEL[bit6]に従います。
  TMBTSEL = 0（リセット時）/ 1：ブート領域はブート・クラスタ 0 /ブート領域はブート・クラスタ 1

FSSET レジスタ(リセット時:0x00):

7	6	5	4	3	2	1	0
TMSPMD	**TMBTSEL**	0	FSET4	FSET3	FSET2	FSET1	FSET0
R/W	R/W	R	R/W	R/W	R/W	R/W	R/W

4.5.3 ブート・スワップ機能の操作

ブート・スワップ機能の操作には、「即時ブート・スワップを実行」する方法と「リセット後にブート・スワップを実行」する方法があります。

注意：**FSSET レジスタを書き込む場合（TMSPMD ビット、TMBTSEL ビットを操作する場合）、FSSET レジスタの FSET4-0（CPU の動作周波数）の値を破壊しないよう、FSSET レジスタを書き込む前に FSSET レジスタを読み出し、その値を反映して FSSET レジスタに書き込んでください。**
FSSET レジスタに正しい CPU の動作周波数が設定されていない場合、フラッシュ・メモリ・シーケンサの動作は不定となり、書き換えられたフラッシュ・メモリの値は保証されませんのでご注意ください。

4.5.3.1 即時ブート・スワップを実行

指定されたブート・クラスタを、ブート領域(0x00000~0x03FFF)に即時設定（ブート・スワップ）します。
※BTPR=0 の場合、TMSPMD ビットの設定ができないため、ブート・スワップを実行できません。
本操作の対象関数：R_RFD_SetBootAreaImmediately
≪操作方法≫
(1) TMSPMD ビット=0（BTFLG に従う）の場合：
・FSAST レジスタの MBTSEL ビットを読み出し、その値を FSSET レジスタの TMBTSEL ビットに設定します。
a) TMBTSEL ビット=1 の場合：
・TMSPMD ビットに’1’（TMBTSEL に従う）、TMBTSEL ビットに’0’を設定します。ブート・スワップが即時実行されます。
b) TMBTSEL ビット=0 の場合：
・TMSPMD ビットに’1’（TMBTSEL に従う）、TMBTSEL ビットに’1’を設定します。ブート・スワップが即時実行されます。
(2) TMSPMD ビット=1（TMBTSEL に従う）の場合：
a) TMBTSEL ビット=1 の場合：
・TMBTSEL ビットに’0’を設定します。ブート・スワップが即時実行されます。
b) TMBTSEL ビット=0 の場合：
・TMBTSEL ビットに’1’を設定します。ブート・スワップが即時実行されます。

4.5.3.2 リセット後にブート・スワップを実行

BTFLG が書かれた直後にブート・スワップせず、リセット後にブート・スワップを実行します。

※BTPR=0 の場合、TMSPMD ビットの設定、エクストラ領域書き込みによる BTFLG の設定が共にできないため、ブート・スワップを実行できません。

本操作の対象関数：R_RFD_SetExtraBootAreaReq

≪操作方法≫

(1) TMSPMD ビット = 0（BTFLG に従う）の場合：

・FLSEC レジスタの BTFLG ビットを読み出します。

a) FLSEC レジスタの BTFLG ビット=0 の場合：

・TMSPMD ビットに'1'（TMBTSEL に従う）、TMBTSEL ビットに'1'を設定します。

・**エクストラ領域**の BTFLG を書き込みます。（ブート領域に設定するブート・クラスタを指定します。FSSE レジスタの ESQMD = 1H）

・リセット操作を実行後にブート・スワップし、指定ブート・クラスタのリセット・ベクタへ分岐します。

b) FLSEC レジスタの BTFLG ビット=1 の場合：

・TMSPMD ビットに'1'（TMBTSEL に従う）、TMBTSEL ビットに'0'を設定します。

・**エクストラ領域**の BTFLG を書き込みます。（ブート領域に設定するブート・クラスタを指定します。FSSE レジスタの ESQMD = 1H）

・リセット操作を実行後にブート・スワップし、指定ブート・クラスタのリセット・ベクタへ分岐します。

(2) TMSPMD ビット = 1（TMBTSEL に従う）の場合：

・**エクストラ領域**の BTFLG を書き込みます。（ブート領域に設定するブート・クラスタを指定します。FSSE レジスタの ESQMD = 1H）

・リセット操作を実行後にブート・スワップし、指定ブート・クラスタのリセット・ベクタへ分岐します。

4.6 フラッシュ・シールド・ウインドウ機能

4.6.1 概要

フラッシュ・シールド・ウインドウ(FSW)機能は、指定したウインドウ範囲以外の書き込みおよび消去を、セルフ・プログラミング時のみ禁止にするセキュリティ機能です。FSW 機能のウインドウ範囲は、スタート・ブロックとエンド・ブロック+1 で指定します。

4.6.2 フラッシュ・シールド・ウインドウ機能の動作

FSW 機能の動作は、フラッシュ FSW モニタ・レジスタ(FLFSWE / FLFSWS)の値により決定し、FLFSWE / FLFSWS の値は、エクストラ領域に書込まれた FSW 用の情報が反映されます。FSW の設定を変更する場合、エクストラ領域シーケンサで FSW 設定用のエクストラ領域へ設定データを書き込みます。

FLFSWE レジスタ(リセット時、またはエクストラ領域書き込み時に**対象のエクストラ領域の値が反映**):

15	14	13	12	11	10	9	8
0	0	0	0	0	0	FSWE9	FSWE8
R	R	R	R	R	R	R	R
7	6	5	4	3	2	1	0
FSWE7	FSWE6	FSWE5	FSWE4	FSWE3	FSWE2	FSWE1	FSWE0
R	R	R	R	R	R	R	R

FLFSWS レジスタ(リセット時、またはエクストラ領域書き込み時に**対象のエクストラ領域の値が反映**):

15	14	13	12	11	10	9	8
0	0	0	0	0	0	FSWS9	FSWS8
R	R	R	R	R	R	R	R
7	6	5	4	3	2	1	0
FSWS7	FSWS6	FSWS5	FSWS4	FSWS3	FSWS2	FSWS1	FSWS0
R	R	R	R	R	R	R	R

- FLFSWE レジスタの FSWE[bit9-0]は、ウインドウ範囲のエンド・ブロック番号+1 を指定します。
- FLFSWS レジスタの FSWS[bit9-0]は、ウインドウ範囲のスタート・ブロック番号を指定します。

4.6.3 フラッシュ・シールド・ウインドウ機能の操作

4.6.3.1 フラッシュ・シールド・ウインドウの設定

本操作の対象関数：R_RFD_SetExtraFSWReq

≪操作方法≫

・**エクストラ領域**の FSWE, FSWS を書き込みます。(FSSE レジスタの ESQMD = 1H)

FSWE：FSW のウインドウ範囲のエンド・ブロック番号＋1

FSWS：FSW のウインドウ範囲のスタート・ブロック番号

※予約ビット[bit15-10]は'0'に設定してください。FSWS = FSWE を設定した場合、コード・フラッシュ・メモリの全領域が書き換え許可となります。

例）対象デバイス：R7F124FPJ

スタート・ブロック：03H、エンド・ブロック：05H をウインドウ範囲に指定

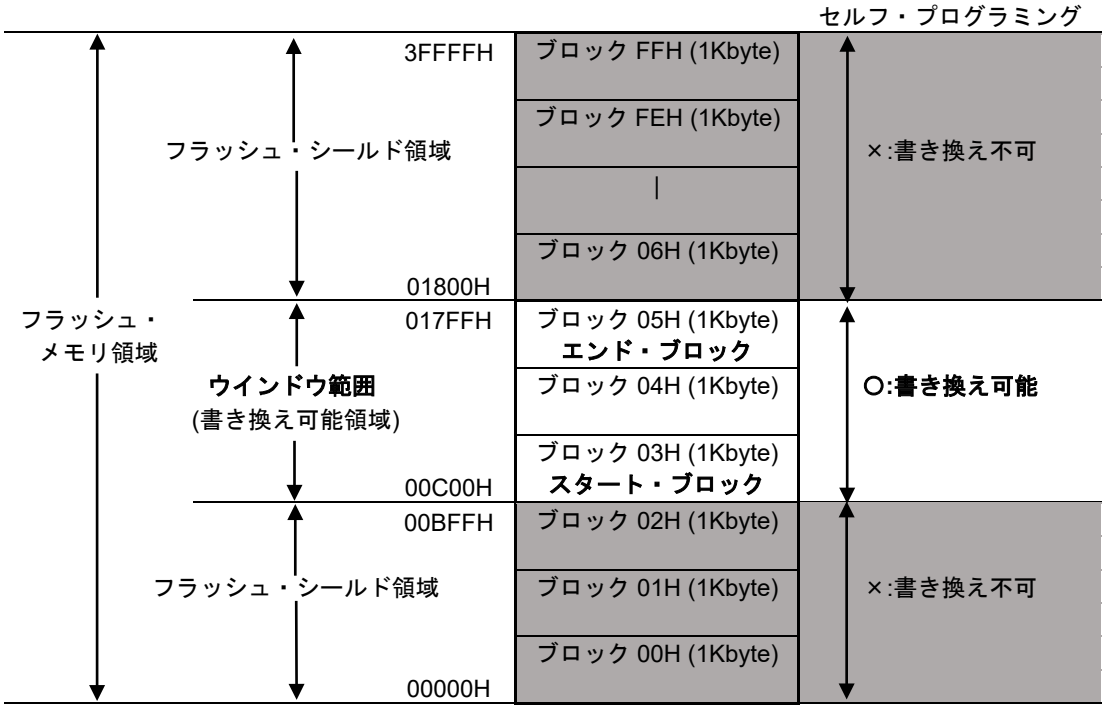


図 4-1 FSW の設定例

4.7 フラッシュ領域書き換え時のコマンドの実行例

4.7.1 コード・フラッシュ領域書き換え時のコマンド実行例

コード・フラッシュ領域書き換え時のコマンド実行フローを図 4-2 に示します。

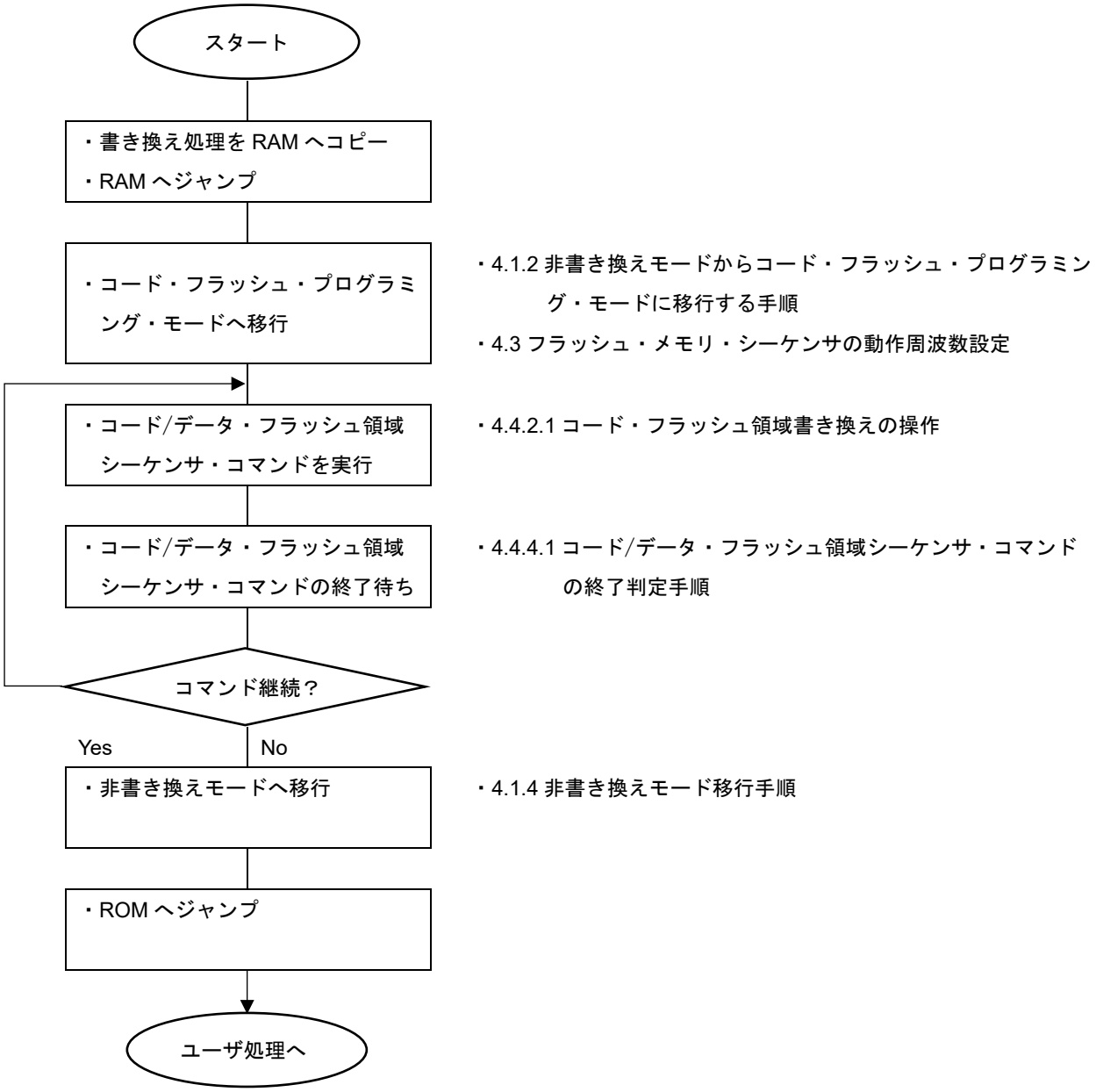


図 4-2 コード・フラッシュ領域書き換え時のコマンド実行フロー

4.7.2 データ・フラッシュ領域書き換え時のコマンド実行例

データ・フラッシュ領域書き換え時のコマンド実行フローを図 4-3 に示します。

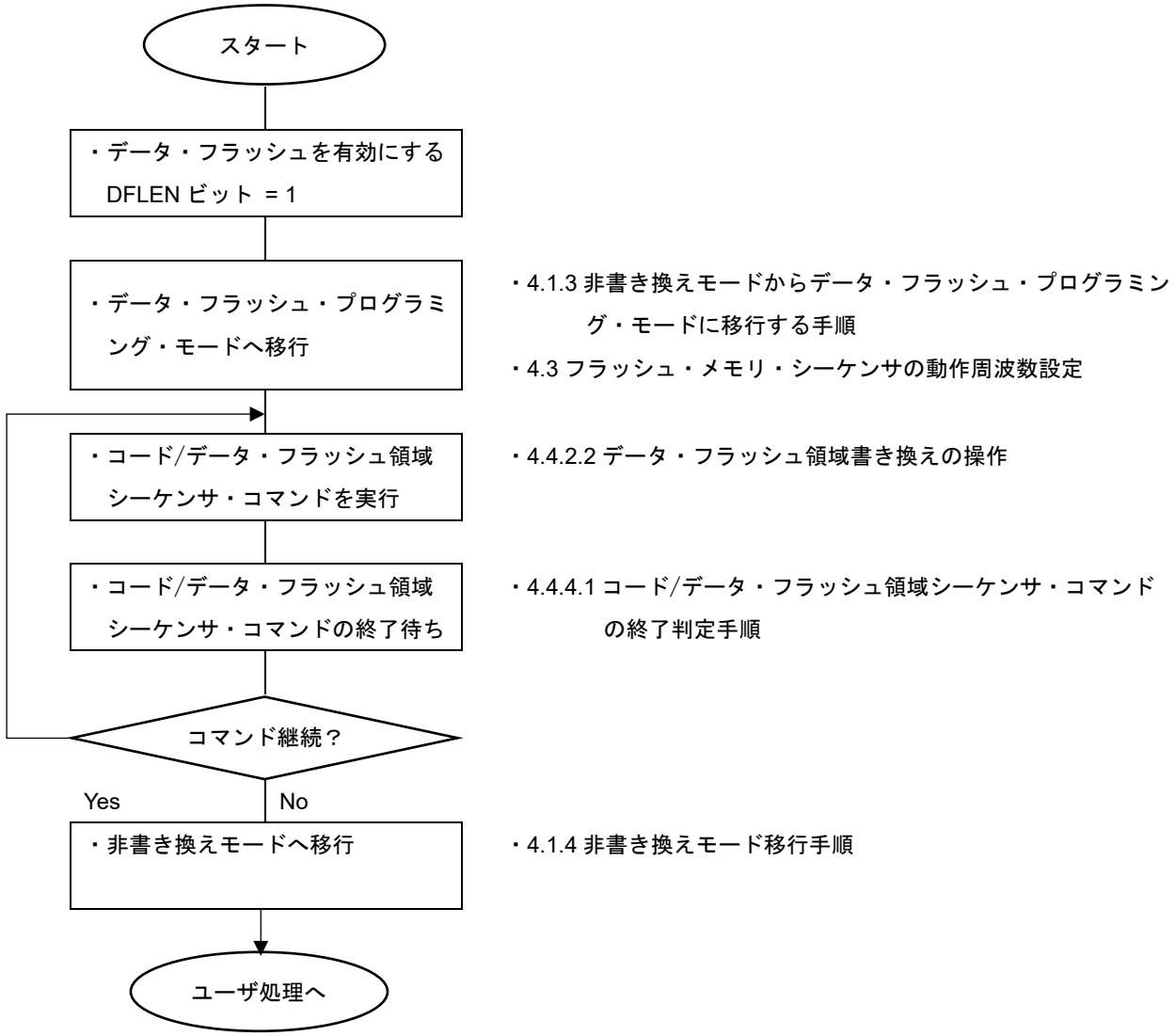


図 4-3 データ・フラッシュ領域書き換え時のコマンド実行フロー

4.7.3 エクストラ領域書き換え時のコマンド実行例

エクストラ領域書き換え時のコマンド実行フローを図 4-4 に示します。

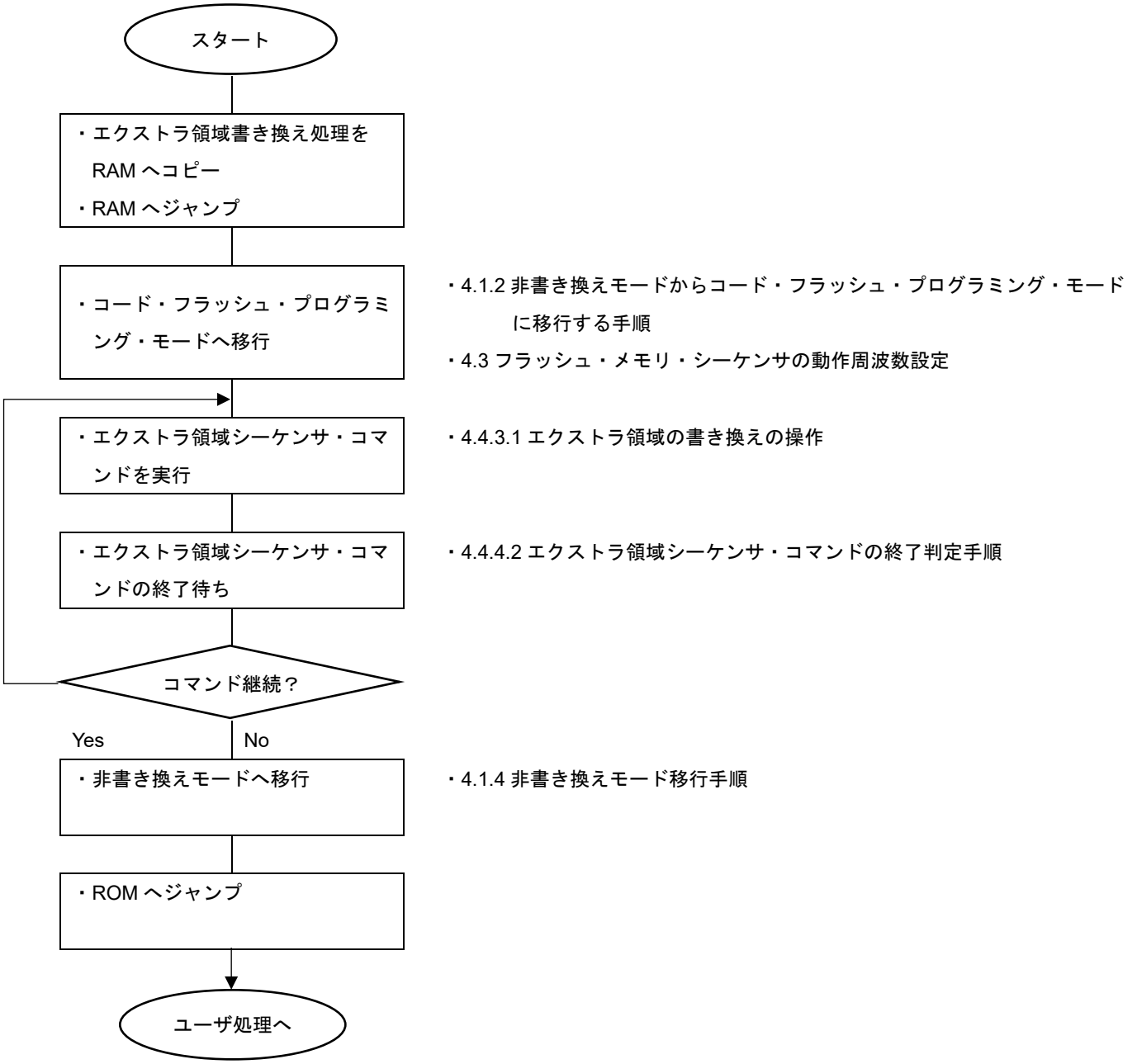


図 4-4 エクストラ領域書き換え時のコマンド実行フロー

5 サンプル・プログラム

RFD RL78 Type02 に添付しているサンプル・プログラムについて説明します。

5.1 ファイル構成

5.1.1 フォルダ構成

サンプル・プログラムのフォルダ構成を図 5-1 に示します。

図 5-1 は、RL78/F24 を使用する場合の例です。実際にインストールした"sample"フォルダには、デバイスグループごとのサンプル用フォルダが含まれます（例: RL78_F24）。

RL78_F24 フォルダは、RL78/F23, F24 で使用することができます。

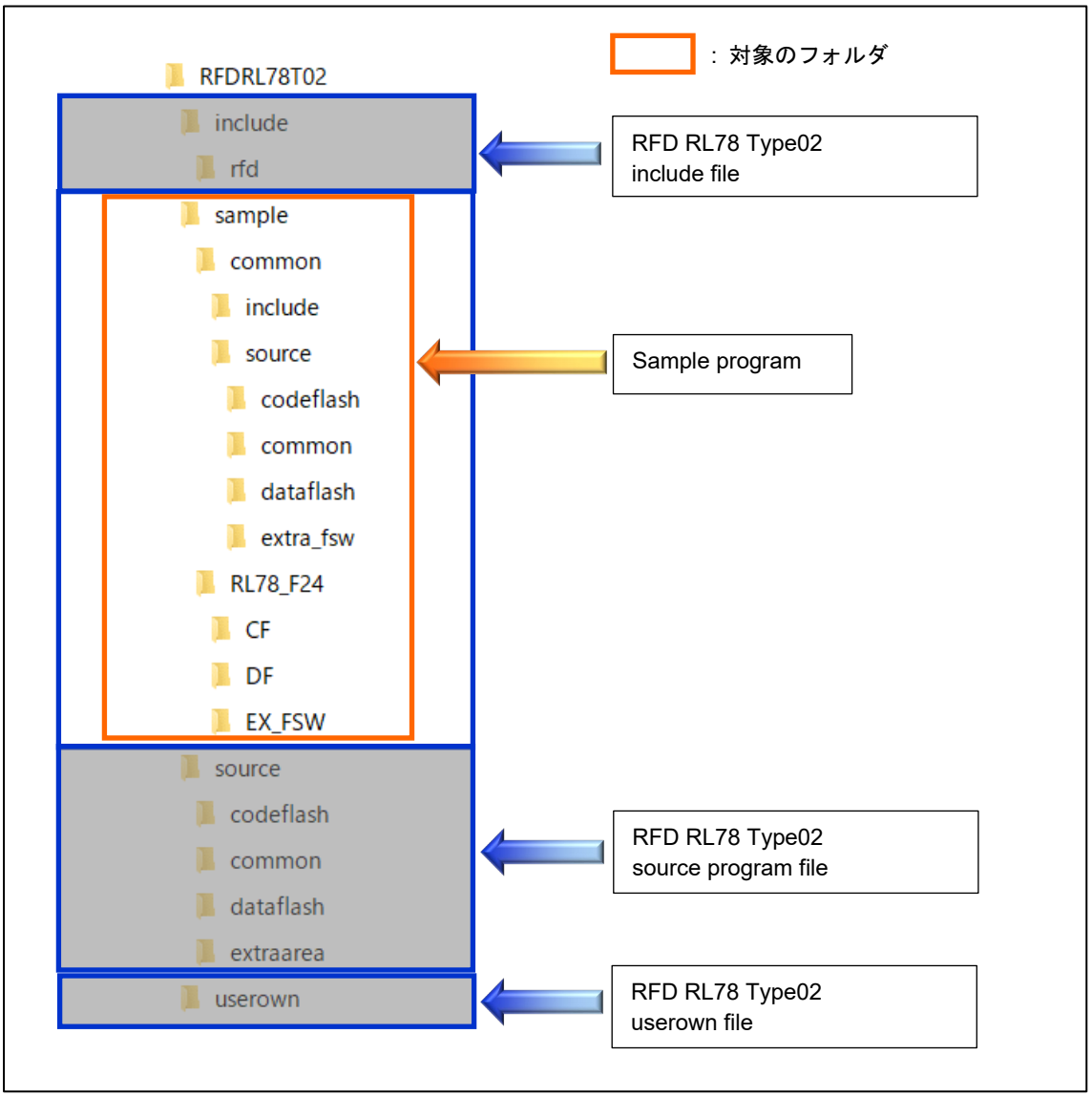


図 5-1 サンプル・プログラムのフォルダ構成

5.1.2 ファイル・リスト

5.1.2.1 ソース・ファイル・リスト

“sample\common\source\common\”フォルダ内のプログラム・ソース・ファイルを表 5-1 に示します。

表 5-1 “sample\common\source\common\”フォルダ内プログラム・ソース・ファイル

No	ソース・ファイル名	概略(Summary)
1	sample_control_common.c	共通フラッシュ・メモリ制御用関数サンプル・ファイル

“sample\common\source\dataflash\”フォルダ内のプログラム・ソース・ファイルを表 5-2 に示します。

表 5-2 “sample\common\source\dataflash\”フォルダ内プログラム・ソース・ファイル

No	ソース・ファイル名	概略(Summary)
1	sample_control_data_flash.c	データ・フラッシュ・メモリ制御用関数サンプル・ファイル

“sample\common\source\codeflash\”フォルダ内のプログラム・ソース・ファイルを表 5-3 に示します。

表 5-3 “sample\common\source\codeflash\”フォルダ内プログラム・ソース・ファイル

No	ソース・ファイル名	概略(Summary)
1	sample_control_code_flash.c	コード・フラッシュ・メモリ制御用関数サンプル・ファイル

“sample\common\source\extra_fsw\”フォルダ内のプログラム・ソース・ファイルを表 5-4 に示します。

表 5-4 “sample\common\source\extra_fsw\”フォルダ内プログラム・ソース・ファイル

No	ソース・ファイル名	概略(Summary)
1	sample_control_extra_fsw.c	エクストラ領域 FSW 制御用関数サンプル・ファイル

“sample\RL78_F24”フォルダ内のコード・フラッシュ制御[CF]、データ・フラッシュ制御[DF]、エクストラ領域 FSW 制御[EX_FSW]の各メイン処理のプログラム・ソース・ファイルを表 5-5 に示します。

- コード・フラッシュ[CF]のメイン処理：“sample\RL78_F24\CF\[コンパイラ名]\source\”フォルダ
- データ・フラッシュ[DF]のメイン処理：“sample\RL78_ F24\DF\[コンパイラ名]\source\”フォルダ
- エクストラ領域 FSW 制御処理[EX_FSW]のメイン処理：
  “sample\RL78_ F24\EX_FSW\[コンパイラ名]\source\”フォルダ

表 5-5 メイン処理のプログラム・ソース・ファイル

No	ソース・ファイル名	概略(Summary)
1	main.c (for code flash)	コード・フラッシュ制御用メイン処理関数サンプル・ファイル
2	main.c (for data flash)	データ・フラッシュ制御用メイン処理関数サンプル・ファイル
3	main.c (for extra area FSW control)	エクストラ領域(FSW 機能)制御用メイン処理関数サンプル・ファイル

5.1.2.2 ヘッダ・ファイル・リスト

“sample\common\include\”フォルダ内のプログラム・ヘッダ・ファイルを表 5-6 に示します。

表 5-6 “sample\common\include\”フォルダ内プログラム・ヘッダ・ファイル

No	ヘッダ・ファイル名	概略(Summary)
1	sample_control_common.h	共通フラッシュ・メモリ制御用関数サンプルのプロトタイプ宣言を定義したファイル
2	sample_control_data_flash.h	データ・フラッシュ・メモリ制御用関数サンプルのプロトタイプ宣言を定義したファイル
3	sample_control_code_flash.h	コード・フラッシュ・メモリ制御用関数サンプルのプロトタイプ宣言を定義したファイル
4	sample_control_extra_fsw.h	エクストラ領域 FSW 制御用関数サンプルのプロトタイプ宣言を定義したファイル
5	sample_defines.h	フラッシュ・メモリ制御用関数サンプルのマクロを定義したファイル
6	sample_memmap.h	フラッシュ・メモリ制御用関数サンプルで使用するセクションを記述するためのマクロを定義したファイル
7	sample_types.h	サンプル・プログラム用列挙型返り値を定義したファイル

5.2 データ型定義

5.2.1 列挙型

- e_sample_ret (列挙変数名 : e_sample_ret_t)

フラッシュ・メモリ・シーケンサ実行結果(正常/エラー)と実行後ステータスを表 5-7 に示します。

表 5-7 フラッシュ・メモリ・シーケンサ実行結果(正常/エラー)と実行後ステータス

Symbol Name	Value	Description
SAMPLE_ENUM_RET_STS_OK	0x00u	ステータス(正常終了)
SAMPLE_ENUM_RET_ERR_PARAMETER	0x10u	パラメータ・エラー
SAMPLE_ENUM_RET_ERR_CONFIGURATION	0x11u	初期設定エラー
SAMPLE_ENUM_RET_ERR_MODE_MISMATCHED	0x12u	モード不一致エラー
SAMPLE_ENUM_RET_ERR_CHECK_WRITE_DATA	0x13u	書き込みデータ確認エラー
SAMPLE_ENUM_RET_ERR_CFDF_SEQUENCER	0x20u	コード/データ・フラッシュ領域シーケンサ・エラー
SAMPLE_ENUM_RET_ERR_EXTRA_SEQUENCER	0x21u	エクストラ領域シーケンサ・エラー
SAMPLE_ENUM_RET_ERR_ACT_ERASE	0x22u	消去動作エラー
SAMPLE_ENUM_RET_ERR_ACT_WRITE	0x23u	書き込み動作エラー
SAMPLE_ENUM_RET_ERR_ACT_BLANKCHECK	0x24u	ブランク・チェック動作エラー
SAMPLE_ENUM_RET_ERR_ACT_IVERIFY	0x25u	内部ベリファイ動作エラー
SAMPLE_ENUM_RET_ERR_CMD_ERASE	0x30u	消去コマンド・エラー
SAMPLE_ENUM_RET_ERR_CMD_WRITE	0x31u	書き込みコマンド・エラー
SAMPLE_ENUM_RET_ERR_CMD_BLANKCHECK	0x32u	ブランク・チェック・コマンド・エラー
SAMPLE_ENUM_RET_ERR_CMD_IVERIFY	0x33u	内部ベリファイ・コマンド・エラー
SAMPLE_ENUM_RET_ERR_CMD_SET_EXTRA_AREA	0x34u	エクストラ領域コマンド設定エラー

5.3 サンプル・プログラム関数

サンプル・プログラム関数一覧を表 5-8 に示します。

表 5-8 サンプル・プログラム関数一覧

	API Name	Outline
1	main (for code flash)	コード・フラッシュ書き換え制御サンプル・プログラムのメイン処理
2	Sample_CodeFlashControl	コード・フラッシュ・メモリの書き換え処理を実行
3	main (for data flash)	データ・フラッシュ書き換え制御サンプル・プログラムのメイン処理
4	Sample_DataFlashControl	データ・フラッシュ・メモリの書き換え処理を実行
5	main (for extra area FSW control)	エクストラ領域(FSW 機能)書き換え制御サンプル・プログラムのメイン処理
6	Sample_ExtraFSWControl	エクストラ領域(FSW 機能)の書き換え処理を実行
7	Sample_CheckCFDFSeqEnd	コード/データ・フラッシュ領域シーケンサ・コマンドの完了待ち処理
8	Sample_CheckExtraSeqEnd	エクストラ領域シーケンサ・コマンドの完了待ち処理

5.3.1 コード・フラッシュ書き換え制御サンプル・プログラム

RFD RL78 Type02 のコード・フラッシュ書き換え制御サンプルでは、コード・フラッシュ領域のブロック 28(0x00007000)を消去し、ブロック 28 の先頭から 256 ワード(1024byte)のデータを書き込み、書き換えたブロックの内部ベリファイをします。

注)コード・フラッシュ・プログラミング・モード中は、コード・フラッシュ上のプログラムを実行できないため、Sample_CodeFlashControl 関数、およびその内部で実行される処理と参照するデータは事前に RAM へコピーして、RAM 上で実行、参照する必要があります。

動作条件(RL78/F24 用サンプル・プログラムの例):

・CPU 動作周波数: 40MHz(メイン・システム・クロックに高速オンチップ・オシレータ・クロックを使用)

・コード・フラッシュ消去/書き込みアドレス: 0x00007000

・消去ブロック No.: 0x001C

・書き込みデータ・サイズ: 256 ワード(1024byte)

RFD RL78 Type02 のコード・フラッシュ書き換え制御サンプルのメイン処理実行フローを図 5-2 に示します。

5.3.1.1 main 関数

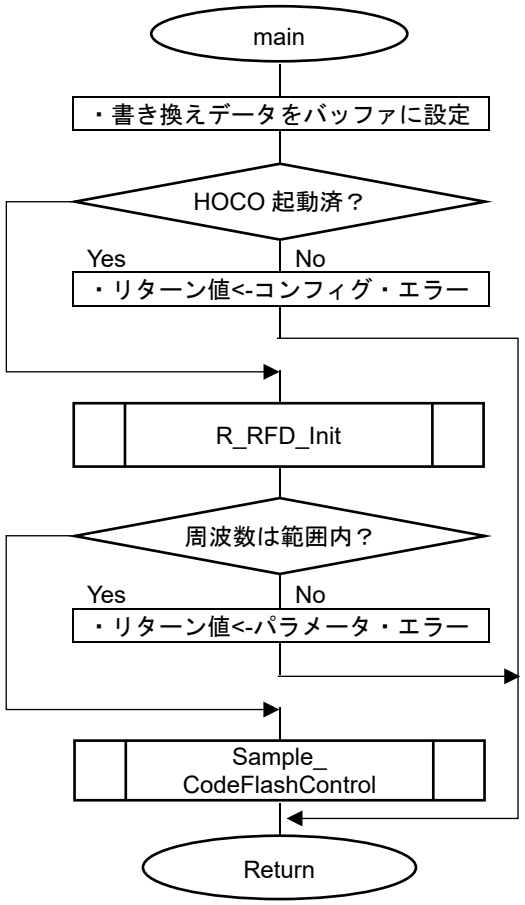


図 5-2 コード・フラッシュ書き換え制御サンプルのメイン処理実行フロー

5.3.1.2 Sample_CodeFlashControl 関数

・コード・フラッシュ・プログラミング・モードへ移行、ブランク・チェック、ブロック消去を実行

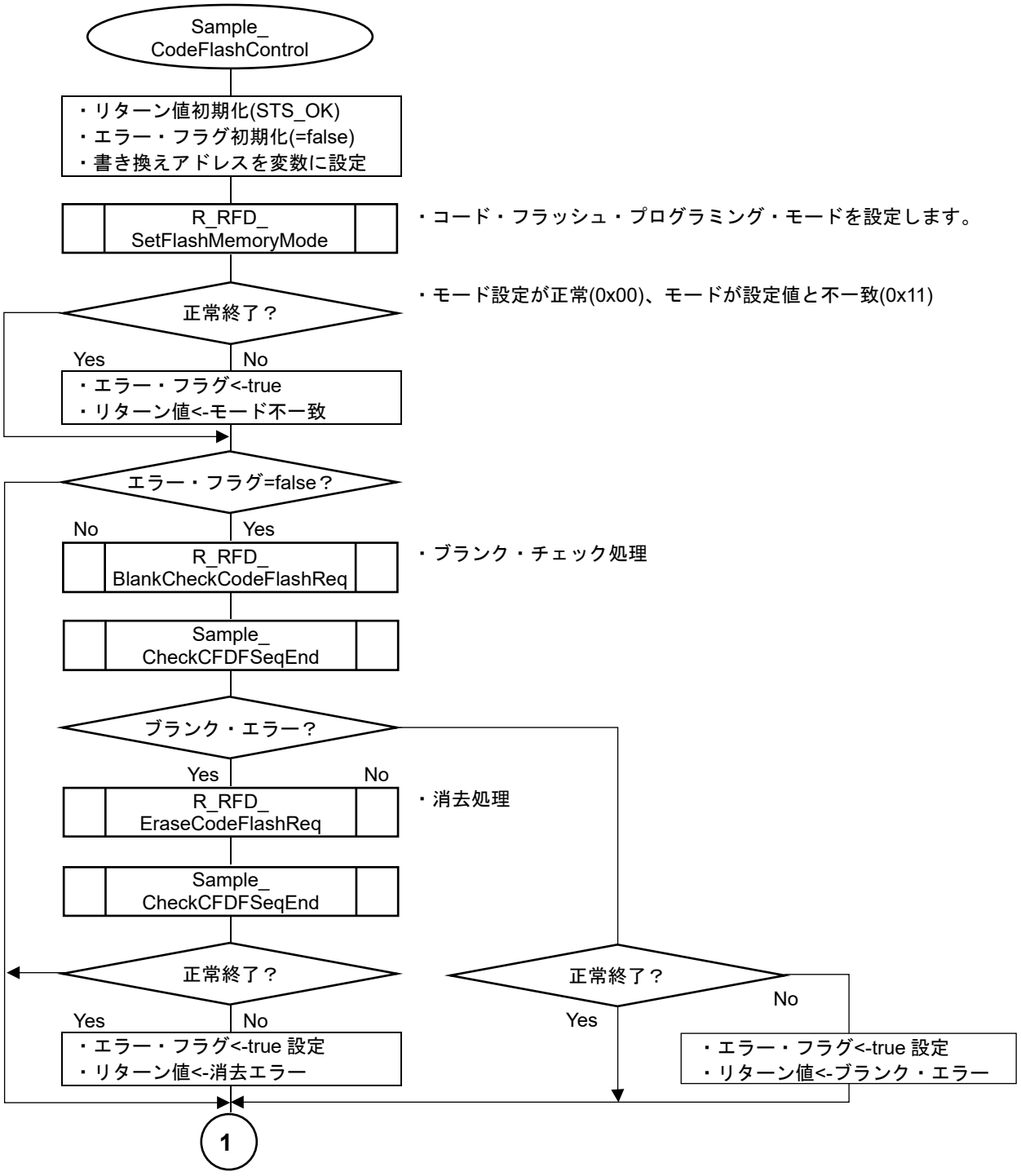


図 5-3 コード・フラッシュ書き換え制御サンプルの処理実行フロー(1/3)

・書き込みと内部ベリファイを実行

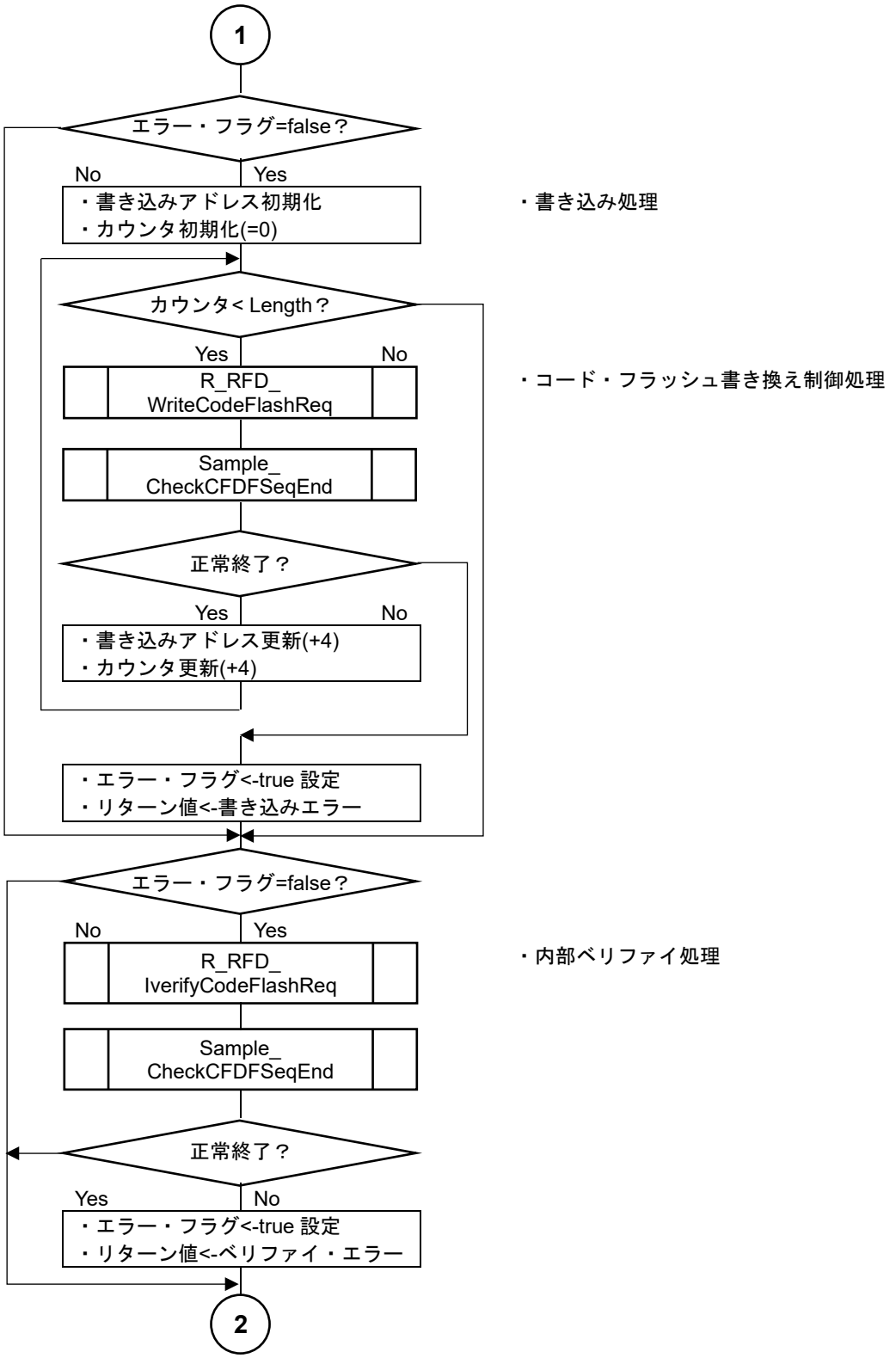
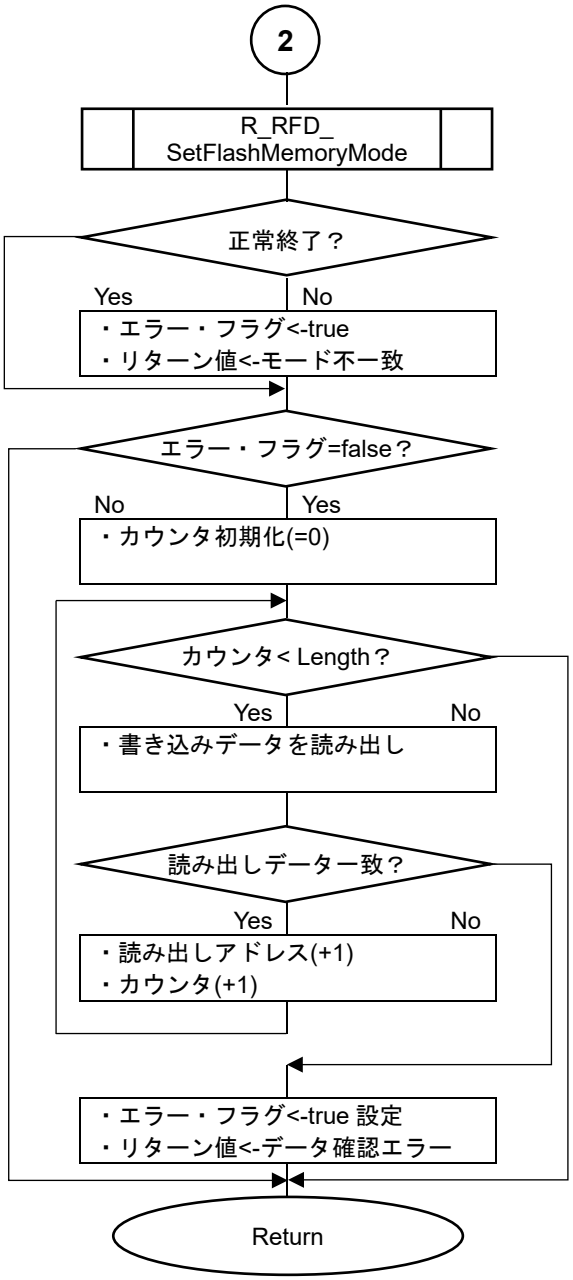


図 5-4 コード・フラッシュ書き換え制御サンプルの処理実行フロー(2/3)

・非書き換えモードへ移行、CPU 読み出しによるベリファイ・チェックを実行



・非書き換えモードを設定します。

・モード設定が正常(0x00)、モードが設定値と不一致(0x11)

・CPU 読み出しによるベリファイ・チェック

図 5-5 コード・フラッシュ書き換え制御サンプルの処理実行フロー(3/3)

5.3.2 データ・フラッシュ書き換え制御サンプル・プログラム

RFD RL78 Type02 のデータ・フラッシュ書き換え制御サンプルでは、データ・フラッシュ領域のブロック 0(0x000F1000)を消去し、ブロック 0 の先頭から 64byte のデータを書き込み、書き換えたアドレスの内部ベリファイをします。

注)データ・フラッシュ・プログラミング・モード中は、データ・フラッシュ上のデータを参照できないため、Sample_DataFlashControl 関数、およびその内部で参照するデータは、事前に RAM へコピーして、RAM 上で参照する必要があります。

動作条件(RL78/F24 用サンプル・プログラムの例):

- CPU 動作周波数: 40MHz(メイン・システム・クロックに高速オンチップ・オシレータ・クロックを使用)
- データ・フラッシュ消去/書き込みアドレス: 0x000F1000
- 消去ブロック No.: 0x0000
- 書き込みデータ・サイズ: 64byte

RFD RL78 Type02 のデータ・フラッシュ書き換え制御サンプルのメイン処理実行フローを図 5-6 に示します。

5.3.2.1 main 関数

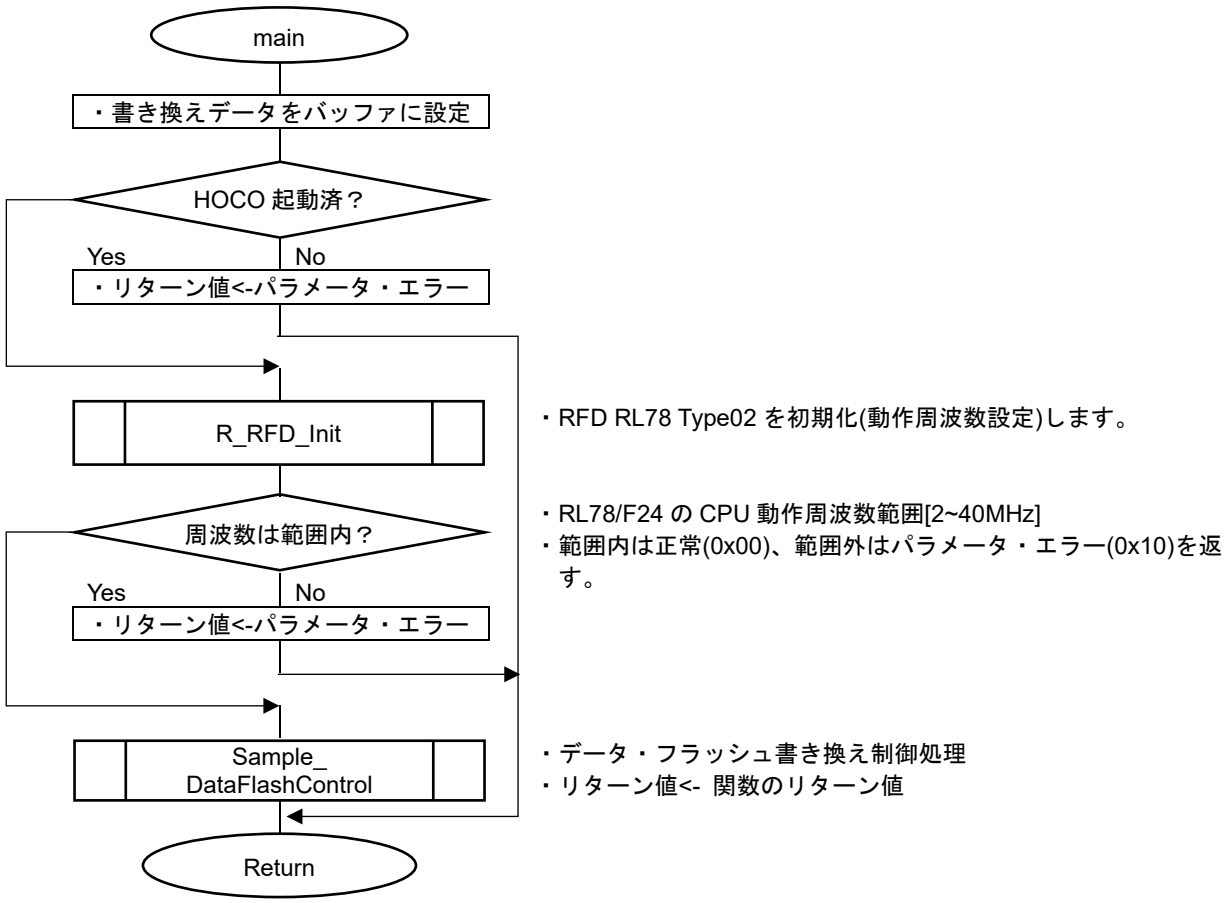


図 5-6 データ・フラッシュ書き換え制御サンプルのメイン処理実行フロー

5.3.2.2 Sample_DataFlashControl 関数

・データ・フラッシュ・プログラミング・モードへ移行、ブランク・チェック、ブロック消去を実行

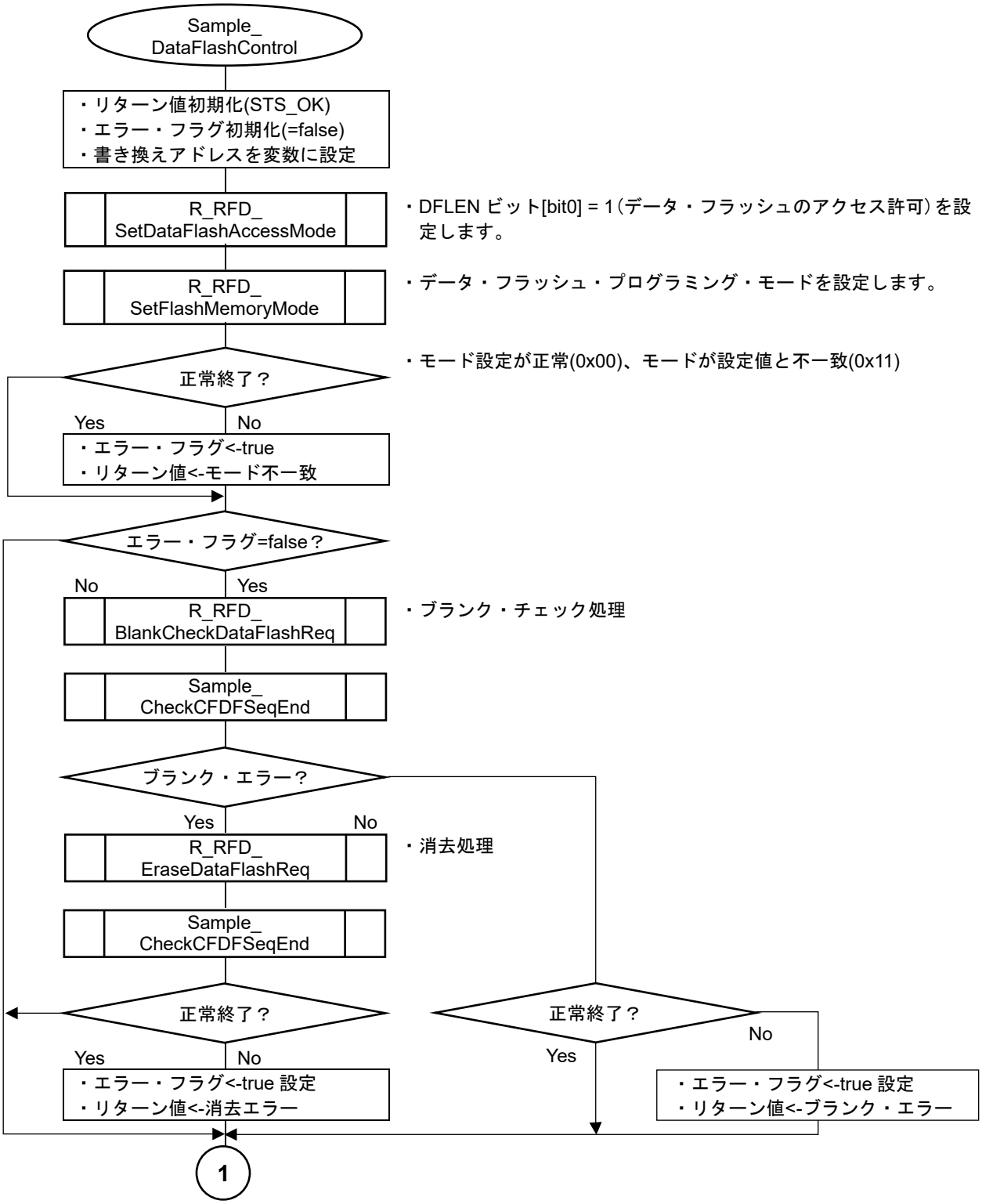


図 5-7 データ・フラッシュ書き換え制御サンプルの処理実行フロー(1/3)

・書き込みと内部ベリファイを実行

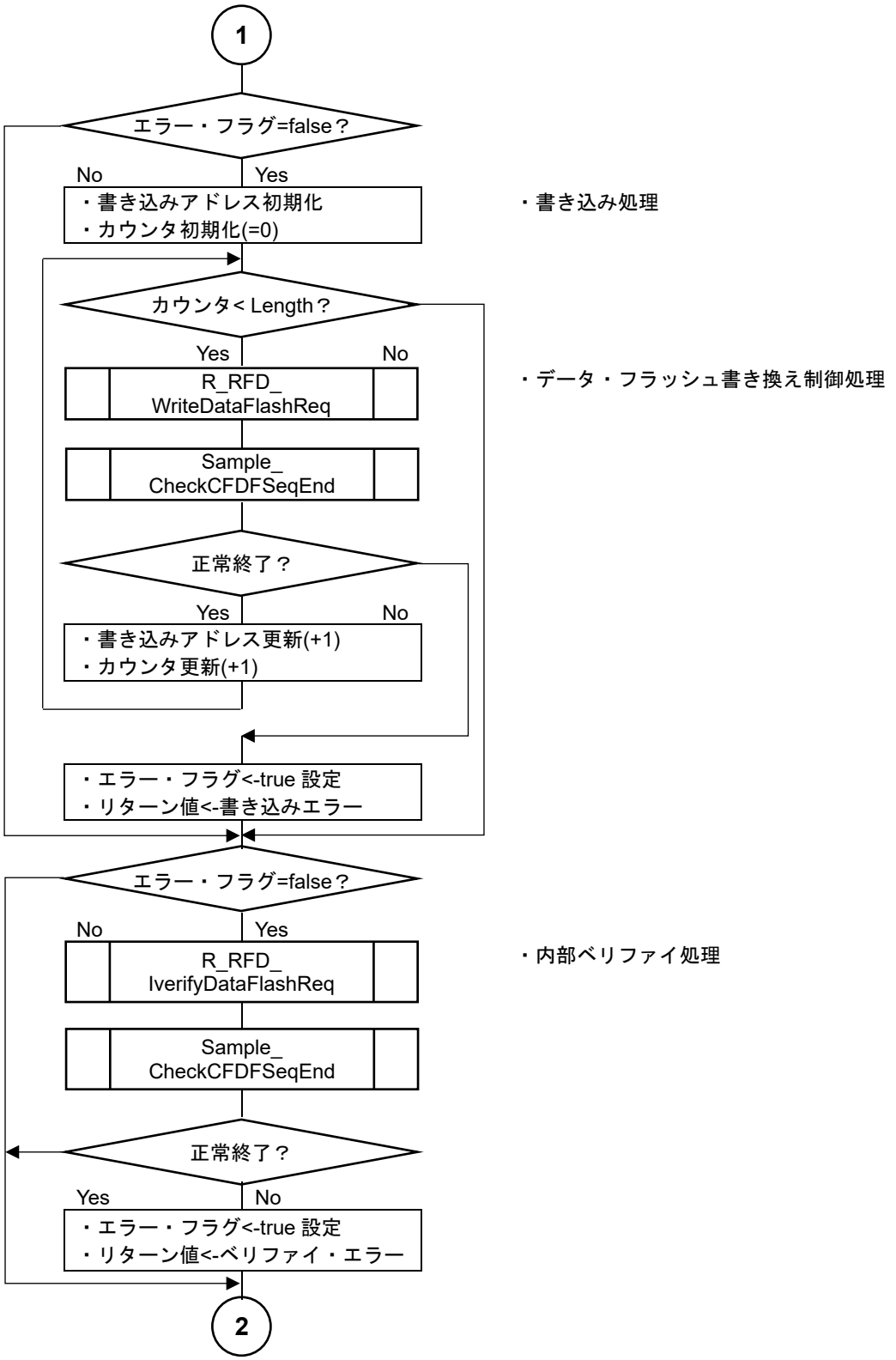
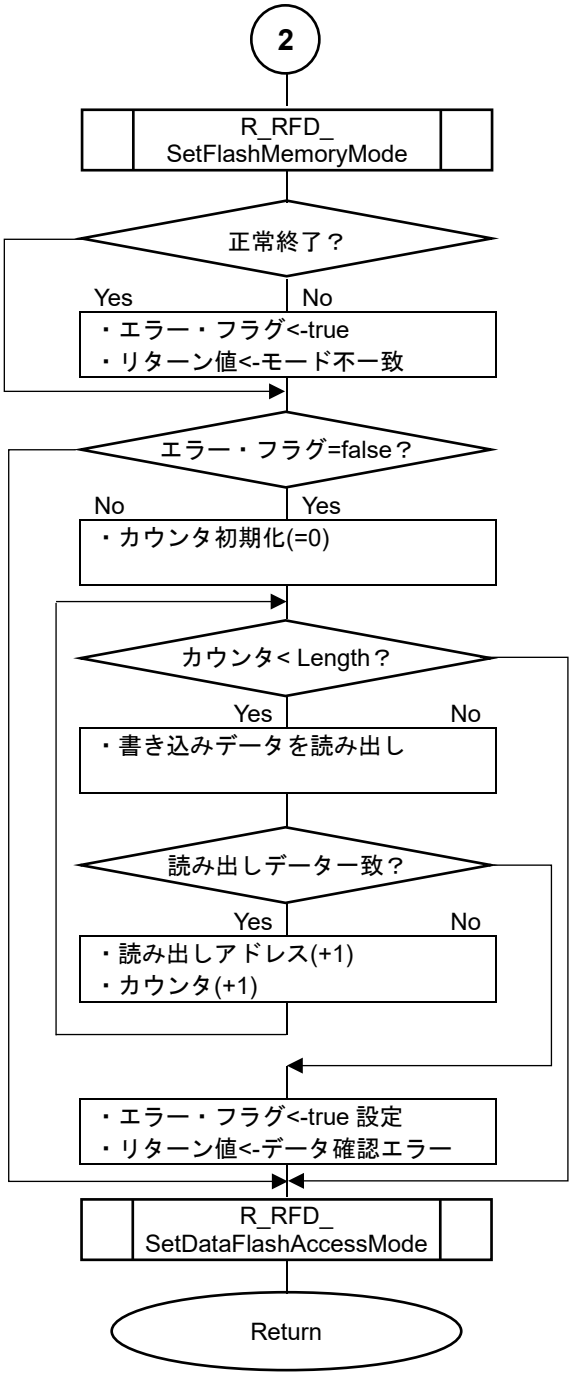


図 5-8 データ・フラッシュ書き換え制御サンプルの処理実行フロー(2/3)

・非書き換えモードへ移行、CPU 読み出しによるベリファイ・チェックを実行



・非書き換えモードを設定します。

・モード設定が正常(0x00)、モードが設定値と不一致(0x11)

・CPU 読み出しによるベリファイ・チェック

・DFLEN ビット[bit0] = 0（データ・フラッシュのアクセス禁止）を設定します。

図 5-9 データ・フラッシュ書き換え制御サンプルの処理実行フロー(3/3)

5.3.3 エクストラ領域書き換え制御サンプル・プログラム

RFD RL78 Type02 のエクストラ領域書き換え制御サンプルでは、フラッシュ・シールド・ウインドウ(FSW)を制御する 4byte(32bit)の領域を書き換えます。

・FSWS[スタート・ブロック] = 0, FSWE[エンド・ブロック+1] = 256

（コード・フラッシュ領域のブロック 0 からブロック 255 の書き換えを許可）

注)エクストラ領域書き換え時のコード・フラッシュ・プログラミング・モード中は、コード・フラッシュ上のプログラムを実行できないため、Sample_ExtraFSWControl 関数、およびその内部で実行される処理と参照するデータは事前に RAM へコピーして、RAM 上で実行、参照する必要があります。

動作条件（RL78/F24 用サンプル・プログラムの例）:

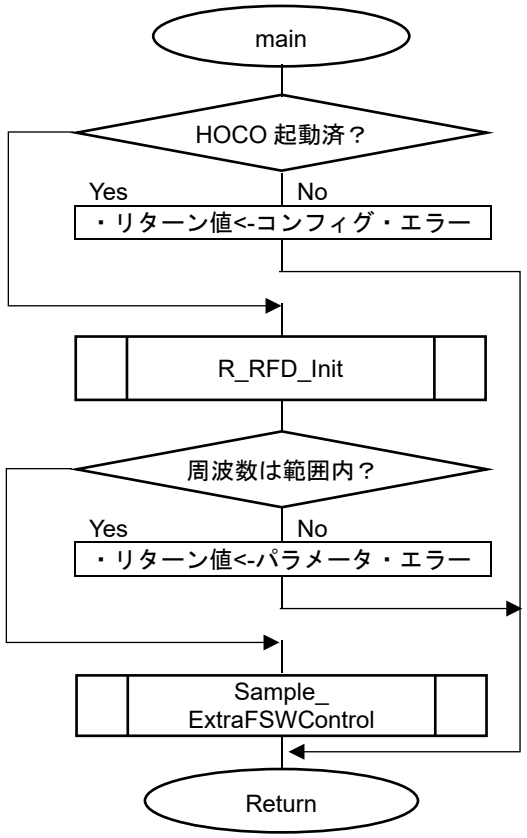
・CPU 動作周波数: 40MHz（メイン・システム・クロックに高速オンチップ・オシレータ・クロックを使用）

・書き込み領域: エクストラ領域（FSW 関連データ）

・書き込みデータ・サイズ: 4byte

RFD RL78 Type02 のエクストラ領域書き換え制御サンプルのメイン処理実行フローを図 5-10 に示します。

5.3.3.1 main 関数



・RFD RL78 Type02 を初期化（動作周波数設定）します。

・RL78/F24 の CPU 動作周波数範囲[2~40MHz]
・範囲内は正常(0x00)、範囲外はパラメータ・エラー(0x10)を返す。

・エクストラ領域書き換え制御処理
・リターン値<- 関数のリターン値

図 5-10 エクストラ領域書き換え(FSW)制御サンプルのメイン処理実行フロー

5.3.3.2 Sample_ExtraFSWControl 関数

・コード・フラッシュ・プログラミング・モードへ移行、FSW 設定を実行する。

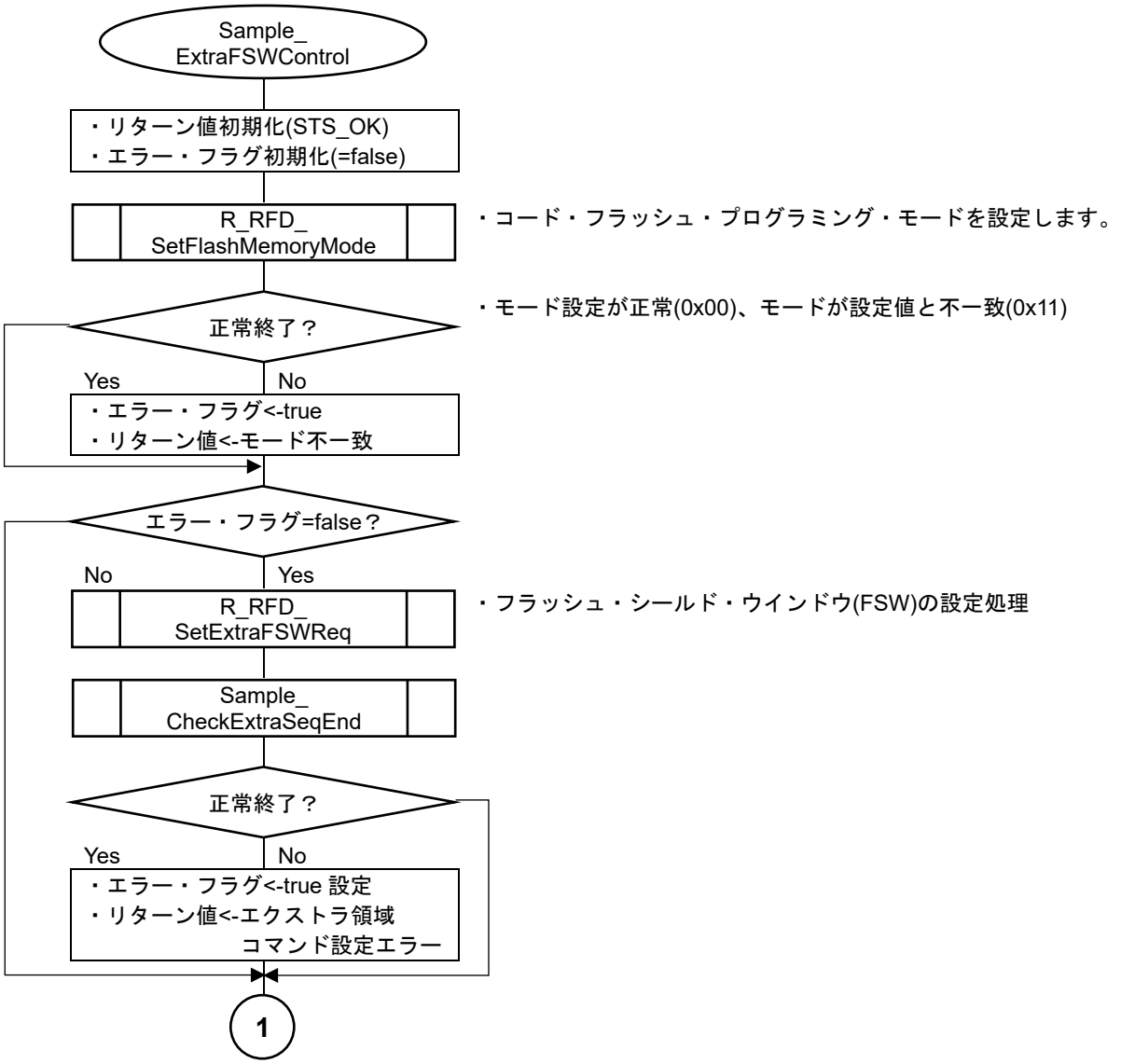
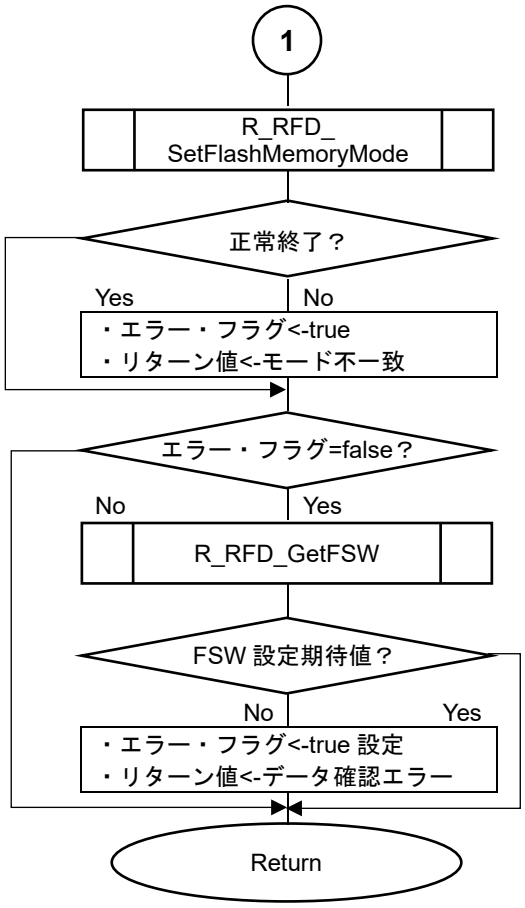


図 5-11 エクストラ領域書き換え(FSW)制御サンプルの処理実行フロー(1/2)

・非書き換えモードへ移行、FSW 設定値を読み出し、期待値通りであるかを確認する。



・非書き換えモードを設定します。

・モード設定が正常(0x00)、モードが設定値と不一致(0x11)

・FSW のスタート・ブロック、エンド・ブロック+1 を読み出します。

図 5-12 エクストラ領域書き換え(FSW)制御サンプルの処理実行フロー(2/2)

5.3.4 共通フラッシュ制御サンプル・プログラム

5.3.4.1 Sample_CheckCFDFSeqEnd 関数

・起動したコード/データ・フラッシュ領域シーケンサの動作終了を確認します。

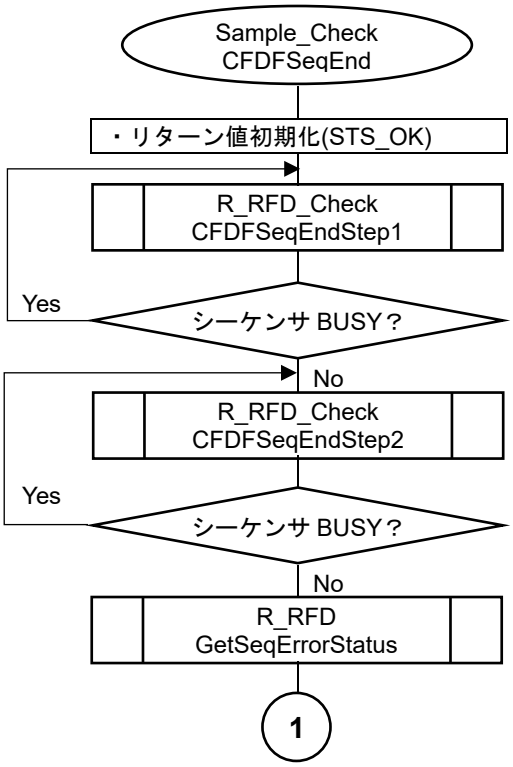


図 5-13 Sample_CheckCFDFSeqEnd 関数の処理実行フロー(1/2)

・実行結果を返します。

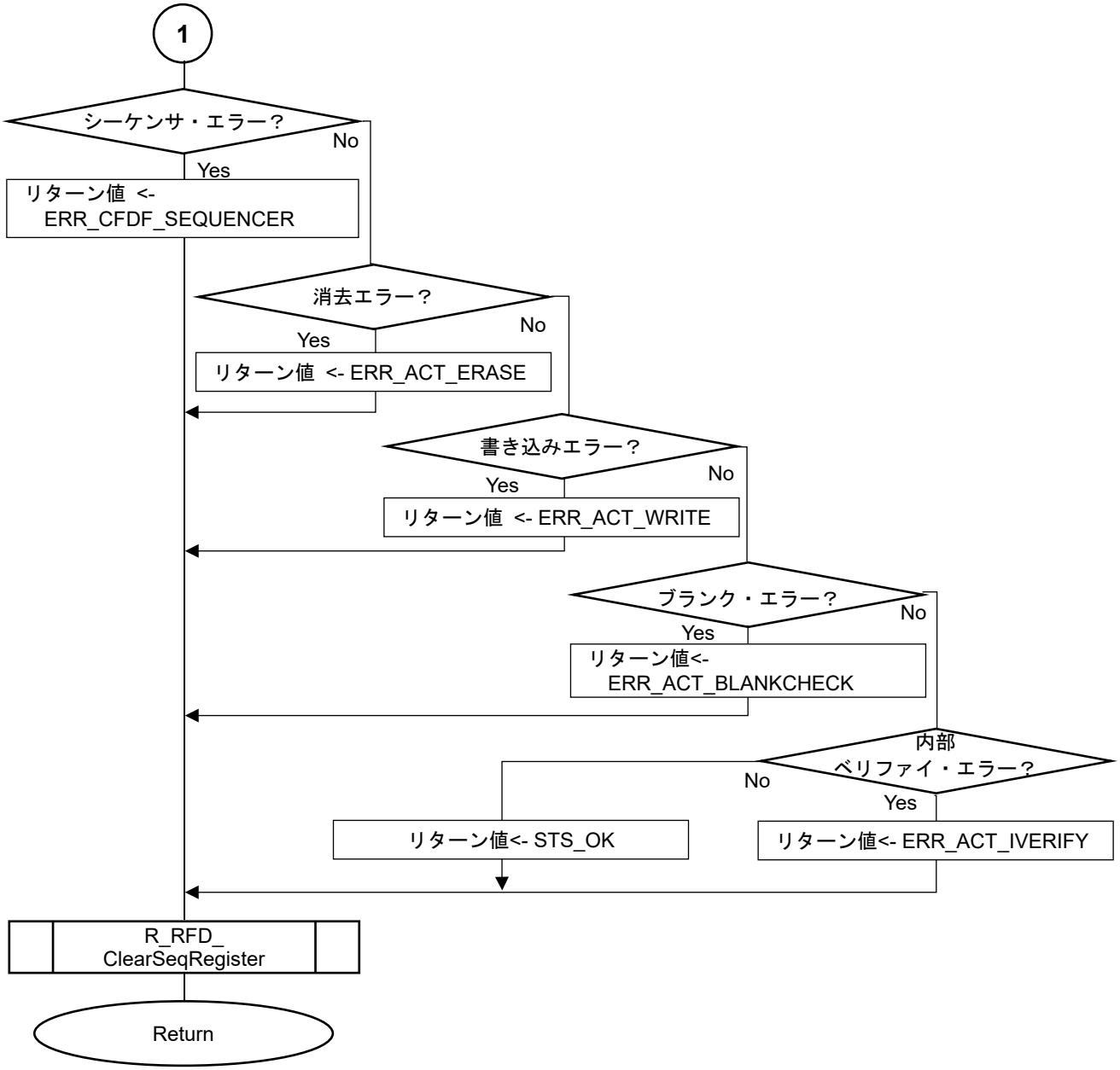


図 5-14 Sample_CheckCFDFSeqEnd 関数の処理実行フロー(2/2)

5.3.4.2 Sample_CheckExtraSeqEnd 関数

・起動したエクストラ領域シーケンサの動作終了を確認し、実行結果を返します。

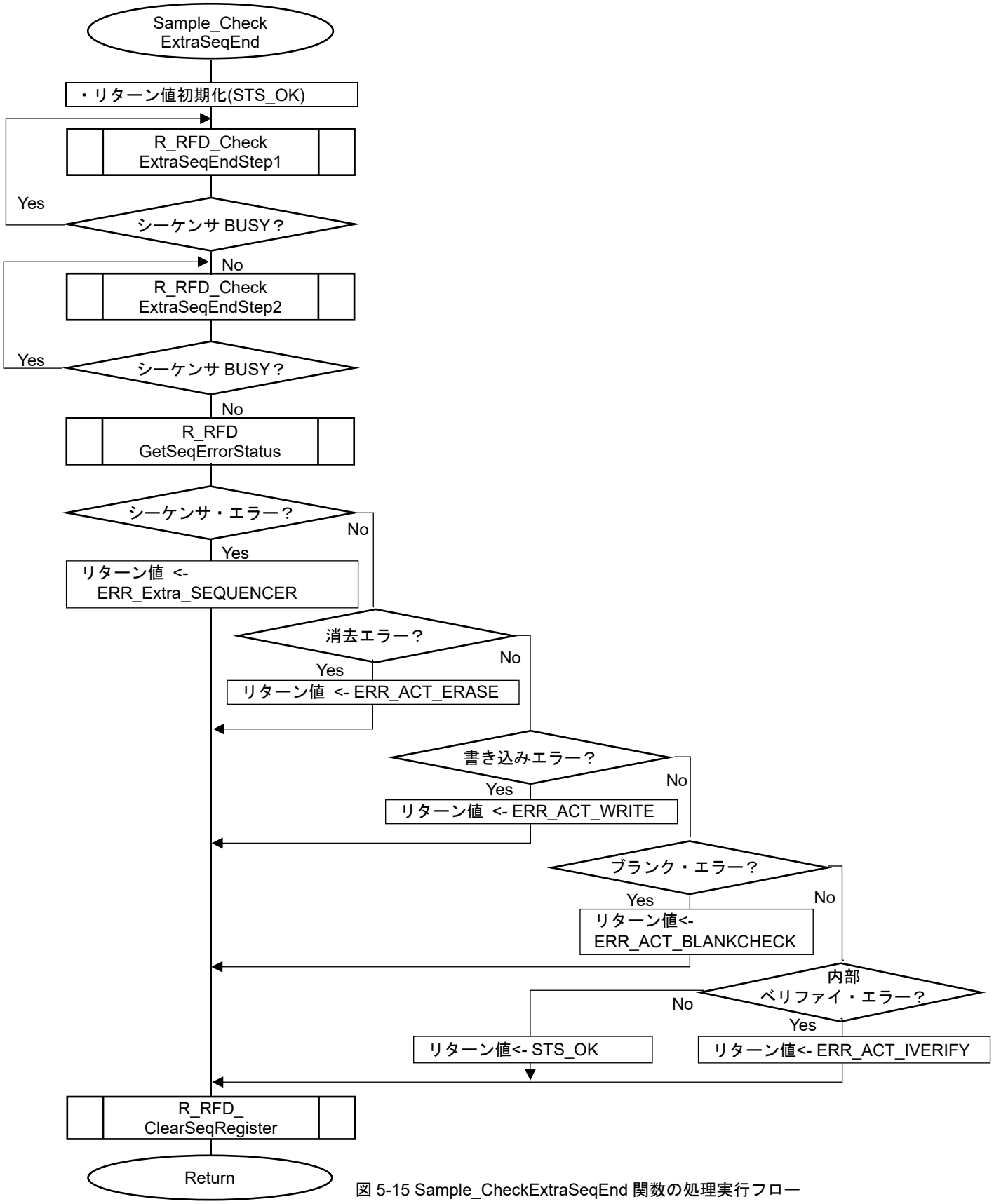


図 5-15 Sample_CheckExtraSeqEnd 関数の処理実行フロー

5.4 サンプル・プログラム関数仕様

この章では、RFD RL78 Type02 のサンプル・プログラム関数仕様について説明します。
RFD RL78 Type02 のサンプル・プログラムは、コード・フラッシュ領域、データ・フラッシュ領域、およびエクストラ領域を書き換える基本的な処理例を示しています。各フラッシュ領域を書き換えるアプリケーションを開発する上で、各サンプル・プログラム関数を参考にしていただくことができます。
開発されたアプリケーション・プログラムについては、お客様自身で必ず十分な動作確認を行ってください。

5.4.1 コード・フラッシュ書き換え制御サンプル・プログラム関数仕様

5.4.1.1 main

Information

Syntax	int main(void);	
Reentrancy	Non-Reentrant	
Parameters (IN)	N/A	
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	int (e_sample_ret_t)	SAMPLE_ENUM_RET_STS_OK : 0x00 [正常終了] SAMPLE_ENUM_RET_ERR_PARAMETER : 0x10 [パラメータ・エラー] SAMPLE_ENUM_RET_ERR_CONFIGURATION : 0x11 [初期設定エラー] SAMPLE_ENUM_RET_ERR_MODE_MISMATCHED : 0x12 [モード不一致エラー] SAMPLE_ENUM_RET_ERR_CHECK_WRITE_DATA : 0x13 [書き込みデータ確認エラー] SAMPLE_ENUM_RET_ERR_CMD_ERASE : 0x30 [消去コマンド・エラー] SAMPLE_ENUM_RET_ERR_CMD_WRITE : 0x31 [書き込みコマンド・エラー] SAMPLE_ENUM_RET_ERR_CMD_BLANKCHECK :0x32 [ブランク・チェック・コマンド・エラー] SAMPLE_ENUM_RET_ERR_CMD_IVERIFY: 0x33 [内部ベリファイ・コマンド・エラー]
Description	コード・フラッシュ書き換え制御サンプル・プログラムのメイン処理	
Preconditions	非書き換えモードで実行してください。 高速オンチップ・オシレータを起動している状態で実行してください。	
Remarks	-	

5.4.1.2 Sample_CodeFlashControl

Information

Syntax	R_RFD_FAR_FUNC e_sample_ret_t Sample_CodeFlashControl (uint32_t i_u32_start_addrr, uint16_t i_u16_write_data_length, uint8_t __near * inp_u08_write_data);	
Reentrancy	Non-Reentrant	
Parameters (IN)	uint32_t i_u32_start_addr	書き換え領域の先頭アドレス
	uint16_t i_u16_write_data_length	書き換えデータ・サイズ
	uint8_t __near * inp_u08_write_data	書き換えデータ・バッファのポインタ
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	e_sample_ret_t	SAMPLE_ENUM_RET_STS_OK : 0x00 [正常終了] SAMPLE_ENUM_RET_ERR_MODE_MISMATCHED : 0x12 [モード不一致エラー] SAMPLE_ENUM_RET_ERR_CHECK_WRITE_DATA : 0x13 [書き込みデータ確認エラー] SAMPLE_ENUM_RET_ERR_CMD_ERASE : 0x30 [消去コマンド・エラー] SAMPLE_ENUM_RET_ERR_CMD_WRITE : 0x31 [書き込みコマンド・エラー] SAMPLE_ENUM_RET_ERR_CMD_BLANKCHECK :0x32 [ブランク・チェック・コマンド・エラー] SAMPLE_ENUM_RET_ERR_CMD_IVERIFY: 0x33 [内部ベリファイ・コマンド・エラー]
Description	コード・フラッシュ・メモリの書き換え処理を実行 - コード・フラッシュ・プログラミング・モードで、ブランク・チェック、消去、書き込み、内部ベリファイの各コマンドを実行します。 - 非書き換えモードにおいて、書き込まれたデータを読み出し、正しく書かれたかを確認します。	
Preconditions	非書き換えモードで実行してください。 高速オンチップ・オシレータを起動している状態で実行してください。	
Remarks	-	

5.4.2 データ・フラッシュ書き換え制御サンプル・プログラム関数仕様

5.4.2.1 main

Information

Syntax	int main(void);	
Reentrancy	Non-Reentrant	
Parameters (IN)	N/A	
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	Int (e_sample_ret_t)	SAMPLE_ENUM_RET_STS_OK : 0x00 [正常終了] SAMPLE_ENUM_RET_ERR_PARAMETER : 0x10 [パラメータ・エラー] SAMPLE_ENUM_RET_ERR_CONFIGURATION : 0x11 [初期設定エラー] SAMPLE_ENUM_RET_ERR_MODE_MISMATCHED : 0x12 [モード不一致エラー] SAMPLE_ENUM_RET_ERR_CHECK_WRITE_DATA : 0x13 [書き込みデータ確認エラー] SAMPLE_ENUM_RET_ERR_CMD_ERASE : 0x30 [消去コマンド・エラー] SAMPLE_ENUM_RET_ERR_CMD_WRITE : 0x31 [書き込みコマンド・エラー] SAMPLE_ENUM_RET_ERR_CMD_BLANKCHECK :0x32 [ブランク・チェック・コマンド・エラー] SAMPLE_ENUM_RET_ERR_CMD_IVERIFY: 0x33 [内部ベリファイ・コマンド・エラー]
Description	データ・フラッシュ書き換え制御サンプル・プログラムのメイン処理	
Preconditions	非書き換えモードで実行してください。 高速オンチップ・オシレータを起動している状態で実行してください。	
Remarks	-	

5.4.2.2 Sample_DataFlashControl

Information

Syntax	R_RFD_FAR_FUNC e_sample_ret_t Sample_DataFlashControl (uint32_t i_u32_start_addrr, uint16_t i_u16_write_data_length, uint8_t __near * inp_u08_write_data);	
Reentrancy	Non-Reentrant	
Parameters (IN)	uint32_t i_u32_start_addr	書き換え領域の先頭アドレス
	uint16_t i_u16_write_data_length	書き換えデータ・サイズ
	uint8_t __near * inp_u08_write_data	書き換えデータ・バッファのポインタ
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	e_sample_ret_t	SAMPLE_ENUM_RET_STS_OK : 0x00 [正常終了] SAMPLE_ENUM_RET_ERR_MODE_MISMATCHED : 0x12 [モード不一致エラー] SAMPLE_ENUM_RET_ERR_CMD_ERASE : 0x30 [消去コマンド・エラー] SAMPLE_ENUM_RET_ERR_CMD_BLANKCHECK :0x32 [ブランク・チェック・コマンド・エラー] SAMPLE_ENUM_RET_ERR_CMD_WRITE : 0x31 [書き込みコマンド・エラー] SAMPLE_ENUM_RET_ERR_CMD_IVERIFY: 0x33 [内部ベリファイ・コマンド・エラー] SAMPLE_ENUM_RET_ERR_CHECK_WRITE_DATA : 0x13 [書き込みデータ確認エラー]
Description	データ・フラッシュ・メモリの書き換え処理を実行 - データ・フラッシュ・プログラミング・モードで、ブランク・チェック、消去、書き込み、内部ベリファイの各コマンドを実行します。 - 非書き換えモードにおいて、書き込まれたデータを読み出し、正しく書かれたかを確認します。	
Preconditions	非書き換えモードで実行してください。 高速オンチップ・オシレータを起動している状態で実行してください。 本関数の最初にデータ・フラッシュのアクセスを許可し、データ・フラッシュの書き換え完了後にデータ・フラッシュのアクセスを禁止します。	
Remarks	-	

5.4.3 エクストラ領域書き換え制御サンプル・プログラム関数仕様

5.4.3.1 main

Information

Syntax	int main(void);	
Reentrancy	Non-Reentrant	
Parameters (IN)	N/A	
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	int (e_sample_ret_t)	SAMPLE_ENUM_RET_STS_OK : 0x00 [正常終了] SAMPLE_ENUM_RET_ERR_PARAMETER : 0x10 [パラメータ・エラー] SAMPLE_ENUM_RET_ERR_CONFIGURATION : 0x11 [初期設定エラー] SAMPLE_ENUM_RET_ERR_MODE_MISMATCHED : 0x12 [モード不一致エラー] SAMPLE_ENUM_RET_ERR_CHECK_WRITE_DATA : 0x13 [書き込みデータ確認エラー] SAMPLE_ENUM_RET_ERR_CMD_SET_EXTRA_AREA : 0x34 [エクストラ領域コマンド設定エラー]
Description	エクストラ領域(FSW 機能)書き換え制御サンプル・プログラムのメイン処理	
Preconditions	非書き換えモードで実行してください。 高速オンチップ・オシレータを起動している状態で実行してください。	
Remarks	-	

5.4.3.2 Sample_ExtraFSWControl

Information

Syntax	R_RFD_FAR_FUNC e_sample_ret_t Sample_ExtraFSWControl (uint16_t i_u16_start_block_number, uint16_t i_u16_end_block_number);	
Reentrancy	Non-Reentrant	
Parameters (IN)	uint16_t i_u16_start_block_numberr	スタート・ブロック番号 例:RL78/F24 では、Max. 256Kbyte[0~255]
	uint16_t i_u16_end_block_number	エンド・ブロック番号+1 例:RL78/F24 では、Max. 256Kbyte[1~256]
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	e_sample_ret_t	SAMPLE_ENUM_RET_STS_OK : 0x00 [正常終了] SAMPLE_ENUM_RET_ERR_MODE_MISMATCHED : 0x12 [モード不一致エラー] SAMPLE_ENUM_RET_ERR_CHECK_WRITE_DATA : 0x13 [書き込みデータ確認エラー] SAMPLE_ENUM_RET_ERR_CMD_SET_EXTRA_AREA : 0x34 [エクストラ領域コマンド設定エラー]
Description	エクストラ領域(FSW 機能)の書き換え処理を実行 - コード・フラッシュ・プログラミング・モードで、エクストラ領域書き込み(FSW のプログラミング)コマンドを実行します。 - 非書き換えモードにおいて、書き込まれたデータに該当する内蔵レジスタを読み出し、正しく書かれたかをチェックします。	
Preconditions	非書き換えモードで実行してください。 高速オンチップ・オシレータを起動している状態で実行してください。	
Remarks	-	

5.4.4 共通サンプル・プログラムの関数仕様

5.4.4.1 Sample_CheckCFDFSeqEnd

Information

Syntax	R_RFD_FAR_FUNC e_sample_ret_t Sample_CheckCFDFSeqEnd(void);	
Reentrancy	Non-Reentrant	
Parameters (IN)	N/A	
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	e_sample_ret_t	SAMPLE_ENUM_RET_STS_OK : 0x00 [正常終了] SAMPLE_ENUM_RET_ERR_CFDF_SEQUENCER : 0x20 [コード/データ・フラッシュ領域シーケンサ・エラー] SAMPLE_ENUM_RET_ERR_ACT_ERASE : 0x22 [消去動作エラー] SAMPLE_ENUM_RET_ERR_ACT_WRITE : 0x23 [書き込み動作エラー] SAMPLE_ENUM_RET_ERR_ACT_BLANKCHECK : 0x24 [ブランク・チェック動作エラー] SAMPLE_ENUM_RET_ERR_ACT_IVERIFY : 0x25 [内部ベリファイ動作エラー]
Description	コード/データ・フラッシュ領域シーケンサ・コマンドの完了待ち処理	
Preconditions	コード・フラッシュ・プログラミング・モード、または、データ・フラッシュ・プログラミング・モードで使用してください。 高速オンチップ・オシレータを起動している状態で実行してください。 データ・フラッシュ書き換え実行時は、データ・フラッシュ・アクセス許可状態で使用してください(DFLEN = 1)。	
Remarks	-	

5.4.4.2 Sample_CheckExtraSeqEnd

Information

Syntax	R_RFD_FAR_FUNC e_sample_ret_t Sample_CheckExtraSeqEnd(void);	
Reentrancy	Non-Reentrant	
Parameters (IN)	N/A	
Parameters (IN/OUT)	N/A	
Parameters (OUT)	N/A	
Return Value	e_sample_ret_t	SAMPLE_ENUM_RET_STS_OK : 0x00 [正常終了] SAMPLE_ENUM_RET_ERR_EXTRA_SEQUENCER : 0x21 [エクストラ・シーケンサ・エラー] SAMPLE_ENUM_RET_ERR_ACT_ERASE : 0x22 [消去動作エラー] SAMPLE_ENUM_RET_ERR_ACT_WRITE : 0x23 [書き込み動作エラー] SAMPLE_ENUM_RET_ERR_ACT_BLANKCHECK : 0x24 [ブランク・チェック動作エラー] SAMPLE_ENUM_RET_ERR_ACT_IVERIFY : 0x25 [内部ベリファイ動作エラー]
Description	エクストラ領域シーケンサ・コマンドの完了待ち処理	
Preconditions	コード・フラッシュ・プログラミング・モードで実行してください。 高速オンチップ・オシレータを起動している状態で実行してください。	
Remarks	-	

6 RFD RL78 Type02 サンプル・プロジェクトの作成

RFD RL78 Type02 は、コード・フラッシュ・メモリ領域とデータ・フラッシュ・メモリ領域の書き換えサンプル・プログラムが含まれます。RFD RL78 Type02 で使用できるコンパイラは、CC-RL コンパイラと IAR コンパイラです。それぞれのコンパイラに対応する統合開発環境を使用してサンプル・プロジェクトを作成することができます。

この項では、RL78/F24(R7F124FPJ)用のサンプル・プログラムを例に説明しています。RL78/F24(R7F124FPJ)以外を使用する場合、セクション設定のアドレスなどを、対象デバイスのユーザーズマニュアルを参照いただき、変更する必要があります

RL78/F23 を使用する場合は、RL78/F24 のサンプル・プログラムを使用できます。

注意　対象の統合開発環境、およびコンパイラは、RL78/F23,F24 を対象としたバージョンをご使用いただくことを前提としています。RL78/F23､F24 が対象製品であることをご確認の上、ご使用ください。

6.1 CC-RL コンパイラを使用する場合のプロジェクトの作成

RENESAS 製 CC-RL コンパイラは、統合開発環境として CS+、および e^{2}studio を使用して作成したプロジェクトへ RFD RL78Type02 を登録し、ビルドすることができます。各統合開発環境を使用した場合のサンプル・プロジェクトの作成例を示します。CC-RL コンパイラ、および各統合開発環境を理解するため、それぞれのツール製品のユーザーズマニュアルを参照してください。

6.1.1 サンプル・プロジェクト作成例

(1) 統合開発環境 CS+を使用したサンプル・プロジェクト作成例

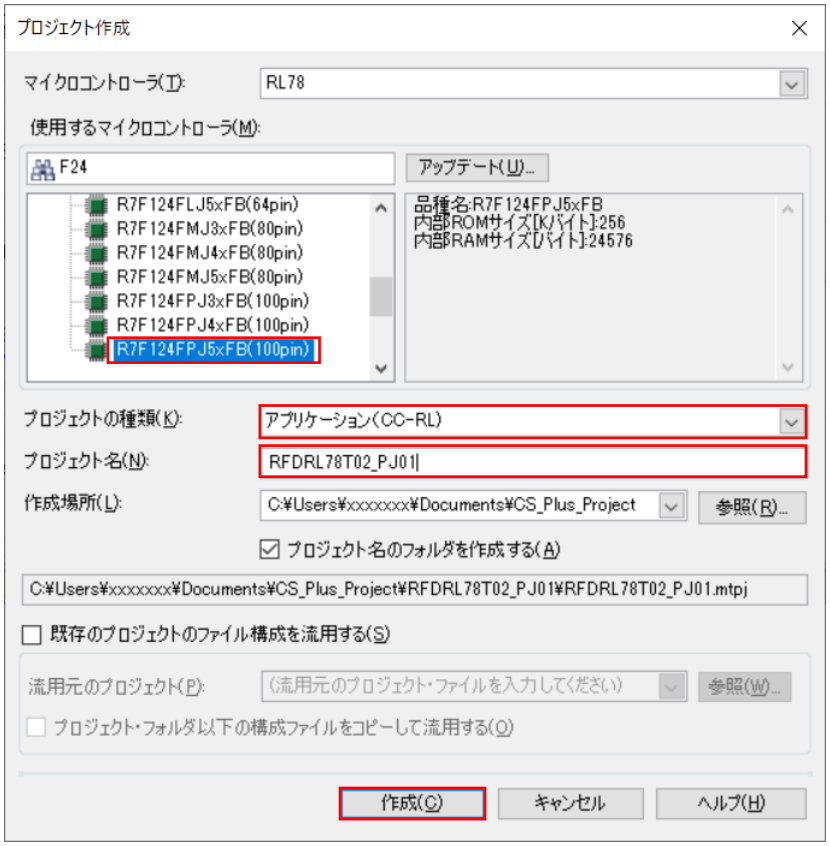
CS+を起動し、[プロジェクト]メニューの[新しいプロジェクトを作成]を選択し、以下に示す ”プロジェクト作成”ウインドウを起動します。

・[使用するマイクロコントローラ]は、”RL78/F24 (ROM: 256KB)” - “R7F124FPJ5xFB (100pin)”を選択します。

・[プロジェクトの種類]は、”アプリケーション(CC-RL)”を選択します。

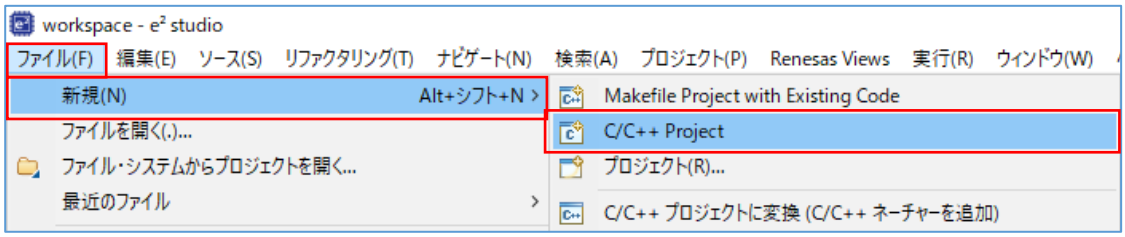
・ここでの[プロジェクト名]は、仮に”RFDRL78T02_PJ01”とします。

・[作成]ボタンを押すと、新しいプロジェクトが作成されます。

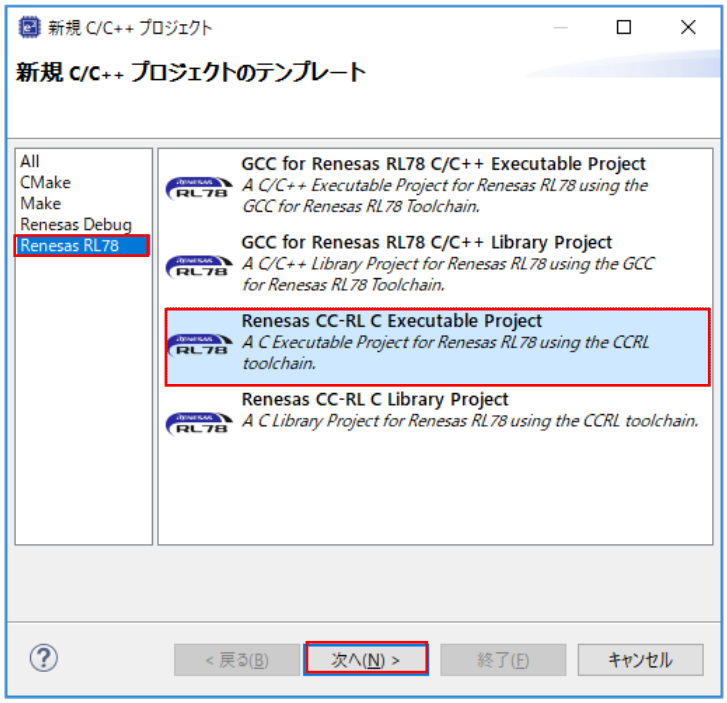


(2) 統合開発環境 e²studio を使用したサンプル・プロジェクト作成例

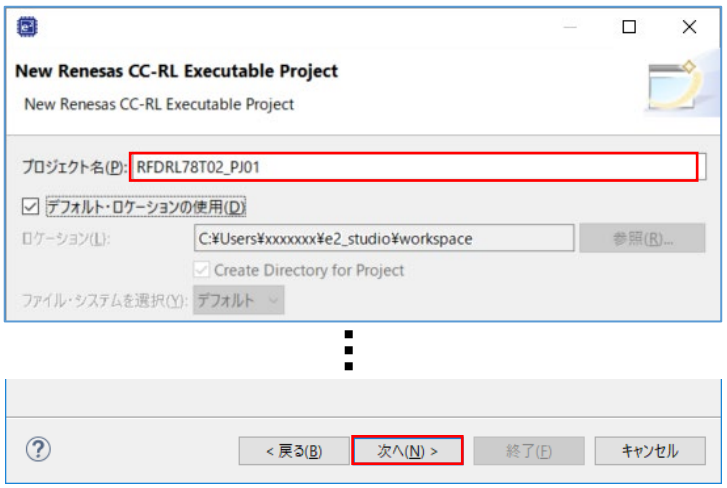
e²studio を起動し、[ファイル]メニューの[新規]から[C/C++ Project]を選択し、”新規 C/C++ プロジェクトのテンプレート”ウインドウを起動します。



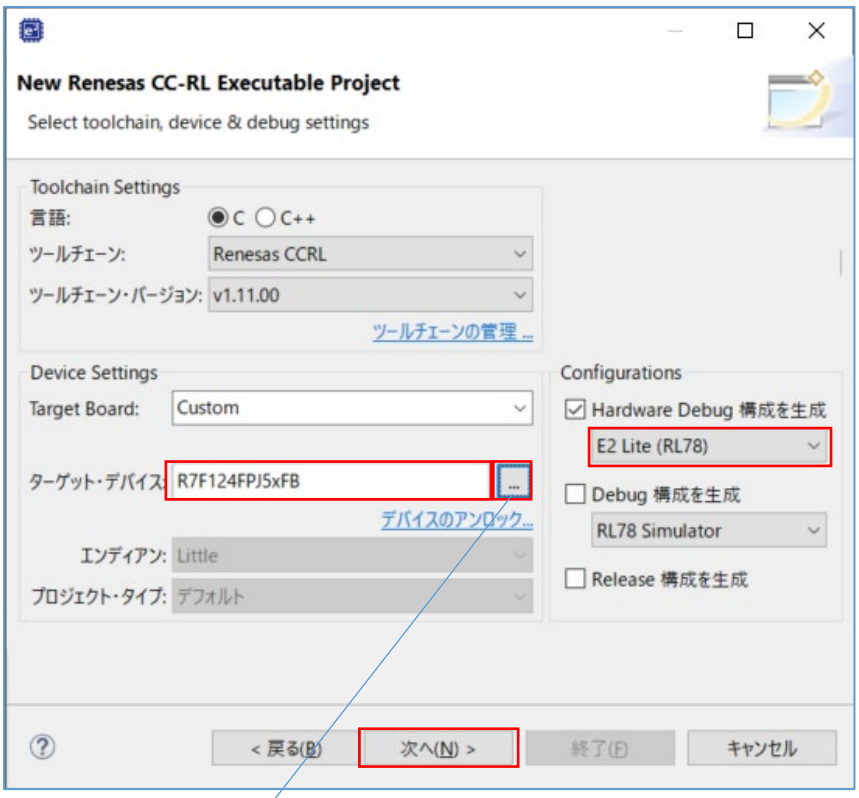
・[Renesas RL78]を選択して表示した[Renesas CC-RL C Executable Project]を選択、”次へ”ボタンを押します。

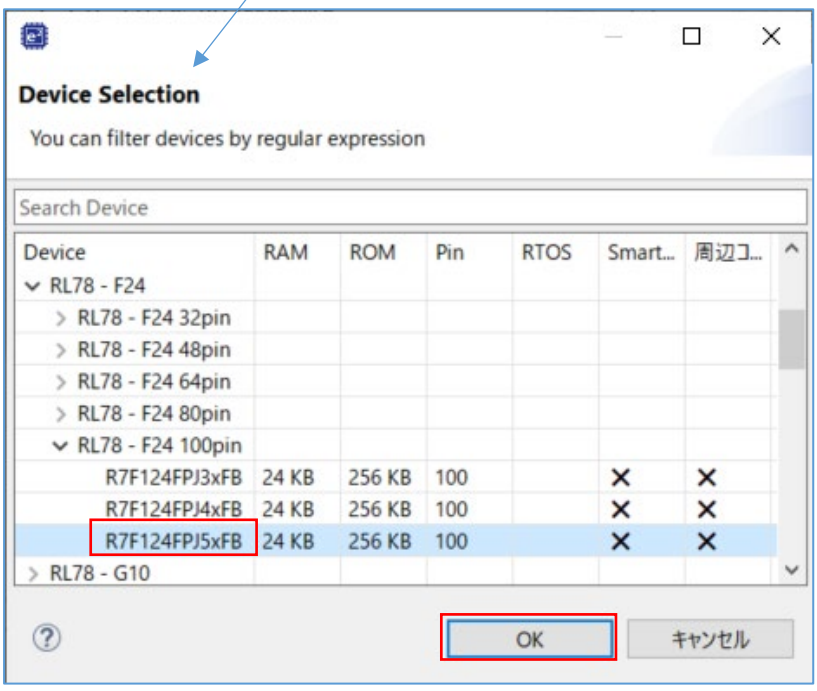


・”New Renesas CC-RL Executable Project”ウインドウで、プロジェクト名を入力して”次へ”ボタンを押します。（ここでは、仮に”RFDRL78T02_PJ01”とします。）



・[Device Settings]の[ターゲット・デバイスから、”RL78 - F24 100 pin” - “R7F124FPJ5xFB”を選択します。
・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。
[Configurations]で”Hardware Debug 構成を生成”にチェックが入った状態で、E2 Lite (RL78)を選択します。
・[次へ]ボタンを押すと”Select Coding Assistant settings”ウインドウが表示されるので、[終了]ボタンを押します。

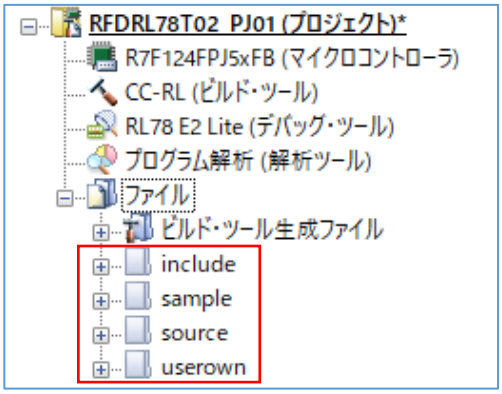




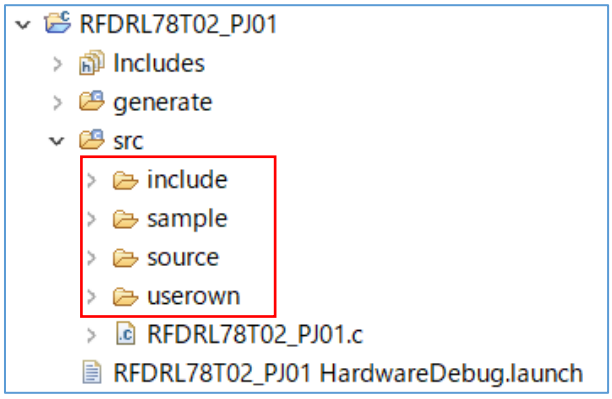
6.1.2 対象フォルダと対象ファイルの登録例

RFD RL78 Type02 を使用して、各領域[(1) コード・フラッシュ・メモリ、(2) データ・フラッシュ・メモリ、(3) エクストラ領域]を書き換える場合に必要なファイルの登録例を記述します。RFD RL78 Type02 ソースプログラムファイルの各フォルダは、”include”, ”source”, ”userown”, ”sample”で、書き換える領域により各フォルダ内の対象ファイルを選択して登録します。

その他の手順として、”include”, ”source”, ”userown”, ”sample”の全てのフォルダを登録し、不要なファイルとフォルダを、[プロジェクトから外す] 機能(CS+)、[リソース構成] – [ビルドから除外…]機能により、対象から外すこともできます。



CS+の RFD RL78 Type02 登録時のツリー画面



e²studio RFD RL78 Type02 登録時のツリー画面

・統合開発環境で対象製品用に出力された最新の I/O ヘッダ・ファイルの登録

”iodefine.h”は、CS+、または e²studio が対象製品用に出力する I/O ヘッダ・ファイルです。RFD RL78 Type02 に含まれている”iodefine.h”の代わりに置き換えてご使用頂くことを推奨いたします。"対象フォルダと対象ファイルの登録"を実施した後、統合開発環境が出力した”iodefine.h”を RFD RL78 Type02 の書き換え対象エリアごとのフォルダ内の”iodefine.h”と入れ替える、もしくは上書きしてください。

統合開発環境が I/O ヘッダ・ファイル”iodefine.h”を出力するフォルダ：

- CS+：[プロジェクト名]フォルダ
- e²studio：[プロジェクト名]/generate フォルダ

”iodefine.h”ファイルを入れ替え、もしくは上書きするフォルダ：

- コード・フラッシュ書き換え時：”\[プロジェクト名]\sample\RL78_F24\CF\CCRL\include”
- データ・フラッシュ書き換え時：”\[プロジェクト名]\sample\RL78_F24\DF\CCRL\include”
- エクストラ領域(FSW)書き換え時：”\[プロジェクト名]\sample\RL78_F24\EX_FSW\CCRL\include”

・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

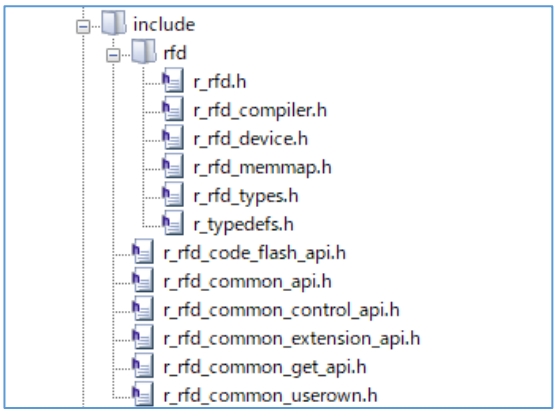
作成されたプロジェクトには、自動的に追加されるファイルがあります。これらと同様のファイルは、RFD RL78 Type02 の”sample”フォルダ内にも存在するため、ツリーで各ファイルを選択し、各統合開発環境の機能を使用して、プロジェクトから外します。

- CS+ではツリーでファイルをマウス右クリック、”プロジェクトから外す”機能で対象ファイルを除外します。
  [プロジェクト名]フォルダ内の cstart.asm, f24opt.asm, hdwinit.asm, stkinit.asm, main.c, iodefine.h が対象。
- e²studio ではツリーで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（対象フォルダ）を除外します。（フォルダから削除も可能）
  [プロジェクト名]/generate フォルダ内の cstart.asm, f24opt.asm, hdwinit.asm, iodefine.h, stkinit.asm,および
  [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”RFDRL78T02_PJ01.c”）が対象。

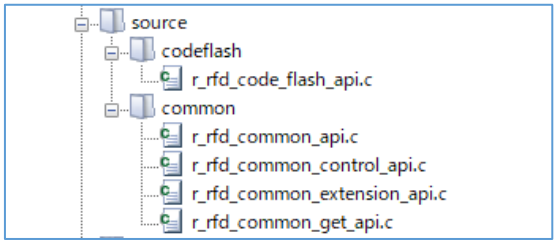
(1) コード・フラッシュ・メモリを書き換える場合の対象フォルダと対象ファイルの登録

RFD RL78 Type02 ソースプログラムファイルの各フォルダ(”include”, ”source”, ”userown”, ”sample”)と登録ファイルを以下に示します。

include フォルダ内

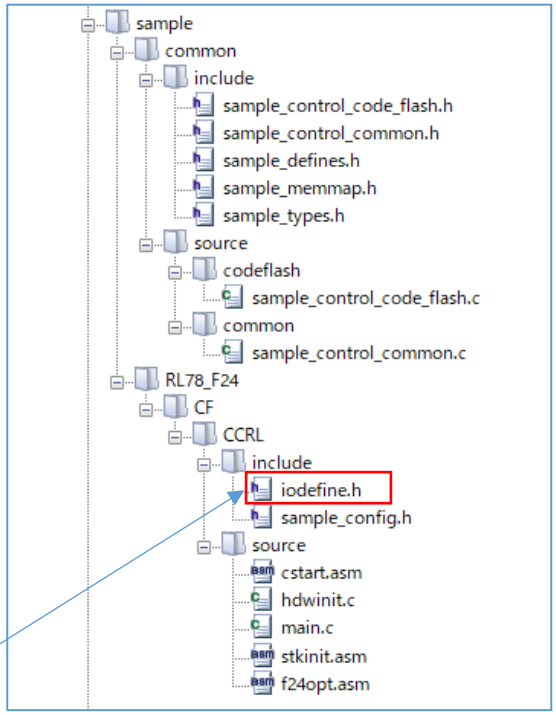


source フォルダ内



userown フォルダ内

sample フォルダ内

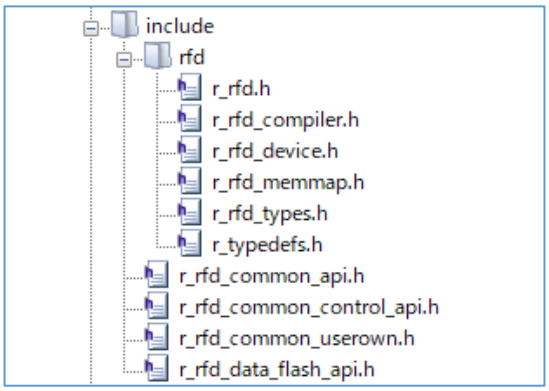


CS+、または e^{2}studio で出力した”iodefine.h” に置き換えてください。

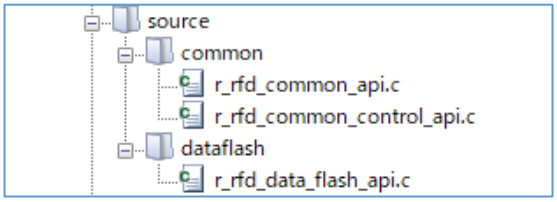
(2) データ・フラッシュ・メモリを書き換える場合の対象フォルダと対象ファイルの登録

RFD RL78 Type02 ソースプログラムファイルの各フォルダ(”include”, ”source”, ”userown”, ”sample”)と登録ファイルを以下に示します。

include フォルダ内

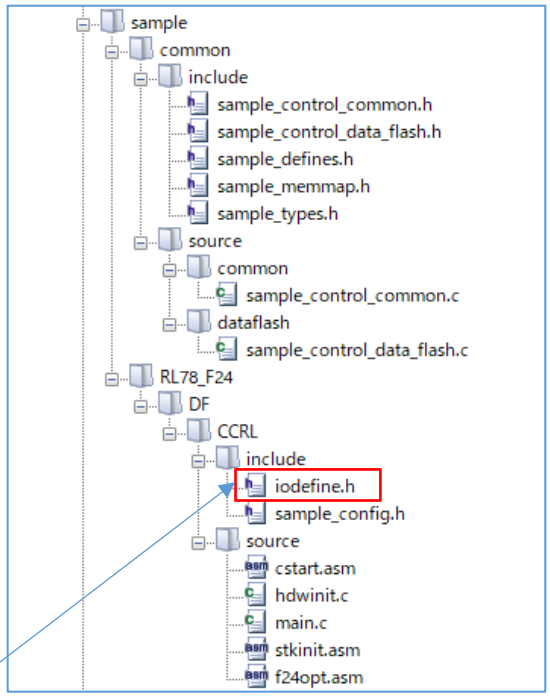


source フォルダ内



userown フォルダ内

sample フォルダ内

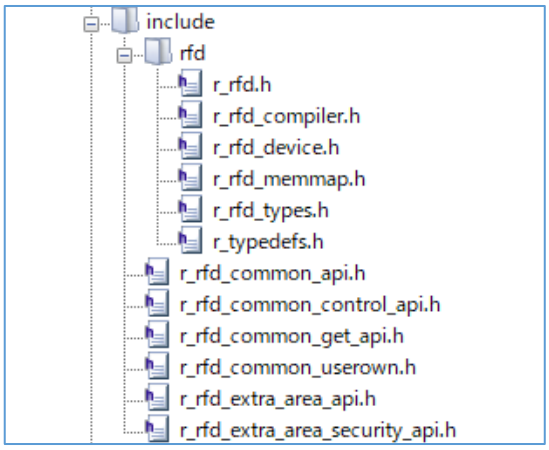


CS+、または e^{2}studio で出力した”iodefine.h” に置き換えてください。

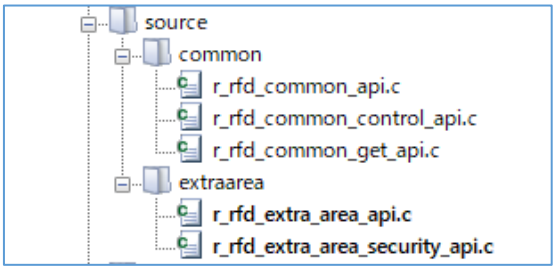
(3) エクストラ領域(FSW 設定)を書き換える場合の対象フォルダと対象ファイルの登録

RFD RL78 Type02 ソースプログラムファイルの各フォルダ(”include”, ”source”, ”userown”, ”sample”)と登録ファイルを以下に示します。

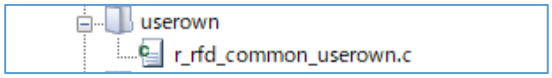
include フォルダ内



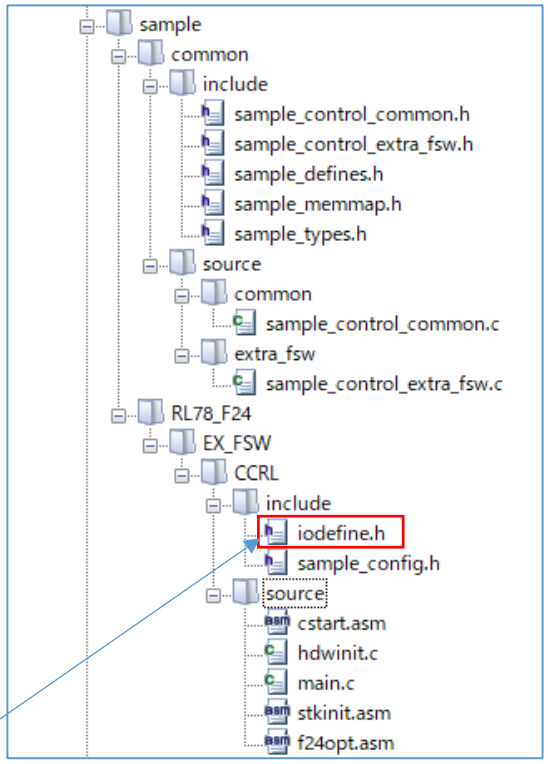
source フォルダ内



userown フォルダ内



sample フォルダ内



CS+、または e²studio で出力した”iodefine.h” に置き換えてください。

6.1.3　ビルド・ツールの設定

CC-RL コンパイラで RFD RL78 Type02 をビルドして実行するための各統合開発環境の設定を行います。

CS+ではツリーで”CC-RL（ビルド・ツール）“のマウス右クリックで”プロパティ”を選択、e²studio ではツリーでプロジェクト（ここでは”RFDRL78T02_PJ01“）のマウス右クリックで”プロパティ”を選択することにより、表示された画面内のビルド・ツールの各設定を行います。

6.1.3.1　インクルード・パスの設定

・CS+でのインクルード・パスの設定は、 “共通オプション”タブで設定（対象領域により変更）

- [よく使うオプション（コンパイル）] – [追加のインクルード・パス]で”パス編集”ウインドウを表示して、インクルード・ファイルのパスを設定します。

(1)コード・フラッシュ・メモリ書き換え

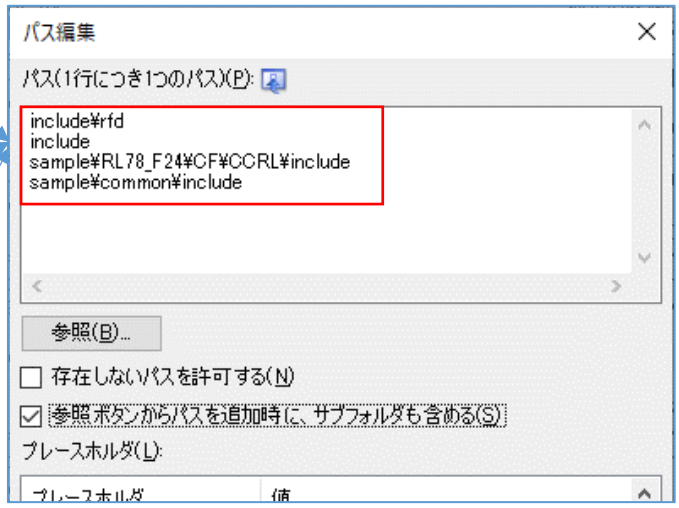
include\rfd
include
sample\RL78_F24\CF\CCRL\include
sample\common\include

(2)データ・フラッシュ・メモリ書き換え

include\rfd
include
sample\RL78_F24\DF\CCRL\include
sample\common\include

(3)エクストラ領域(FSW)書き換え

include\rfd
include
sample\RL78_F24\EX_FSW\CCRL\include
sample\common\include



・e²studio でのインクルード・パスの設定は、”プロパティ”ウインドウで設定（対象領域により変更）

- ”C/C++ビルド” [設定] – “Compiler” [ソース] で表示した画面でインクルード・ファイルのパスを設定します。

(1)コード・フラッシュ・メモリ書き換え

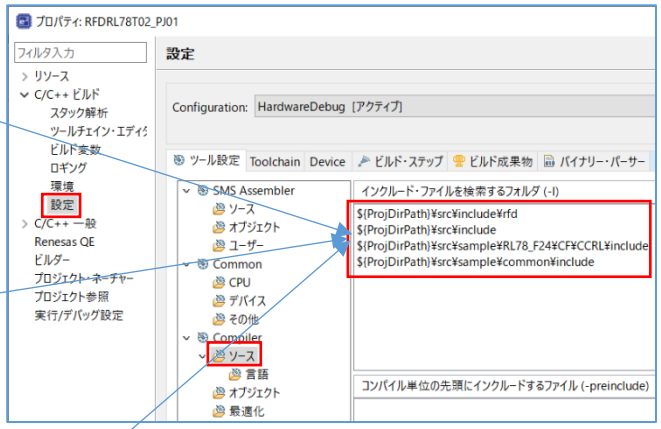
${ProjDirPath}\src\include\rfd
${ProjDirPath}\src\include
${ProjDirPath}\src\sample\RL78_F24\CF\CCRL\include
${ProjDirPath}\src\sample\common\include

(2)データ・フラッシュ・メモリ書き換え

${ProjDirPath}\src\include\rfd
${ProjDirPath}\src\include
${ProjDirPath}\src\sample\RL78_F24\DF\CCRL\include
${ProjDirPath}\src\sample\common\include

(3)エクストラ領域(FSW)書き換え

${ProjDirPath}\src\include\rfd
${ProjDirPath}\src\include
${ProjDirPath}\src\sample\RL78_F24\EX_FSW\CCRL\include
${ProjDirPath}\src\sample\common\include



6.1.3.2 デバイス項目の設定

・CS+でのデバイス項目の設定は、“リンク・オプション”タブで設定（各領域共通）

- [デバイス] 項目を設定します。

[オンチップ・デバッグの許可／禁止をリンク・オプション設定する]を”はい(-OCDBG)”に設定します。

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。

[オンチップ・デバッグ・オプション・バイト制御値]を”A5”に設定します。（オンチップ・デバッグ動作許可の例）

[セキュリティ・オプション・バイトを設定する]を"はい(-SECURITY_OPT_BYTE)"に設定します。

[セキュリティ・オプション・バイト制御値]を"FE"に設定します。（オンチップ・デバッグおよびフラッシュ・シリアル・プログラミング・セキュリティ ID の読み出し許可の例）([RL78/F24 の例])

注）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「オンチップ・デバッグ・オプション・バイト」、「セキュリティ・オプション・バイト」の内容をご確認いただき、使用する設定値を書き込んでください。

[デバッグ・モニタ領域を設定する]を”はい（範囲指定）(-OCDBG_MONITOR=<アドレス範囲>)”に設定します。

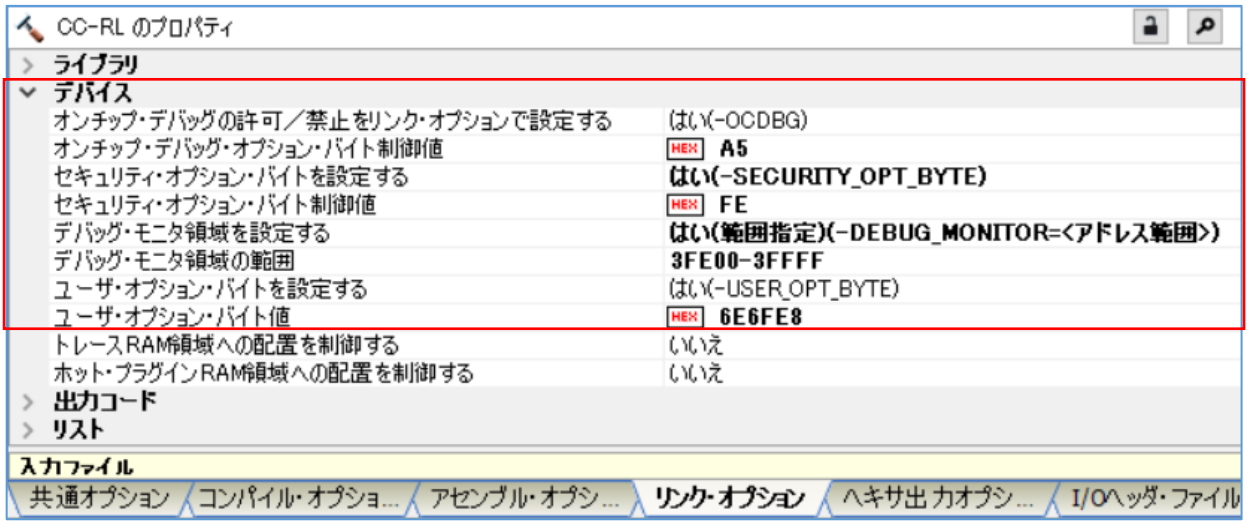
[デバッグ・モニタ領域の範囲]を”3FE00-3FFFF”に設定します。([RL78/F24 の例])

注）対象デバイスのユーザーズマニュアルで「オンチップ・デバッグ機能」の章の「ユーザ資源の確保」で「デバッグ用モニタ・プログラムが配置されるメモリ空間」をご確認いただき、使用する領域の範囲を書き込んでください。

[ユーザ・オプション・バイトを設定する]を”はい(-USER_OPT_BYTE)”に設定します。

[ユーザ・オプション・バイト値]を”6E6FE8”に設定します。（WDT 停止、LVD（reset モード）、40MHz [RL78/F24 の例])

注）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」の内容をご確認いただき、使用する設定値を書き込んでください。



・e²studio でのデバイス項目の設定は、”プロパティ”ウインドウで設定（各領域共通）

- ”C/C++ビルド” [設定] – “Linker” [デバイス] で表示した画面でデバイス項目を設定します。

[OCD モニタのメモリ領域を確保する(-debug_monitor)]をチェックします。

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。

[メモリ領域(-debug_monitor=<start address>-<end address>)]を”3FE00-3FFFF”に設定します。（[RL78/F24 の例]）

注）対象デバイスのユーザーズマニュアルで「オンチップ・デバッグ機能」の章の「デバッグ用モニタ・プログラムが配置されるメモリ空間」をご確認いただき、使用する領域の範囲を書き込んでください。

[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)] をチェックします。

[ユーザ・オプション・バイト値(-user_opt_byte=<value>)]を”6E6FE8”に設定します。（WDT 停止、LVD（reset モード）、40MHz [RL78/F24 の例]）

注）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」の内容をご確認いただき、使用する設定値を書き込んでください。

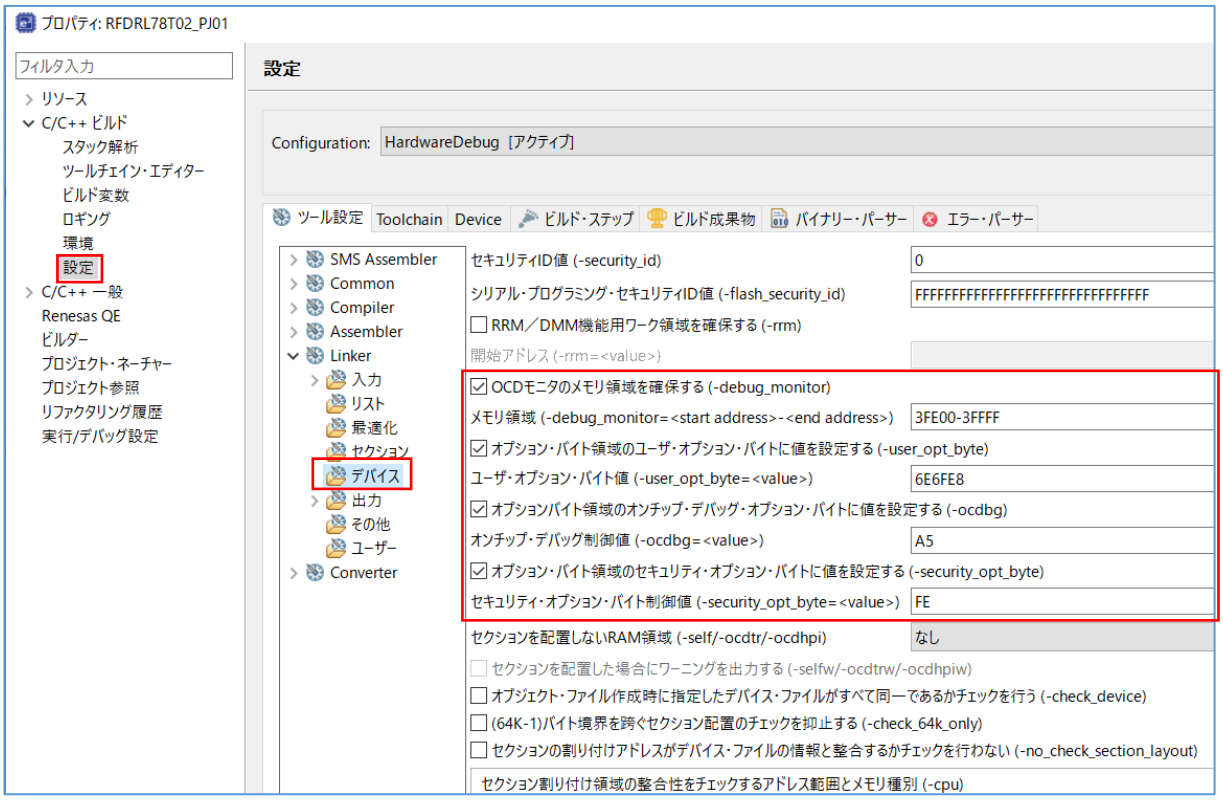
[オプション・バイト領域のオンチップ・デバッグ・オプション・バイトに値を設定する(-ocdbg)]をチェックします。注）オンチップ・デバッグを実施することを前提とした設定例です。

[オンチップ・デバッグ制御値(-ocdbg=<value>)]を”A5”に設定します。（オンチップ・デバッグ動作許可の例）

[オプション・バイト領域のセキュリティ・オプション・バイトに値を設定する (-security_opt_byte)]をチェックします。

[セキュリティ・オプション・バイト制御値 (-security_opt_byte=<value>)]を"FE"に設定します。（オンチップ・デバッグおよびフラッシュ・シリアル・プログラミング・セキュリティ ID の読み出し許可の例）([RL78/F24 の例])

注）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「オンチップ・デバッグ・オプション・バイト」、「セキュリティ・オプション・バイト」の内容をご確認いただき、使用する設定値を書き込んでください。

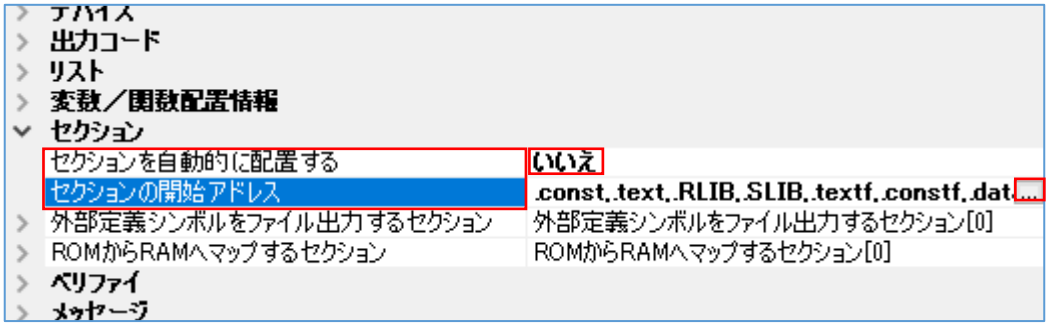


6.1.3.3 セクション項目の設定

・CS+でのセクション項目の設定は、“リンク・オプション”タブで設定（各領域共通）

- [セクション] 項目を設定します。

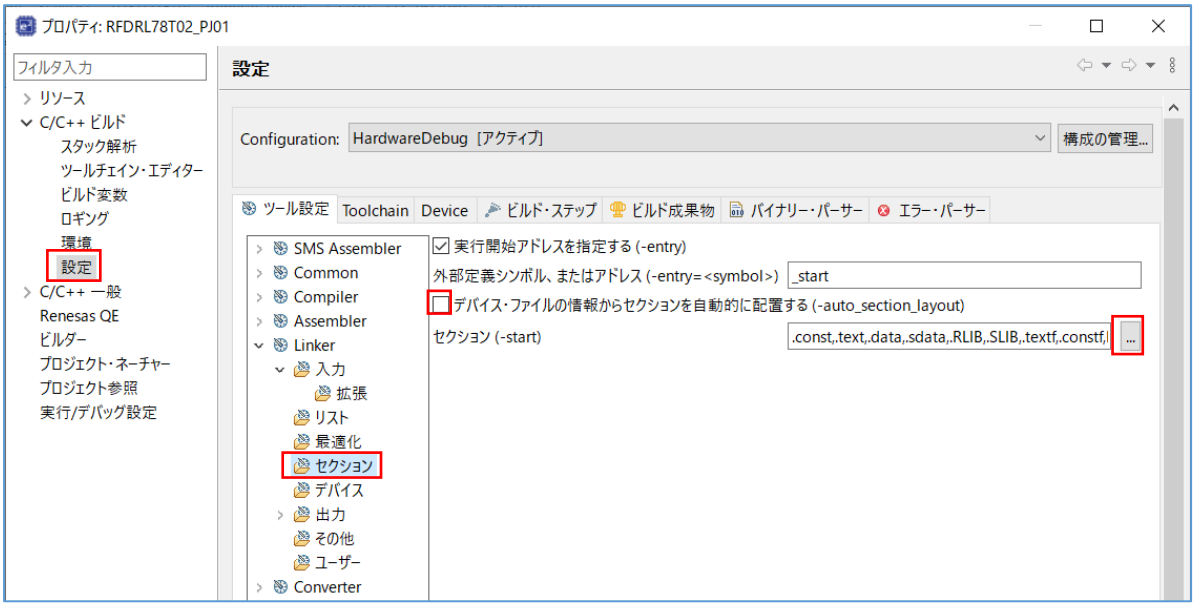
[セクションを自動的に配置する]を一度”いいえ”に設定すると[セクションの開始アドレス]にセクションが表示されるようになり、表示された最も右の” ... ”ボタンで、”セクション設定”画面を表示します。



・e²studio 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(各領域共通)

- ”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ます。ここで、[セクション(-start)]の最も右の” ... ”ボタンで、”セクション設定”画面を表示します。



・CS+、e^{2}studio のセクション設定操作

先頭アドレスに”0x05000”を設定します。

プログラム領域（コード・フラッシュ・メモリ）と RAM 領域に、RFD RL78 Type02 内の”#pragma section”で定義されたセクションを追加します。各セクションの詳細は「2.3.1 RFD RL78 Type02 使用時のセクション」を参照してください。

※本説明では、[コンパイル・オプション]の[メモリ・モデル]がミディアム・モデル（R7F124FPJ での”自動選択”と同じ）であることを前提としています。CC-RL の#pragma section における各プログラムのセクション名は、”__far”の属性で”セクション名+_f”となり、ROM から RAM へマップするセクション名は、”セクション名+_fR”となります。ROM から RAM へマップするセクションのコピー処理は、cstart.asm ファイル内で実施しています。また、”スモール・モデル”を選択した場合の各プログラムのセクション名については CC-RL のユーザーズマニュアルを参照してください。

(1) コード・フラッシュ・メモリ書き換えに必要なセクションの追加

・CS+でのコード・フラッシュ・メモリ書き換えに必要なセクション追加

コード・フラッシュ・メモリ書き換えに必要なセクションを“セクション設定”画面で追加します。

プログラム領域へ追加：RFD_DATA_n, RFD_CMN_f, RFD_CF_f, SMP_CMN_f, SMP_CF_f

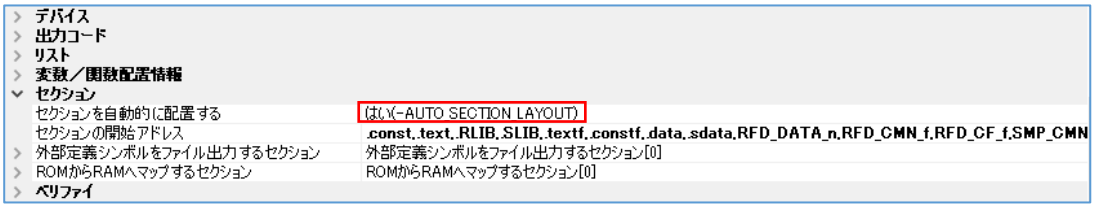
RAM へ追加：.stack_bss, RFD_DATA_nR, RFD_CMN_fR, RFD_CF_fR, SMP_CMN_fR, SMP_CF_fR

追加セクション
RFD_DATA_n
RFD_CMN_f
RFD_CF_f
SMP_CMN_f
SMP_CF_f

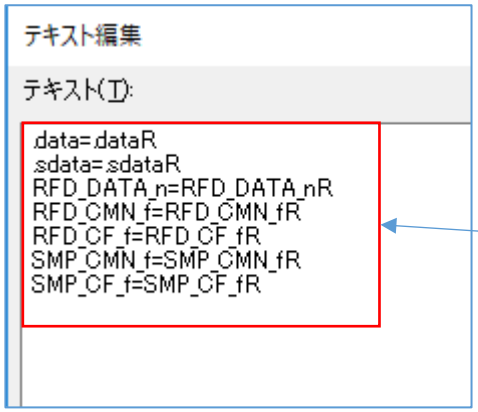
.stack_bss

RFD_DATA_nR
RFD_CMN_fR
RFD_CF_fR
SMP_CMN_fR
SMP_CF_fR

“OK”ボタン押下後、[セクションを自動的に配置する]を”はい”に戻して下さい。



[ROM から RAM へマップするセクション]の最も右の” ... ”ボタンで、”テキスト編集”画面を表示して、ROM から RAM へマップするセクションを追加します。



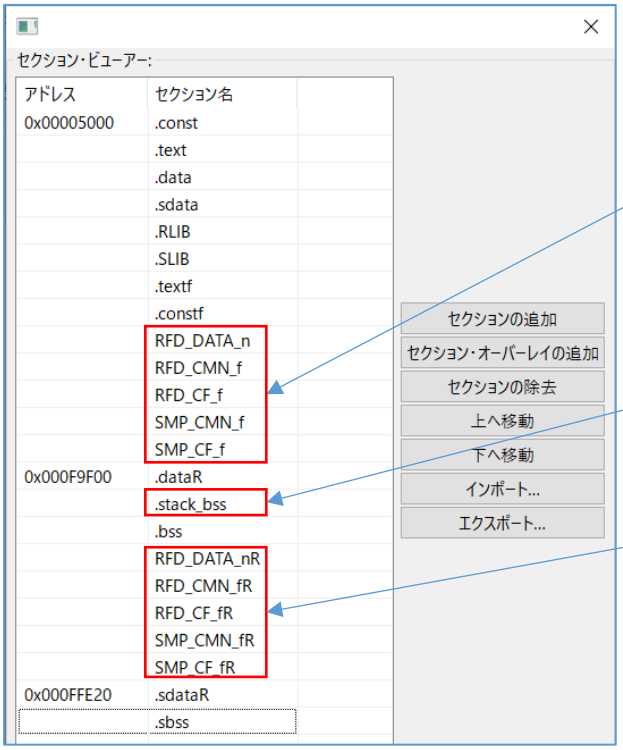
ROM から RAM へ マップするセクション
.data=.dataR
.sdata=.sdataR
RFD_DATA_n=RFD_DATA_nR
RFD_CMN_f=RFD_CMN_fR
RFD_CF_f=RFD_CF_fR
SMP_CMN_f=SMP_CMN_fR
SMP_CF_f=SMP_CF_fR

・e²studio でのコード・フラッシュ・メモリ書き換えに必要なセクション追加

コード・フラッシュ・メモリ書き換えに必要なセクションを“セクション・ビューアー”で追加します。

プログラム領域へ追加：RFD_DATA_n, RFD_CMN_f, RFD_CF_f, SMP_CMN_f, SMP_CF_f

RAM へ追加：.stack_bss, RFD_DATA_nR, RFD_CMN_fR, RFD_CF_fR, SMP_CMN_fR, SMP_CF_fR

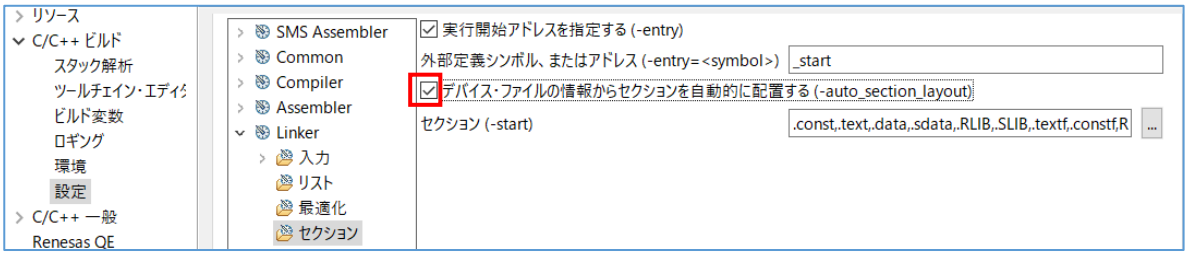


追加セクション
RFD_DATA_n
RFD_CMN_f
RFD_CF_f
SMP_CMN_f
SMP_CF_f

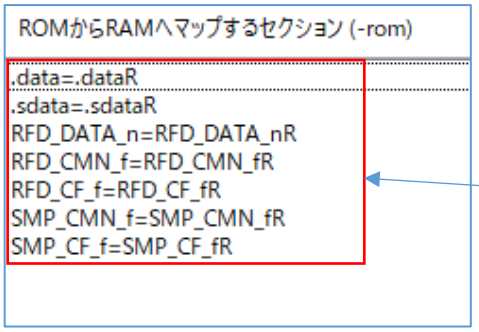
.stack_bss

RFD_DATA_nR
RFD_CMN_fR
RFD_CF_fR
SMP_CMN_fR
SMP_CF_fR

“OK”ボタン押下後、[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)]をチェックして下さい。



”C/C++ビルド” [設定] – “Linker” [出力] で表示した画面で[ROM から RAM へマップするセクション(-rom)]を設定します。



ROM から RAM へ マップするセクション
.data=.dataR
.sdata=.sdataR
RFD_DATA_n=RFD_DATA_nR
RFD_CMN_f=RFD_CMN_fR
RFD_CF_f=RFD_CF_fR
SMP_CMN_f=SMP_CMN_fR
SMP_CF_f=SMP_CF_fR

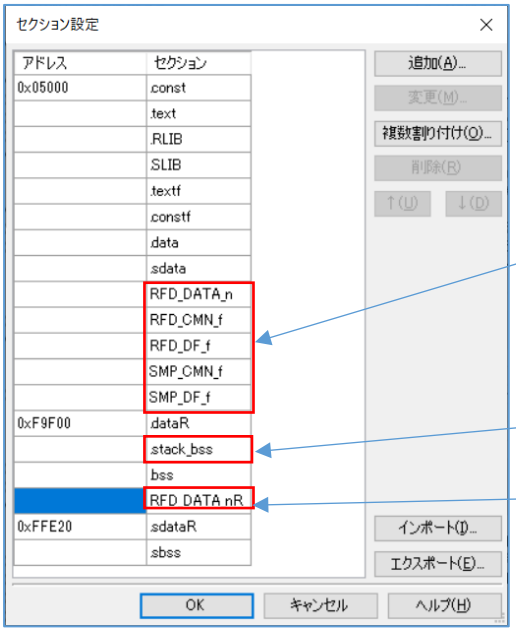
(2) データ・フラッシュ・メモリ書き換えに必要なセクションの追加

・CS+でのデータ・フラッシュ・メモリ書き換えに必要なセクション追加

データ・フラッシュ・メモリ書き換えに必要なセクションを“セクション設定”画面で追加します。

プログラム領域へ追加：RFD_DATA_n, RFD_CMN_f, RFD_DF_f, SMP_CMN_f, SMP_DF_f

RAM へ追加：.stack_bss, RFD_DATA_nR

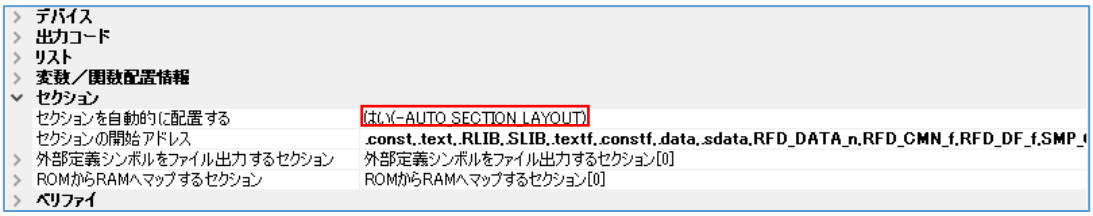


追加セクション
RFD_DATA_n
RFD_CMN_f
RFD_DF_f
SMP_CMN_f
SMP_DF_f

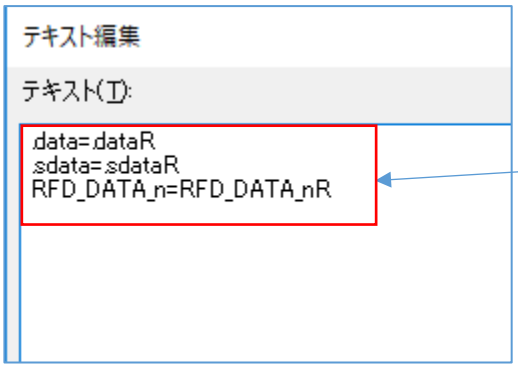
.stack_bss

RFD_DATA_nR

“OK”ボタン押下後、[セクションを自動的に配置する]を”はい”に戻して下さい。



[ROM から RAM へマップするセクション]の最も右の” ... ”ボタンで、”テキスト編集”画面を表示して、ROM から RAM へマップするセクションを追加します。



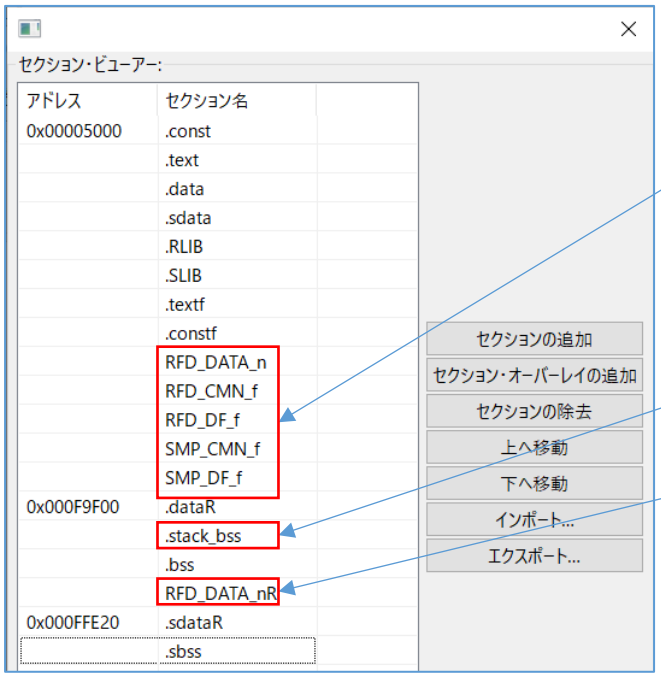
ROM から RAM へ マップするセクション
.data=.dataR
.sdata=.sdataR
RFD_DATA_n=RFD_DATA_nR

・e²studio でのデータ・フラッシュ・メモリ書き換えに必要なセクション追加

データ・フラッシュ・メモリ書き換えに必要なセクションを“セクション・ビューアー”で追加します。

プログラム領域へ追加：RFD_DATA_n, RFD_CMN_f, RFD_DF_f, SMP_CMN_f, SMP_DF_f

RAM へ追加：.stack_bss, RFD_DATA_nR

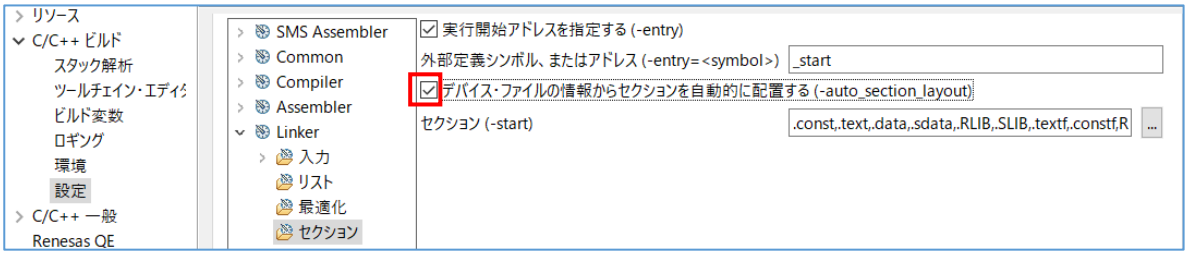


追加セクション
RFD_DATA_n
RFD_CMN_f
RFD_DF_f
SMP_CMN_f
SMP_DF_f

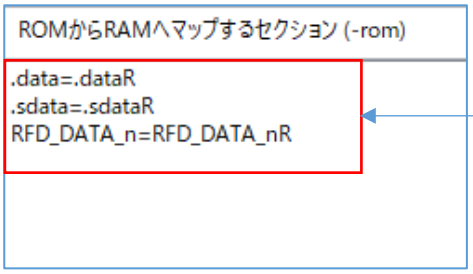
.stack_bss

RFD_DATA_nR

“OK”ボタン押下後、[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)]をチェックして下さい。



”C/C++ビルド” [設定] – “Linker” [出力] で表示した画面で[ROM から RAM へマップするセクション(-rom)]を設定します。



ROM から RAM へ マップするセクション
.data=.dataR
.sdata=.sdataR
RFD_DATA_n=RFD_DATA_nR

(3)　エクストラ領域(FSW)書き換えに必要なセクションの追加

・CS+でのエクストラ領域(FSW)書き換えに必要なセクションの追加

エクストラ領域(FSW)書き換えに必要なセクションを“セクション設定”画面で追加します。

プログラム領域へ追加：RFD_DATA_n, RFD_CMN_f, RFD_EX_f, SMP_CMN_f, SMP_EX_f

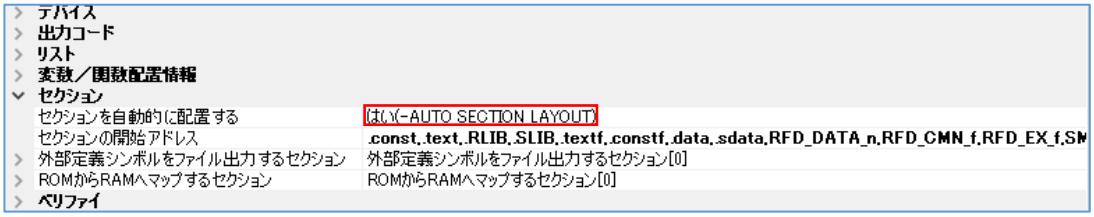
RAM へ追加：.stack_bss, RFD_DATA_nR, RFD_CMN_fR, RFD_EX_fR, SMP_CMN_fR, SMP_EX_fR

追加セクション
RFD_DATA_n
RFD_CMN_f
RFD_EX_f
SMP_CMN_f
SMP_EX_f

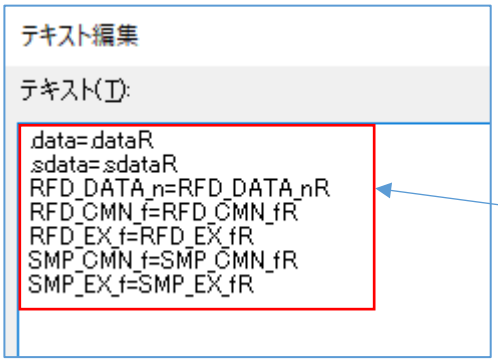
.stack_bss

RFD_DATA_nR
RFD_CMN_fR
RFD_EX_fR
SMP_CMN_fR
SMP_EX_fR

“OK”ボタン押下後、[セクションを自動的に配置する]を”はい”に戻して下さい。



[ROM から RAM へマップするセクション]の最も右の” ... ”ボタンで、”テキスト編集”画面を表示して、ROM から RAM へマップするセクションを追加します。



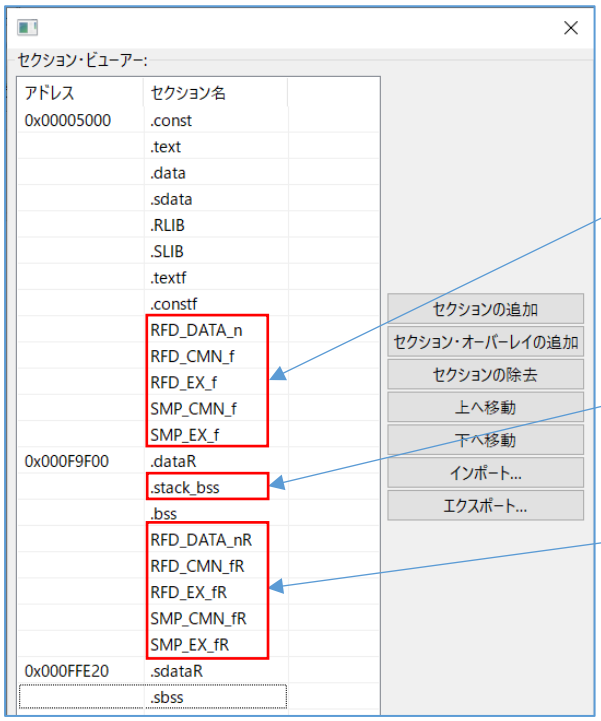
ROM から RAM へ マップするセクション
.data=.dataR
.sdata=.sdataR
RFD_DATA_n=RFD_DATA_nR
RFD_CMN_f=RFD_CMN_fR
RFD_EX_f=RFD_EX_fR
SMP_CMN_f=SMP_CMN_fR
SMP_EX_f=SMP_EX_fR

・e²studio でのエクストラ領域(FSW)書き換えに必要なセクション追加

エクストラ領域(FSW)書き換えに必要なセクションを“セクション・ビューアー”で追加します。

プログラム領域へ追加：RFD_DATA_n, RFD_CMN_f, RFD_EX_f, SMP_CMN_f, SMP_EX_f

RAM へ追加：.stack_bss, RFD_DATA_nR, RFD_CMN_fR, RFD_EX_fR, SMP_CMN_fR, SMP_EX_fR

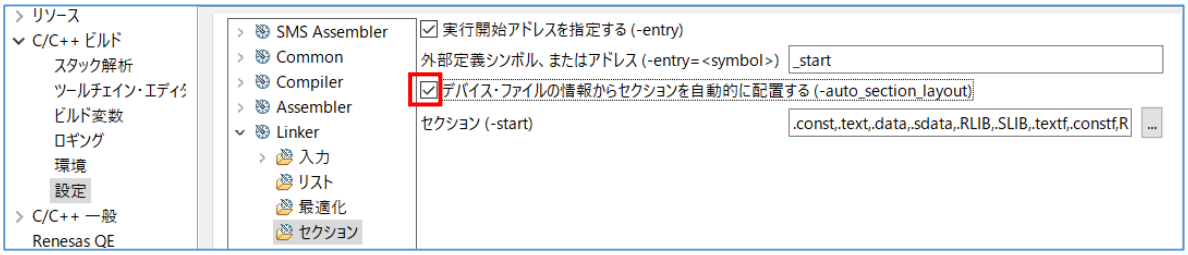


追加セクション
RFD_DATA_n
RFD_CMN_f
RFD_EX_f
SMP_CMN_f
SMP_EX_f

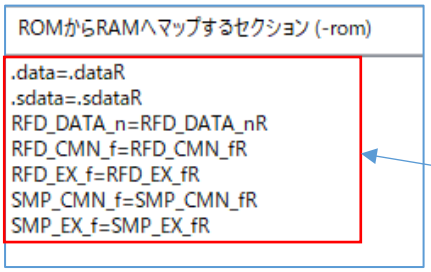
.stack_bss

RFD_DATA_nR
RFD_CMN_fR
RFD_EX_fR
SMP_CMN_fR
SMP_EX_fR

“OK”ボタン押下後、[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)]をチェックして下さい。



”C/C++ビルド” [設定] – “Linker” [出力] で表示した画面で[ROM から RAM へマップするセクション(-rom)]を設定します。



ROM から RAM へ マップするセクション
.data=.dataR
.sdata=.sdataR
RFD_DATA_n=RFD_DATA_nR
RFD_CMN_f=RFD_CMN_fR
RFD_EX_f=RFD_EX_fR
SMP_CMN_f=SMP_CMN_fR
SMP_EX_f=SMP_EX_fR

6.1.4　デバッグ・ツールの設定

ここでは、デバッグ・ツールに E2 Lite を選択してオンチップ・デバッグを行う場合のターゲット・ボードとの接続の設定について説明します。その他のデバッグ・ツール設定の詳細については、各統合開発環境のユーザーズマニュアルを参照してご確認ください。

CS+では、ツリーで”RL78 シミュレータ（ビルドツール）“[初期設定]でマウス右クリックし、表示された”使用するデバッグ・ツール”で”RL78 E2 Lite”を選択します。この時、”RL78 E2 Lite のプロパティ”画面が表示されます。ここで各タブを選択して、デバッグ・ツール設定を行います。

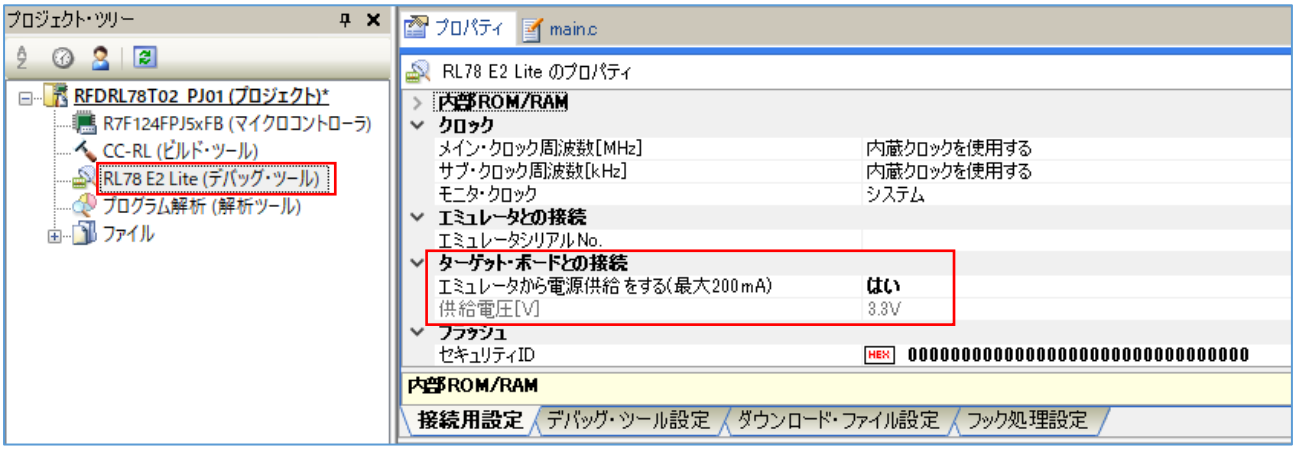
e^{2}studio では、ツリーで対象プロジェクトをマウス右クリックし、[デバッグ] - [デバッグの構成]を選択して表示された”デバッグ構成”画面のツリーで、[Renesas GDB Hardware Debugging]の対象プロジェクト（ここでは、”RFDRL78T02_PJ01 HardwareDebug”）を選択し、表示された”Debugger”タブで、デバッグ・ツール設定を行います。

注）ターゲット・ボードに他の電源が供給されている場合や電源供給容量が不足するなど、E2 Lite を含むエミュレータからターゲット・ボードへの電源供給ができない場合があります。必ず、対象デバイス用のエミュレータのユーザーズマニュアル、およびユーザーズマニュアル別冊(RL78 接続時の注意事項)をご参照の上、ご使用ください。

6.1.4.1　ターゲット・ボードとの接続の設定

・CS+でのターゲット・ボードとの接続（E2 Lite 経由）は、 “接続用設定”タブで設定（各領域共通）
- [ターゲット・ボードとの接続] 項目

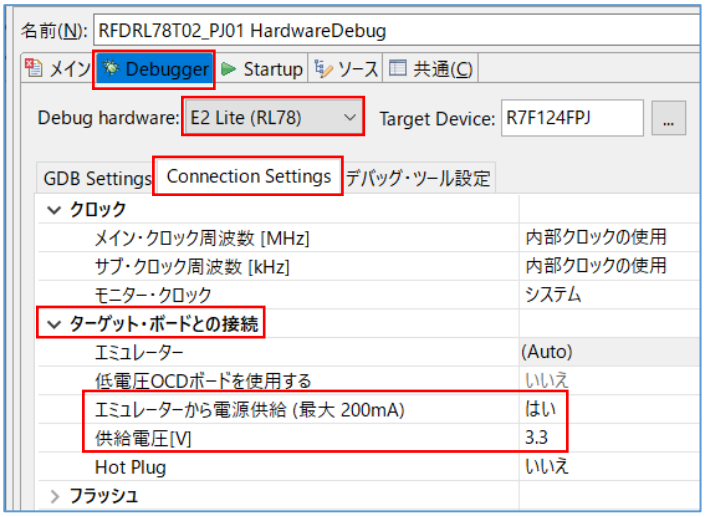
[エミュレータから電源供給をする（最大 200mA）]を”はい”に設定することで、E2 Lite からターゲット・ボードに電源供給（供給電圧:3.3V）することが可能です。



・e²studio でのターゲット・ボードとの接続（E2 Lite 経由）の設定は、 “Connection Setting”タブで設定（各領域共通）

- [ターゲット・ボードとの接続] 項目

[エミュレータから電源供給（最大 200mA）]を”はい”に設定することで、E2 Lite からターゲット・ボードに電源供給（供給電圧:3.3V）することが可能です。



6.2 IAR コンパイラを使用する場合のプロジェクトの作成

IAR コンパイラは、統合開発環境として IAR Embedded Workbench を使用して作成したプロジェクトへ RFD RL78 Type02 を登録し、ビルドすることができます。IAR Embedded Workbench を使用した場合のサンプル・プロジェクトの作成例を示します。IAR コンパイラ、および統合開発環境を理解するため、IAR Systems のツール製品のユーザーズマニュアルを参照してください。

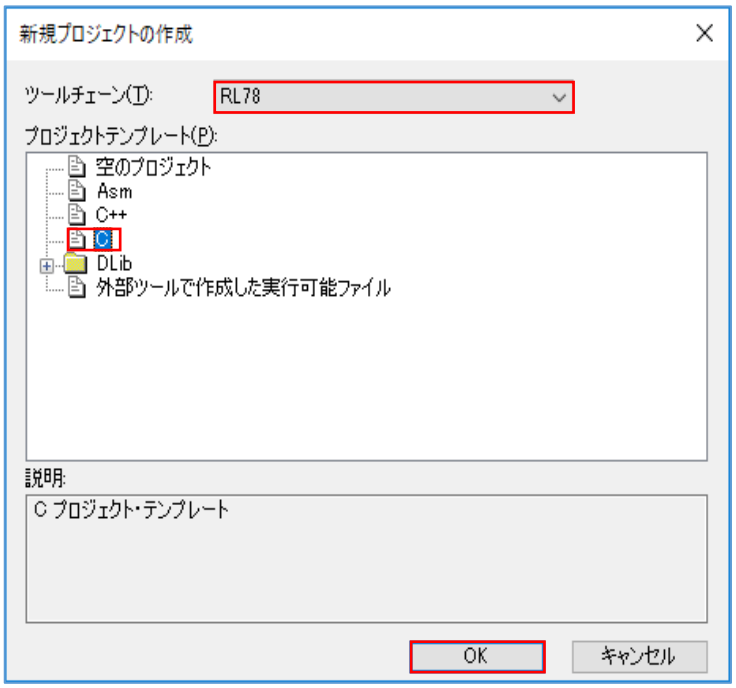
6.2.1 サンプル・プロジェクト作成例

(1) 統合開発環境 IAR Embedded Workbench を使用したサンプル・プロジェクト作成例

IAR Embedded Workbench を起動し、[プロジェクト]メニューの[新規プロジェクトの作成]を選択し、以下に示す ”新規プロジェクトの作成”ウインドウを起動します。

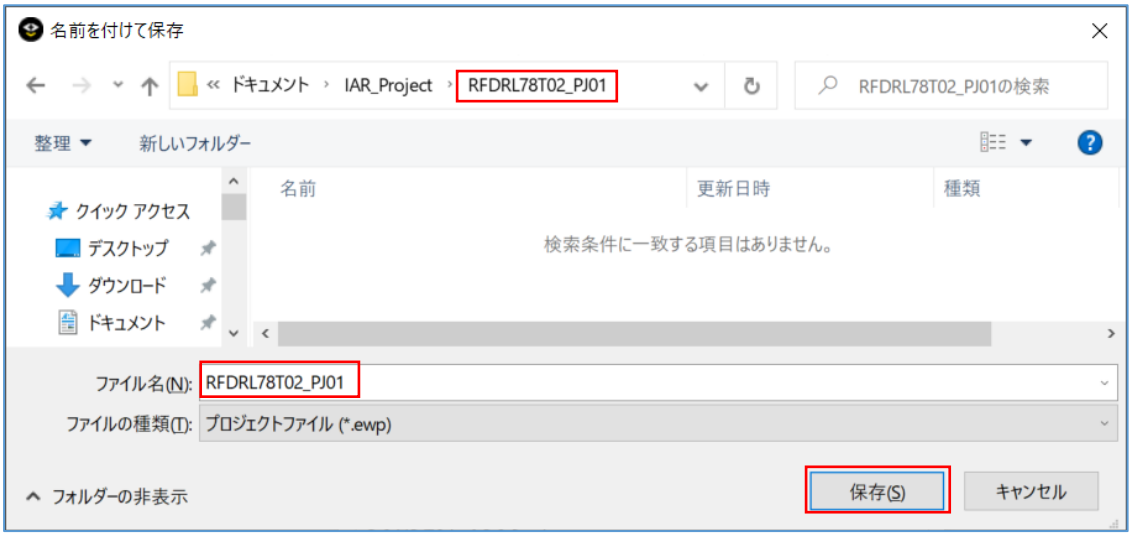
・[プロジェクトテンプレート]で、”C”を選択します。

・[OK]ボタンを押すと、[名前を付けて保存]ウインドウが表示されます。



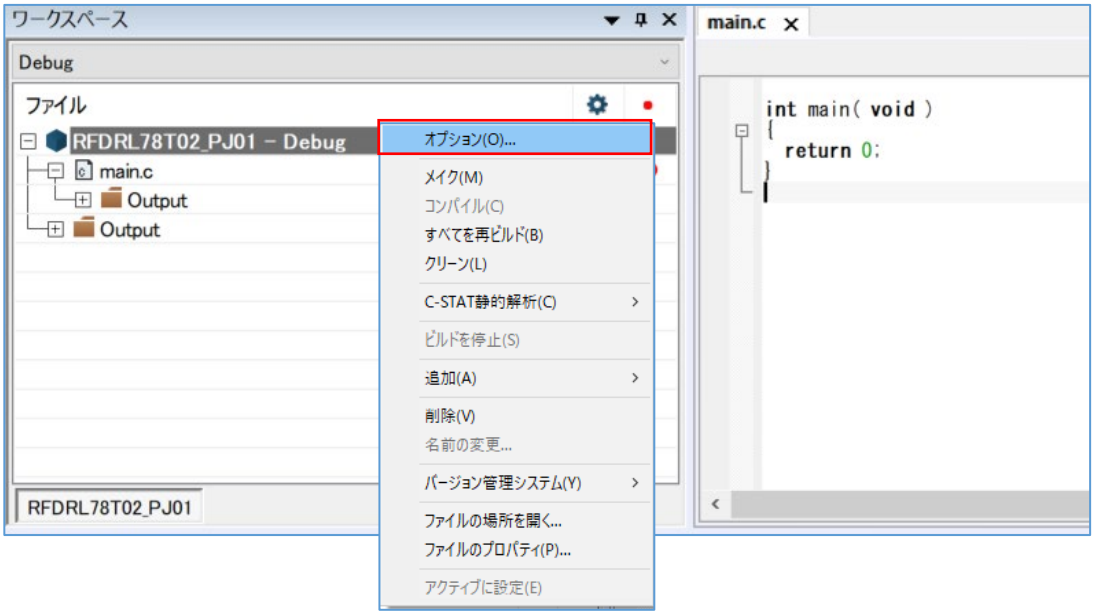
・ここでは、仮に”RFDRL78T02_PJ01”フォルダを作成し、フォルダ内へ移動します。

・ここでの[プロジェクト名]は、仮に”RFDRL78T02_PJ01”として保存します。



(2) 対象デバイスの選択

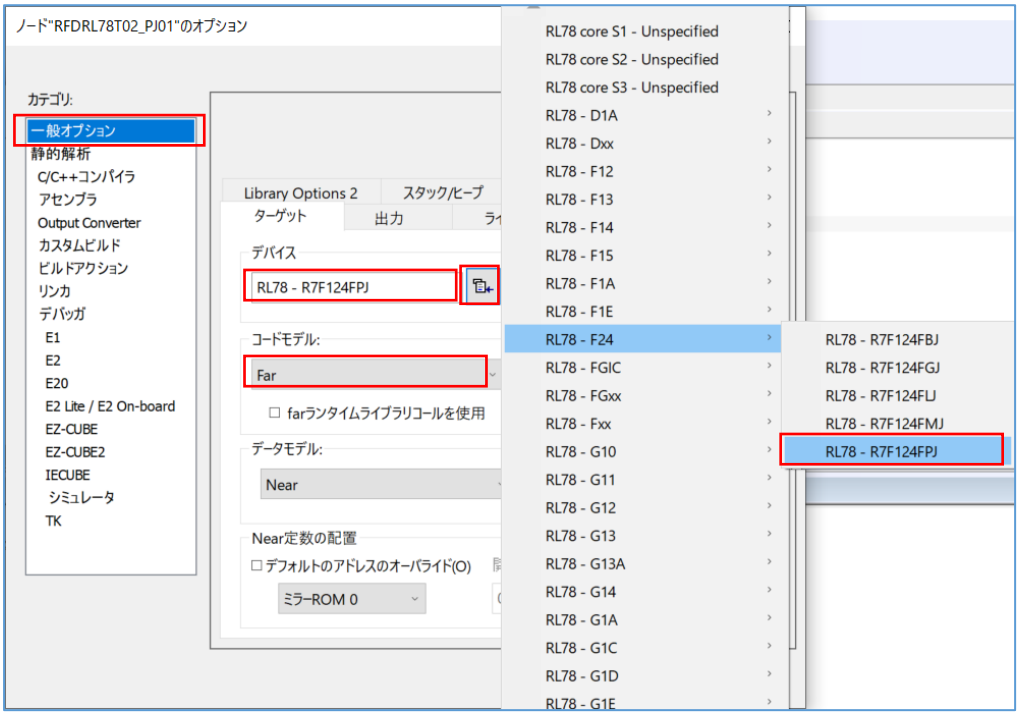
IAR Embedded Workbench ではツリーでプロジェクト(ここでは”RFDRL78T02_PJ01 - Debug“)のマウス右クリックで”オプション”を選択することにより、”オプション”の画面を表示します。



・”オプション”の画面内の[一般オプション] – [ターゲット]タブの各設定を行います。

[デバイス]の” ”ボタンで”RL78 - F24” - “RL78 - R7F124FPJ”を選択します。

ここでは、[コードモデル]に”Far”を選択し、[データモデル]に”Near”を選択します。

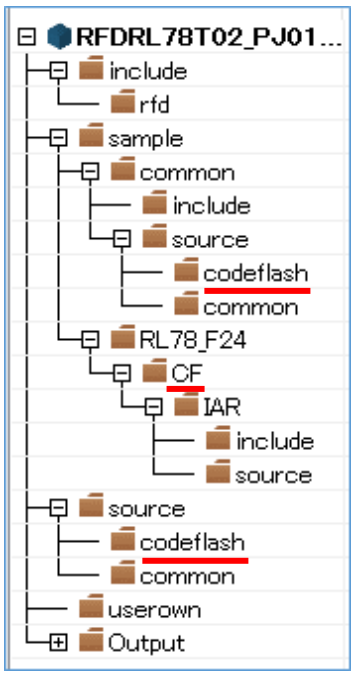


6.2.2 対象フォルダと対象ファイルの登録例

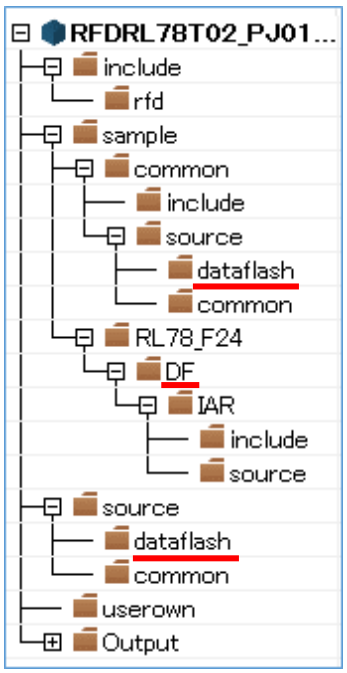
RFD RL78 Type02 を使用して、各領域[(1) コード・フラッシュ・メモリ、(2) データ・フラッシュ・メモリ、(3) エクストラ領域]を書き換える場合に必要なファイル、およびサンプル関数を含むファイルの登録例を記述します。RFD RL78 Type02 ソースプログラムファイルの各フォルダは、”include”, ”source”, ”userown”, ”sample”で、書き換える領域により各フォルダ内の対象ファイルを選択して登録します。

ここでは、IAR Embedded Workbench でフォルダを登録する代わりに、[プロジェクト]メニューの[グループの追加]を選択し、RFD RL78 Type02 のフォルダ構成と同様のグループを追加してファイルを登録する例を示します。（グループを作らずに登録することも可能です。）

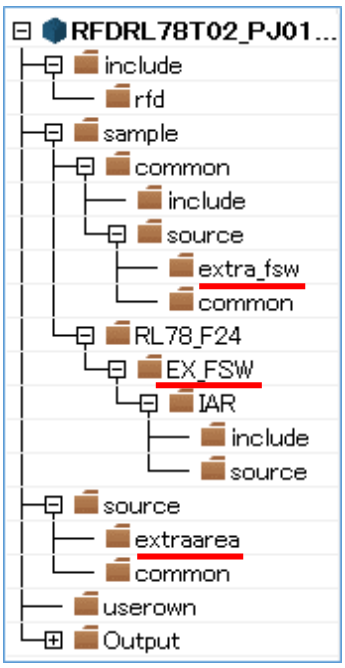
各領域[(1)コード・フラッシュ・メモリ、(2)データ・フラッシュ・メモリ、(3)エクストラ領域]のグループを追加した例を示します。（領域ごとに異なるグループ名を”▬”で示しています。）



(1) コード・フラッシュ・メモリ



(2) データ・フラッシュ・メモリ



(3) エクストラ領域

・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

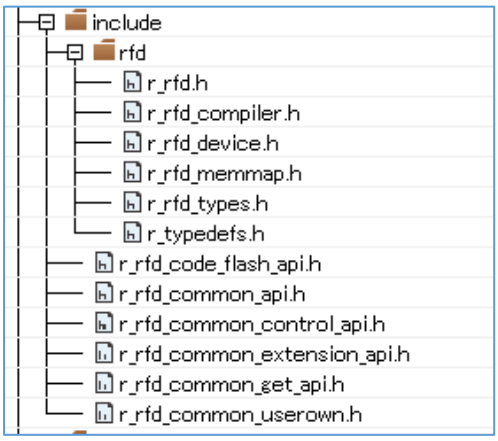
作成されたプロジェクトには、自動的に追加されるファイルがあります。これらと同様のファイルは、RFD RL78 Type02 の”sample”フォルダ内にも存在するため、ツリーで各ファイルを選択し、各統合開発環境の機能を使用して、プロジェクトから外します。

- IAR Embedded Workbench では、ツリーでファイルをマウス右クリック、”削除”機能で対象の”main.c”ファイルを除外します。

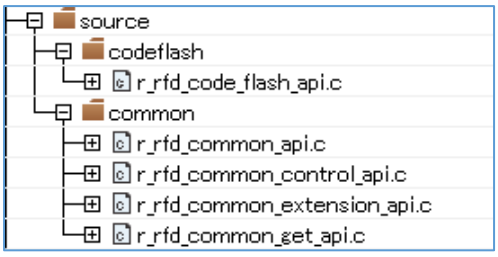
(1) コード・フラッシュ・メモリを書き換える場合の対象グループと対象ファイルの登録

RFD RL78 Type02 ソースプログラムファイルの各グループ(”include”, ”source”, ”userown”, ”sample”)と登録ファイルを以下に示します。

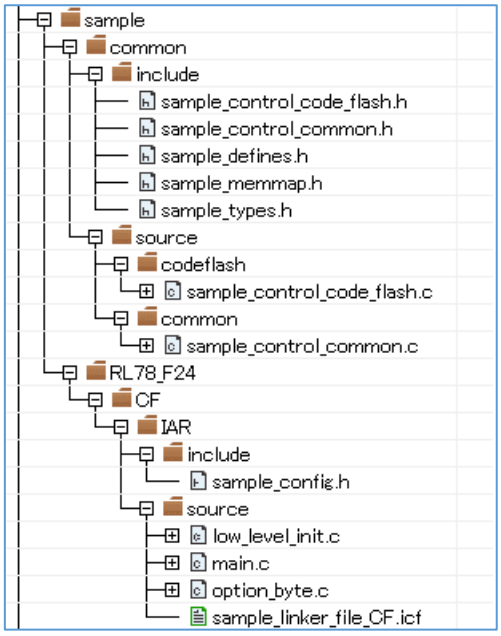
include グループ内



source グループ内



sample グループ内

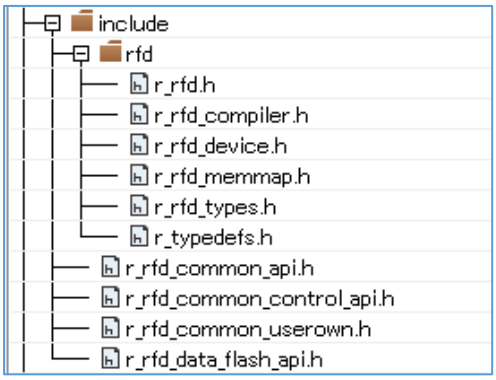


userown グループ内

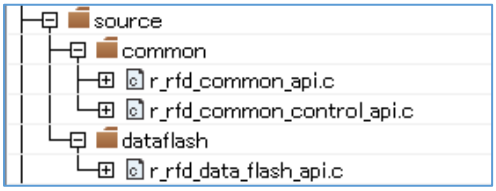
(2) データ・フラッシュ・メモリを書き換える場合の対象グループと対象ファイルの登録

RFD RL78 Type02 ソースプログラムファイルの各グループ(”include”, ”source”, ”userown”, ”sample”)と登録ファイルを以下に示します。

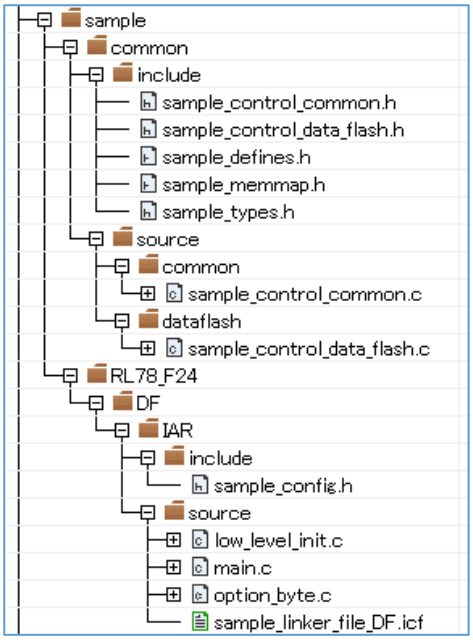
include グループ内



source グループ内



sample グループ内

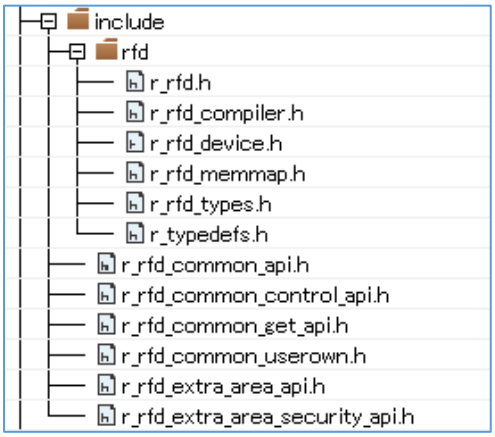


userown グループ内

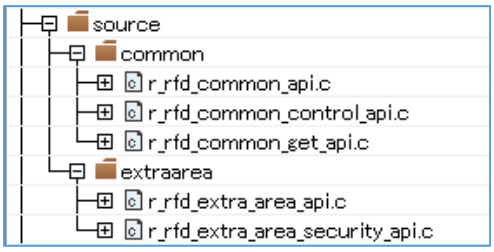
(3) エクストラ領域(FSW 設定)を書き換える場合の対象グループと対象ファイルの登録

RFD RL78 Type02 ソースプログラムファイルの各グループ(”include”, source”, ”userown”, ”sample”)と登録ファイルを以下に示します。

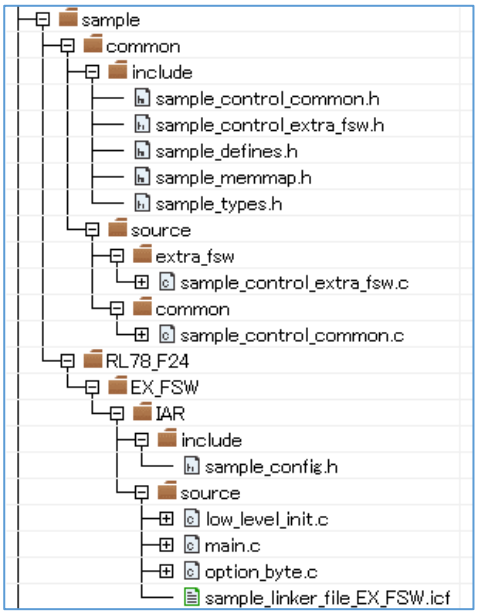
include グループ内



source グループ内



sample グループ内



userown グループ内

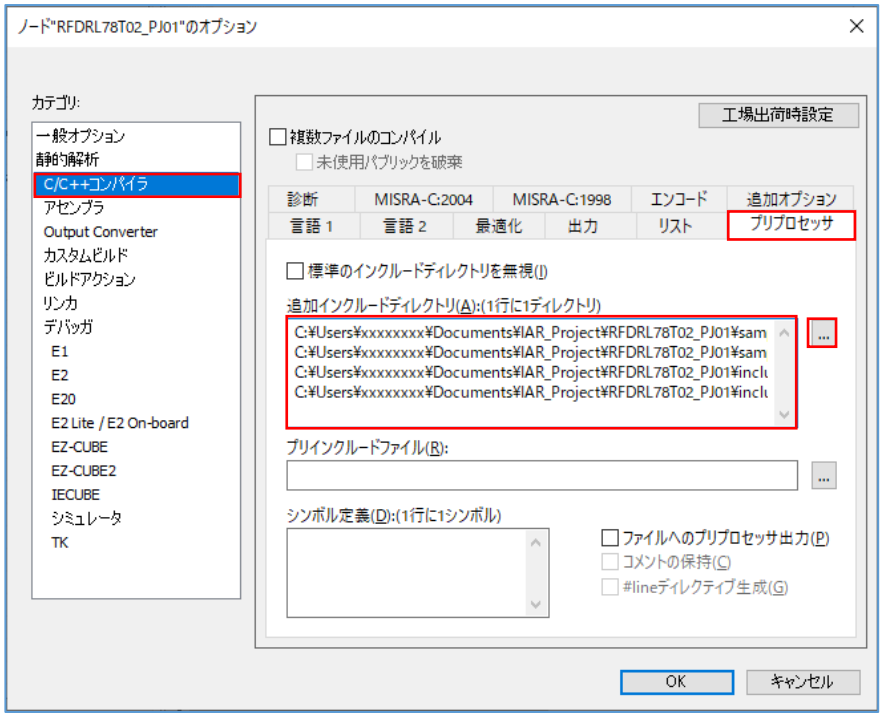
6.2.3 統合開発環境の設定

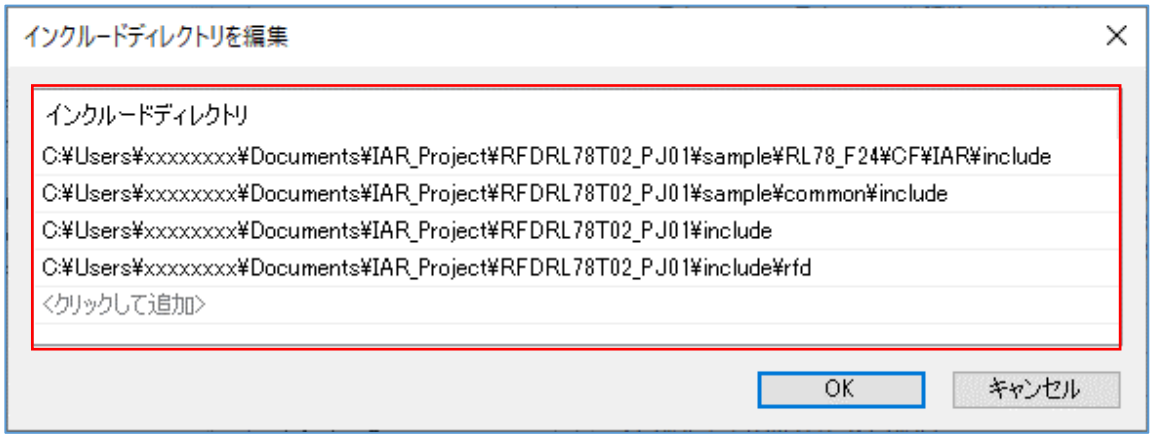
IAR コンパイラで RFD RL78 Type02 をビルドして実行するための統合開発環境の設定を行います。IAR Embedded Workbench ではツリーで[プロジェクト]をマウス右クリックして”オプション”を選択、表示された画面内の”カテゴリ”を選択して各設定を行います。

6.2.3.1 インクルード・パスの設定

IAR Embedded Workbench でのインクルード・パスの設定は、カテゴリの”C/C++コンパイラ”を選択し、“プリプロセッサ”タブで設定します（対象領域により変更）。

- [追加インクルードディレクトリ(A) : (1 行に 1 ディレクトリ)]で”パス編集”ウインドウを表示して、インクルードディレクトリのパスを設定します。





- 設定するディレクトリパスの例

“C:\Users\xxxxxxxx\Documents\IAR_Project\"に、RFD RL78 Type02 ソースプログラムファイルの各フォルダ(”include”, ”source”, ”userown”, ”sample”)を置いた場合の例です。

(1) コード・フラッシュ・メモリ書き換え

C:\Users\xxxxxxxx\Documents\IAR_Project\RFDRL78T02_PJ01\sample\RL78_F24\CF\IAR\include

C:\Users\xxxxxxxx\Documents\IAR_Project\RFDRL78T02_PJ01\sample\common\include

C:\Users\xxxxxxxx\Documents\IAR_Project\RFDRL78T02_PJ01\include

C:\Users\xxxxxxxx\Documents\IAR_Project\RFDRL78T02_PJ01\include\rfd

(2) データ・フラッシュ・メモリ書き換え

C:\Users\xxxxxxxx\Documents\IAR_Project\RFDRL78T02_PJ01\sample\RL78_F24\DF\IAR\include

C:\Users\xxxxxxxx\Documents\IAR_Project\RFDRL78T02_PJ01\sample\common\include

C:\Users\xxxxxxxx\Documents\IAR_Project\RFDRL78T02_PJ01\include

C:\Users\xxxxxxxx\Documents\IAR_Project\RFDRL78T02_PJ01\include\rfd

(3) エクストラ領域(FSW)書き換え

C:\Users\xxxxxxxx\Documents\IAR_Project\RFDRL78T02_PJ01\sample\RL78_F24\EX_FSW\IAR\include

C:\Users\xxxxxxxx\Documents\IAR_Project\RFDRL78T02_PJ01\sample\common\include

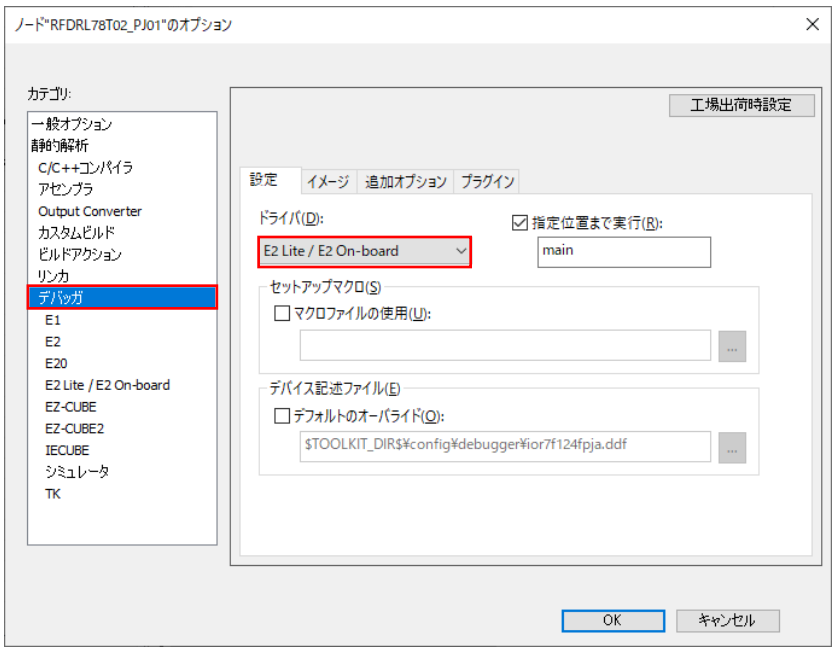
C:\Users\xxxxxxxx\Documents\IAR_Project\RFDRL78T02_PJ01\include

C:\Users\xxxxxxxx\Documents\IAR_Project\RFDRL78T02_PJ01\include\rfd

注)インクルードディレクトリのパス設定については、絶対パスで指定しているとプロジェクトをコピーした時に再設定が必要になります。プロジェクトをコピーしても使用できるよう相対パス($PROJ_DIR$)を指定することも可能です。指定方法については、IAR Embedded Workbench の[Help]から各リファレンスマニュアルをいただき、必要に応じて設定してください。

6.2.3.2 デバッガの設定

・オンチップ・デバッグを実施することを前提として、[デバッガ] – [設定]タブの[ドライバ]で”E2 Lite / E2 On-Board”を選択します。

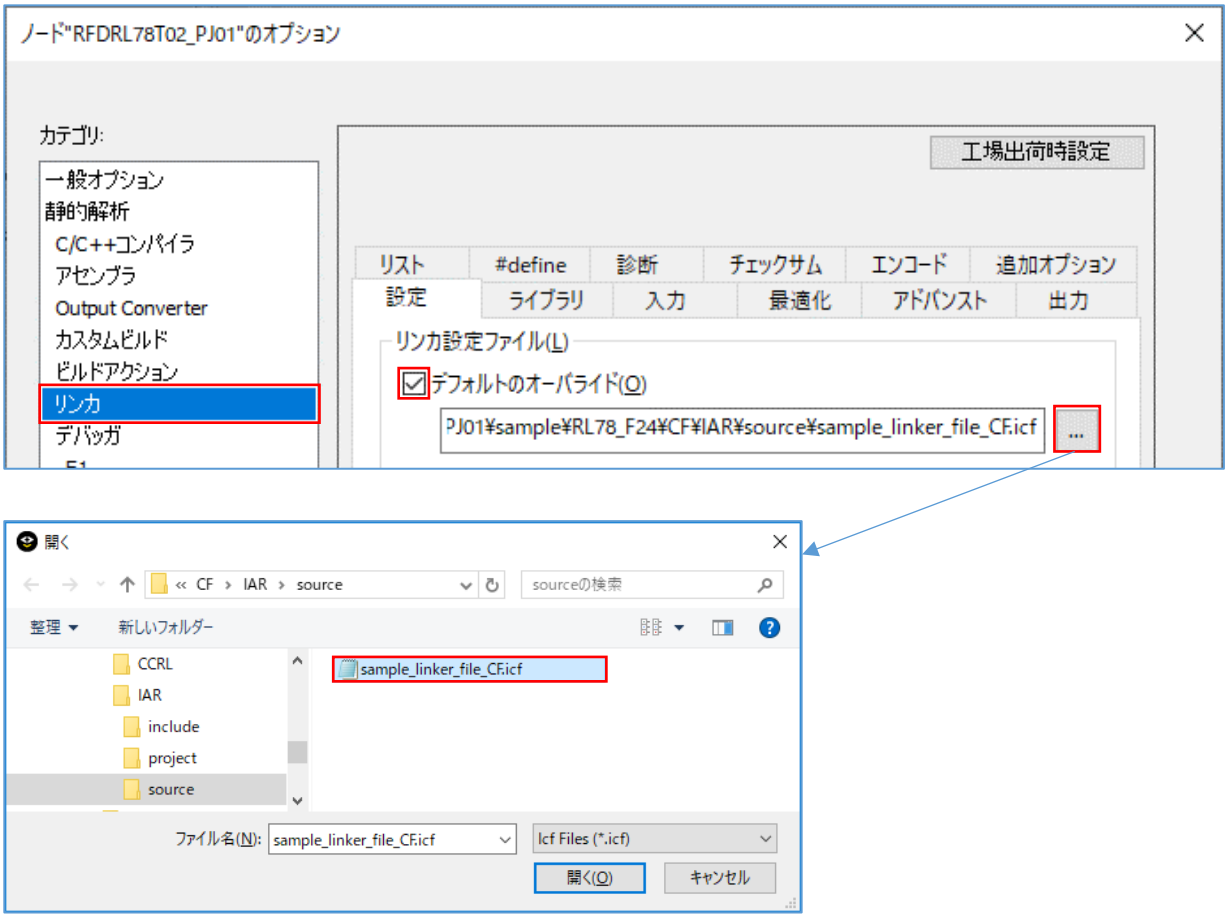


注) その他の設定項目については、IAR Embedded Workbench の[Help]から各リファレンスマニュアルをご参照いただき、必要に応じて設定してください。

6.2.4 リンカ設定ファイル(.icf)の設定

IAR Embedded Workbench では、ビルドで実行するリンク設定をリンカ設定ファイル(*.icf)に記述します。ツリーで[プロジェクト]のマウス右クリックで”オプション”を選択、表示された画面内の[リンカ]で、[設定] – [デフォルトのオーバライド(O)]にチェックを入れ、”...”ボタンの”開く”画面でリンカ設定ファイル(*.icf)を選択します。ここでは、RFD RL78 Type02 用に準備されている” sample_linker_file_（領域名）.icf”ファイルを選択します。書き換え領域ごとのリンカ設定用ファイル(*.icf)は以下の通りです。

- コード・フラッシュ書き換え用：sample_linker_file_CF.icf (\Sample\RL78_F24\CF\IAR\source\)
- データ・フラッシュ書き換え用：sample_linker_file_DF.icf (\Sample\RL78_F24\DF\IAR\source\)
- エクストラ領域(FSW)書き換え用：sample_linker_file_EX_FSW.icf (\Sample\RL78_F24\EX_FSW\IAR\source\)



注)リンカ設定ファイルの記述内容、及び記述方法の詳細については、IAR Embedded Workbench の[Help]から各リファレンスマニュアルをご参照ください。

6.2.4.1 セクション項目の設定

RFD RL78 Type02 で準備されているリンカ設定ファイル(*.icf)で追加しているセクションの概要を記述します。

注)リンカ設定ファイルのセクション項目の設定、及び機能の詳細は、IAR Embedded Workbench の[Help]から各リファレンスマニュアルをご参照ください。

・RFD RL78 Type02 の”リンカ設定ファイル”で記述しているセクション項目の設定概要

(1) コード・フラッシュ・メモリ書き換えに必要なセクションの追加

RFD_DATA, RFD_CMN, RFD_CF, SMP_CMN, SMP_CF の各セクションの初期値を ROM 領域(ROM_far)へ追加し、それぞれ RAM 領域(RAM_near, RAM_code)のセクションへコピーする必要があります。

- ROM_far 領域の追加セクション(RAM 領域へコピーするためのデータとプログラム) :
  RFD_DATA_init, RFD_CMN_init, RFD_CF_init, SMP_CMN_init, SMP_CF_init
- RAM_near 領域の追加セクション(ROM 領域からコピーされるデータ) :
  RFD_DATA
- RAM_code 領域の追加セクション(ROM 領域からコピーされるプログラム) :
  RFD_CMN, RFD_CF, SMP_CMN, SMP_CF

(2) データ・フラッシュ・メモリ書き換えに必要なセクションの追加

RFD_DATA の初期値と RFD_CMN, RFD_DF, SMP_CMN, SMP_DF の各セクションを ROM 領域(ROM_far)へ追加し、RFD_DATA は RAM 領域(RAM_near)のセクションへコピーする必要があります。

- ROM_far 領域の追加セクション(プログラムと RAM 領域へコピーするためのデータ) :
  RFD_DATA_init, RFD_CMN, RFD_DF, SMP_CMN, SMP_DF
- RAM_near 領域の追加セクション(ROM 領域からコピーされるデータ) :
  RFD_DATA

(3) エクストラ領域(FSW)書き換えに必要なセクションの追加

RFD_DATA, RFD_CMN, RFD_EX, SMP_CMN, SMP_EX の各セクションの初期値を ROM 領域(ROM_far)へ追加し、それぞれ RAM 領域(RAM_code)のセクションへコピーする必要があります。

- ROM_far 領域の追加セクション(RAM 領域へコピーするためのデータとプログラム) :
  RFD_DATA_init, RFD_CMN_init, RFD_EX_init, SMP_CMN_init, SMP_EX_init
- RAM_near 領域の追加セクション(ROM 領域からコピーされるデータ) :
  RFD_DATA
- RAM_code 領域の追加セクション(ROM 領域からコピーされるプログラム) :
  RFD_CMN, RFD_EX, SMP_CMN, SMP_EX

6.2.4.2 オプション・バイトの設定

RL78 のオプション・バイト定義は、IAR Embedded Workbench 付属のリンカ設定ファイル(*.icf)、及び RFD RL78 Type02 用に準備されている” sample_linker_file_（領域名）.icf”ファイルに記述されています。RFD RL78 Type02 でのオプション・バイト値は、”option_byte.c”ファイルに記述されています。

注）リンカ設定ファイルのオプション・バイトの設定については、IAR Embedded Workbench の[Help]から各リファレンスマニュアルをご参照ください。

RFD RL78 Type02 用リンカ設定ファイル(*.icf)のオプション・バイトの定義例

```
define block OPT_BYTE with size = 5   { R_OPT_BYTE,
                                        ro section .option_byte,
                                        ro section OPTBYTE };
                                  |
place at address mem:0x000C0          { block OPT_BYTE };
```

”option_byte.c”ファイル内のオプション・バイト値の記述例

```
#pragma location = "OPTBYTE"
__root const unsigned char option_bytes[5] = {
    0x6E,   /* 01101110 */
            /*        | */
            /*  |||||+-- Watchdog timer     */
            /*  |||||    operation stopped  */
            /*  |||||    in HALT/STOP mode  */
            /*  |||+---- Watchdog timer     */
            /*  |||      overflow time is   */
            /*  |||      2^16 / fIL =       */
            /*  |||      3799.18 ms         */
            /*  ||+------ Watchdog timer     */
            /*  ||       operation disabled */
            /*  |+------- 100% window open   */
            /*  |         period             */
            /* +--------- Interval interrupt */
            /*            is not used        */
    0x6F,   /* 01101111 */
            /* |||||||| */
            /* +++|++++-- LVD reset mode */
            /*    +------ Control of clock monitor operation is enabled */
    0xE8,   /* 11101000 */
            /*   |||||| */
            /*   |++++++-- 40 MHz */
            /*   +------- Selects P130 as a general port pin (output only) */
    0xA5,   /* 10100101 */
            /* | |    || */
            /* +-|---++-- OCD: enables on-chip debugging function */
            /*   +------- Enables flash serial programming operation. */
    0xFE    /* Enables read of on-chip debug and flash serial programming security ID */
};
```

- ユーザ・オプション・バイト値の説明：

”option_byte.c”ファイル内のユーザ・オプション・バイト(000C0H-000C2H)の値は”6E6FE8”です。

（WDT 停止、LVD（reset モード）、40MHz [RL78/F24 の例]）

”option_byte.c”ファイル内のオンチップ・デバッグ・オプション・バイト(000C3H/040C3H)の値は”A5”です。

（オンチップ・デバッグ動作許可の例）

注）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、ユーザ・アプリケーションで使用する設定値を書き込んでください。

6.2.5 オンチップ・デバッグの設定

プロジェクトのビルド実行後、E2 Lite を接続した状態で、 [プロジェクト]メニューから[ダウンロードしてデバッグ]を選択して、デバッグを開始します。

6.2.5.1 接続エラーに関する対処の例

ここでは、オンチップ・デバッグを実行時の接続エラーに関する対処(よくある例)として、”ID コード”の不一致や”電源”が正しく設定されていない場合について説明します。

注）その他の原因によりターゲットに接続できない場合は、IAR Embedded Workbench の[Help]から各リファレンスマニュアルをご確認ください。

[ダウンロードしてデバッグ]を選択して、デバッグを開始するときに、”E2 Lite ハードウェア設定”画面が表示される場合があります。原因として、”ID コード”の不一致や”電源”が正しく設定されていない場合が考えられます。

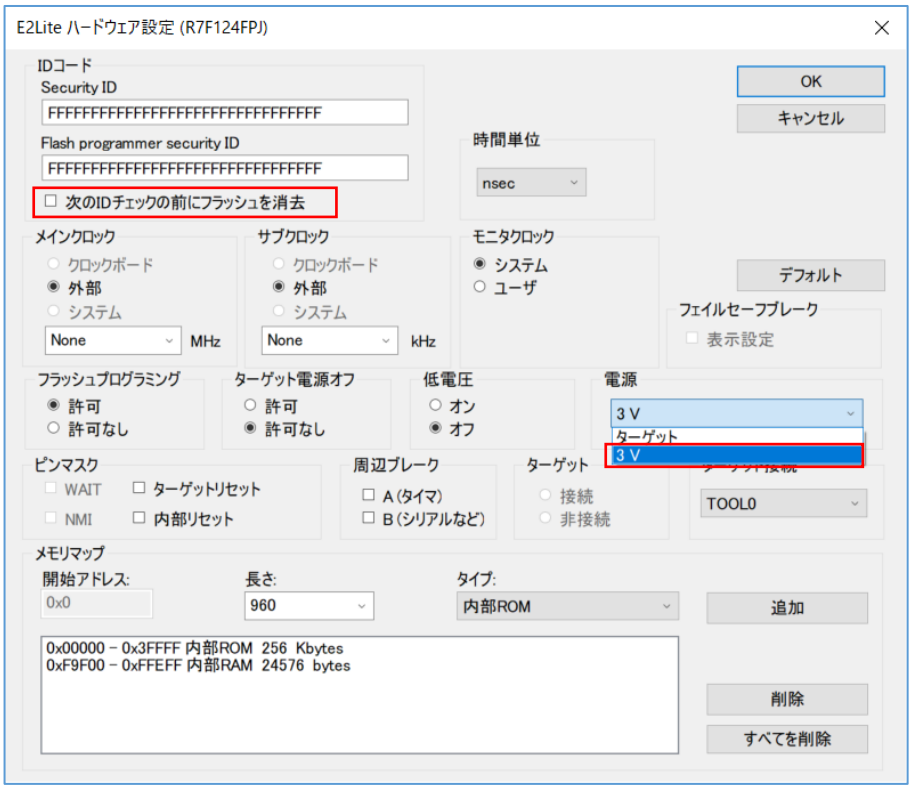
- ID コードが不一致の場合：

“ID コードをベリファイできない。”等のメッセージが表示されることがあります。この場合は、”E2 Lite ハードウェア設定”画面]の[ID コード]で、”次の ID チェックの前にフラッシュを消去”をチェックし、一度フラッシュ・メモリを消去することで、接続できる場合があります。

- 電源が設定されていない場合：

“電源”の初期状態は、”ターゲット”ですが E2 Lite から電源を供給する場合は、プルダウン・メニューで”3V”を選択します。

注）ターゲットに電源が供給されている場合、絶対に”3V”(E2Lite から電源を供給)に設定しないでください。



6.3 デバイス変更に伴う設定

RL78/F24(R7F124FPJ) 以外のデバイスを使用する場合、ROM や RAM、データ・フラッシュ・メモリのサイズが異なるため、セクションのアドレス設定やサンプル・プログラムの一部を変更する必要があります。この項では変更手順、および変更箇所について説明しています。

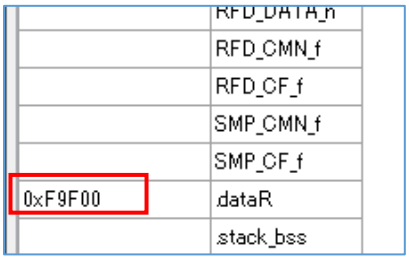
設定値などの変更には"RL78 用 Renesas Flash Driver, EEPROM Emulation Software 対象 MCU リスト - Automotive "（以降、対象 MCU リスト）を参照し、使用しているデバイスにあわせて設定値などを変更します。下記に、対象 MCU リストの参照例と変更箇所の記載例を示します。

- 対象 MCU リストの参照例

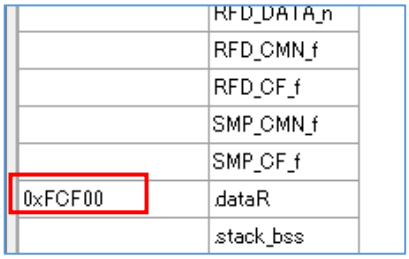
例えば、次の図のように[R-1] が指している箇所の設定値（RAM の先頭アドレス）を変更するとします。ここでは、対象 MCU リストに記載されている RAM の先頭アドレス [R-1] (RAM Start Address)の設定値を参照して、RL78/F23(R7F123FxG)の値を設定します。

例）RAM の先頭アドレス変更箇所：RL78/F24(R7F124FPJ RAM: 24KB)

[R-1] →



例）RL78/F23(R7F123FxG RAM: 12KB)を使用する場合の RAM の先頭アドレス値を設定



[R-1] に設定する値は、対象 MCU リストを参照して対象デバイスの RAM の先頭アドレスを設定します。

対象 MCU リストの Target MCU name の列から、R7F123FxG の行を検索します。次に、[R-1] の列から R7F123FxG の行と交わるセルを検索します。

・対象 MCU リストの表示例

MCU Group	Code Flash memory		User RAM		Data Flash memory		[R-1]	[R-2]	[R-3]	[R-4]	[R-5]	[R-6]	[R-7]	[R-8]	Target MCU name
	Size (bytes)	Start/End Address	Size (bytes)	Start/End Address	Size (bytes)	Start/End Address	RAM Start Address	ROM End Address 1	ROM End Address 2	Data Flash End Address	OCD_ROM	Trace_RAM	Hot plug-in	END_BLOCK	
RL78/F23	128K	0x00000 - 0x1FFFF	12K	0xFCF00 - 0xFFEFF	8K	0xF1000 - 0xF2FFF	0xFCF00	0x0FFFF	0x1FFFF	0xF2FFF	0x1FE00	0xFD300	0xFD500	128	R7F123FxG(x = B, G, L, M)
RL78/F24	256K	0x00000 - 0x3FFFF	24K	0xF9F00 - 0xFFEFF	16K	0xF1000 - 0xF4FFF	0xF9F00	0x0FFFF	0x3FFFF	0xF4FFF	0x3FE00	0xFA300	0xFA500	256	R7F124FxJ(x = B, G, L, M, P)

“0xFCF00”が該当するので、RL78/F23(R7F123FxG) における[R-1] の設定値に“0xFCF00”を設定します

[R-1] RAM Start Address	[R-2] ROM End Address 1	[R-3] ROM End Address 2	[R-4] Data Flash End Address	[R-5] OCD_ROM	[R-6] Trace_RAM	[R-7] Hot plug-in	[R-8] END_BLOCK	Target MCU name
0xFCF00	0x0FFFF	0x1FFFF	0xF2FFF	0x1FE00	0xFD300	0xFD500	128	R7F123FxG(x = B, G, L, M)
0xF9F00	0x0FFFF	0x3FFFF	0xF4FFF	0x3FE00	0xFA300	0xFA500	256	R7F124FxJ(x = B, G, L, M, P)

- 変更箇所の記載例

「6.3.1 CC-RL コンパイラ環境の設定」以降に、RL78/F24(R7F124FPJ)の設定値から変更が必要な箇所を記載しています。その変更が必要な箇所には、"[R- x] →" のように示しているので、対象 MCU リストから使用しているデバイスに該当する[R- x] の設定値を検索し、[R- x] に設定値を入力します。(x = 1, 2, 3…)

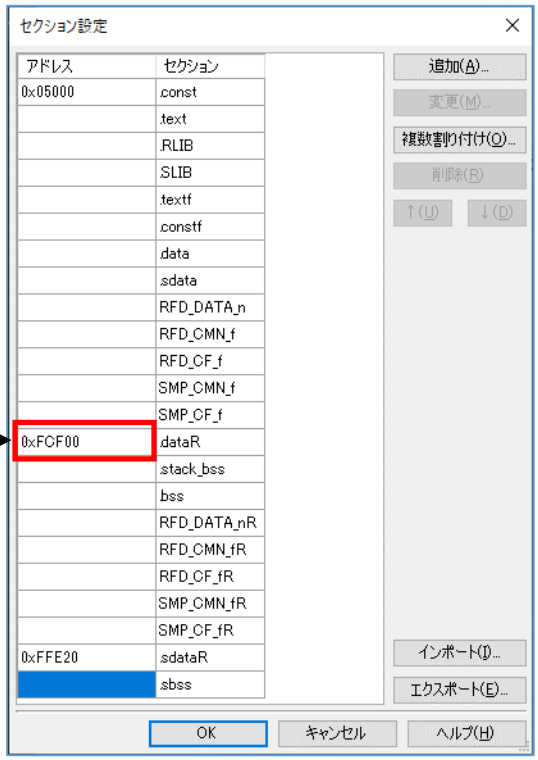
・コード・フラッシュ(CF)書き換え時のセクション設定（RAM の先頭アドレス）の変更箇所の例：

CS+（CC-RL コンパイラ）

例）RL78/F24(R7F124FPJ)用設定 RAM: 24KB

[R-1] →

例）RL78/F23(R7F123FxG)用設定 RAM: 12KB



6.3.1 CC-RL コンパイラ環境の設定

CC-RL コンパイラ環境(CS+、e²studio)を使用する場合の変更箇所と変更例を記載します。

6.3.1.1 RAM の開始アドレス設定

RAM 領域の開始アドレスを設定します。

sample フォルダに格納されている"cstart.asm"は、RL78/F24(R7F124FxJ)を RAM サイズ 24KB で使用する設定のため、RAM サイズを 24KB 以外で使用する場合は設定を変更する必要があります。

ここでは、RL78/F23(R7F123FxG)を RAM サイズ 12KB で使用するための設定変更例を記載します。

RL78/F23(R7F123FxG)を RAM サイズ 12KB で使用するためには、sample フォルダに格納されている"cstart.asm"で RAM 開始アドレス設定レジスタ(RAMSAR)の値を"0x9F"から"0xCF"に変更します。
RAM 開始アドレス設定レジスタ(RAMSAR)の詳細は、デバイスのハードウェアマニュアルでご確認ください。

- sample フォルダに登録されている"cstart.asm"のパス（領域名：CF or DF or EX_FSW）
  \sample\RL78_F24\（領域名）\CCRL\source\cstart.asm

RL78/F24 用設定(RAM:24KB) R7F124FxJ の例

```
;---------------------------------------------------
; setting RAMSAR=0x9F RAM area is 24KB (RL78/F24)
;---------------------------------------------------
MOV !RAMSAR, #0x9F
```

RL78/F23 用設定(RAM:12KB) R7F123FxG の例

```
;---------------------------------------------------
; setting RAMSAR=0x9F RAM area is 12KB (RL78/F23)
;---------------------------------------------------
MOV !RAMSAR, #0xCF
```

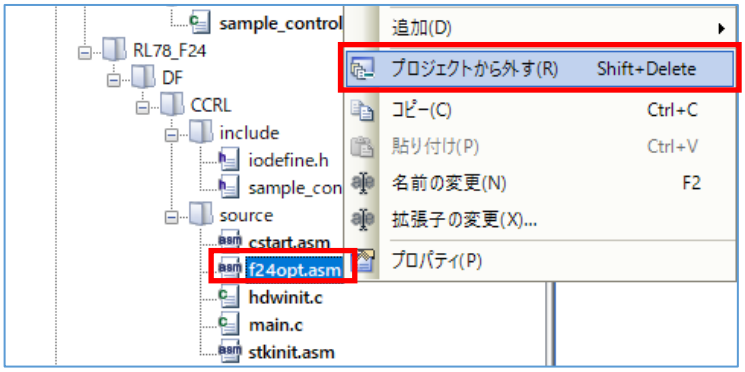
6.3.1.2 未使用ファイルの除外

sample フォルダに格納されている"f24opt.asm"は RL78/F24 でのみ使用するファイルのため、RL78/F23 を使用する場合は"f24opt.asm"をプロジェクトから外します。

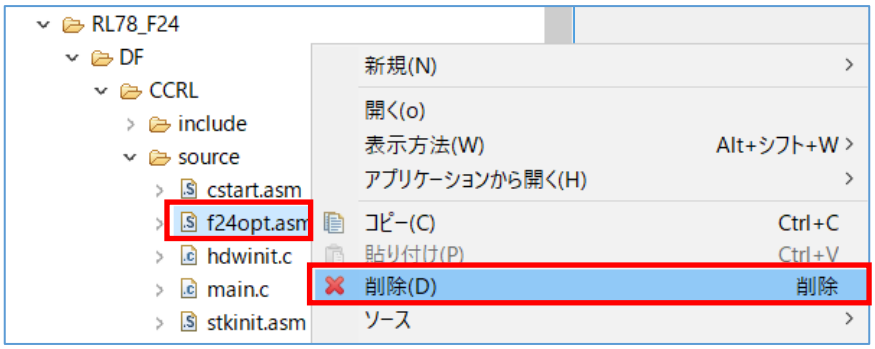
- sample フォルダに登録されている"f24opt.asm"のパス（領域名：CF or DF or EX_FSW）

  \sample\RL78_F24\（領域名）\CCRL\source\f24opt.asm

・CS＋での"f24opt.asm"をプロジェクトから外す例



・e²studio での"f24opt.asm"をプロジェクトから外す例



6.3.1.3 セクション設定

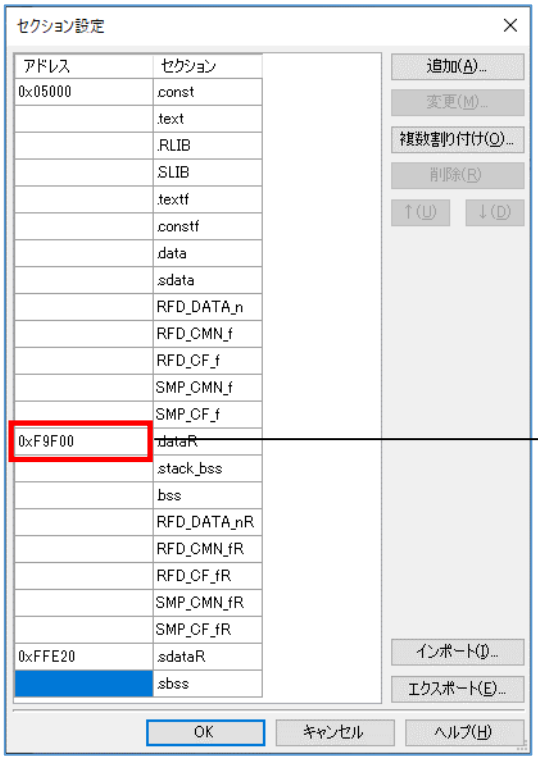
セクション設定で使用する製品の RAM 領域の先頭アドレスを設定します。

RL78/F24(R7F124FxJ)から RL78/F23(R7F123FxG)へ変更する場合を例として示します。RAM のサイズが 24KB から 12KB へ変更されるため、RAM の先頭アドレスを"0xF9F00"から"0xFCF00"へ変更します。各製品の RAM の先頭アドレスについては、対象 MCU リストの[R-1] 列をご確認ください。

・CS＋での CF/DF/EX 書き換え時のセクション設定（RAM の先頭アドレス）の変更箇所の例

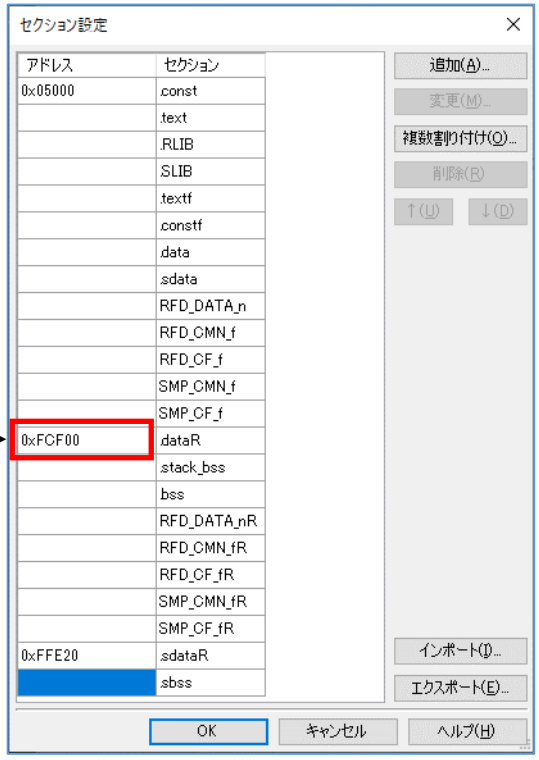
(1) コード・フラッシュ(CF)書き換え時

例）RL78/F24(R7F124FPJ)用設定 RAM: 24KB



[R-1] →

例）RL78/F23(R7F123FxG)用設定 RAM: 12KB

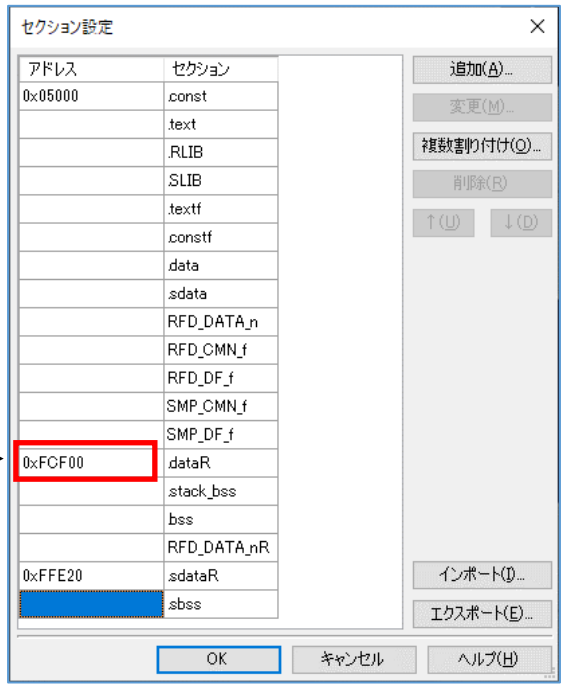


(2) データ・フラッシュ(DF)書き換え時

例）RL78/F24(R7F124FPJ)用設定 RAM: 24KB

[R-1] →

例）RL78/F23(R7F123FxG)用設定 RAM: 12KB

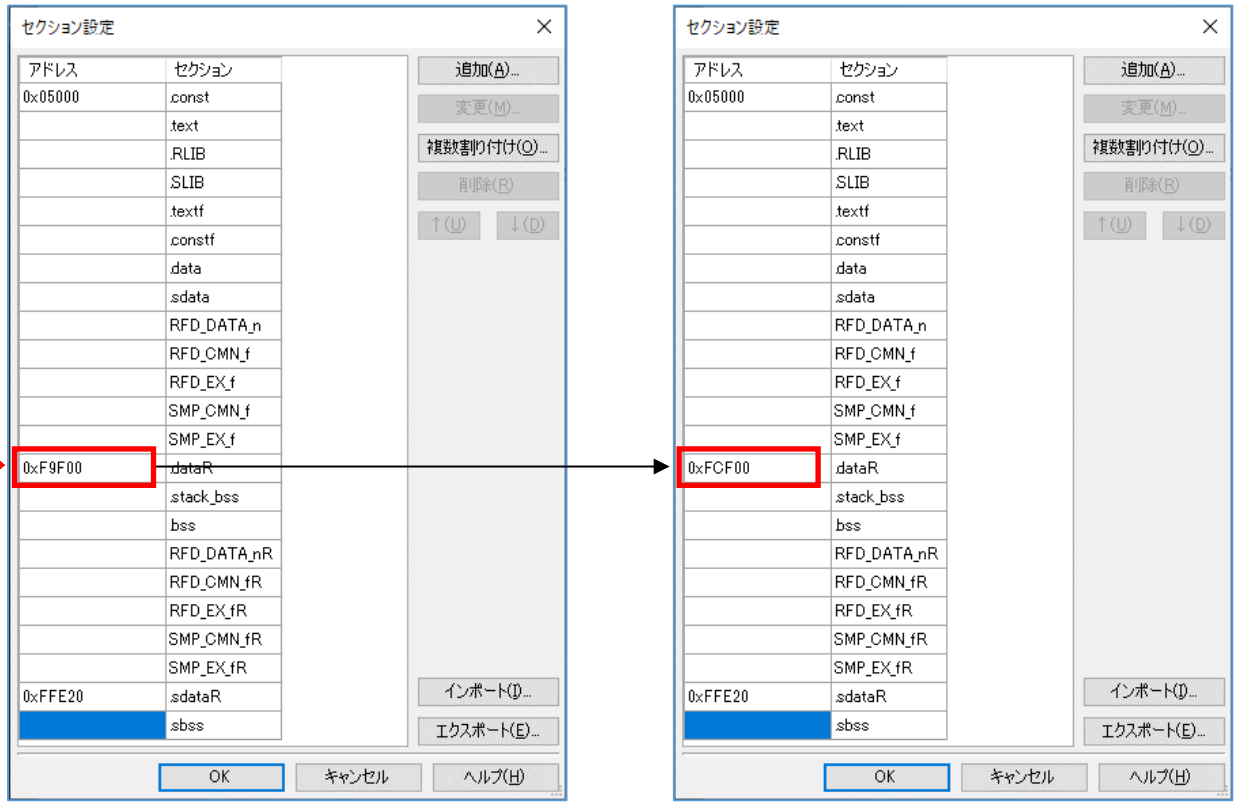


(3) エクストラ領域(EX)書き換え時

例）RL78/F24(R7F124FPJ)用設定 RAM: 24KB

例）RL78/F23(R7F123FxG)用設定 RAM: 12KB

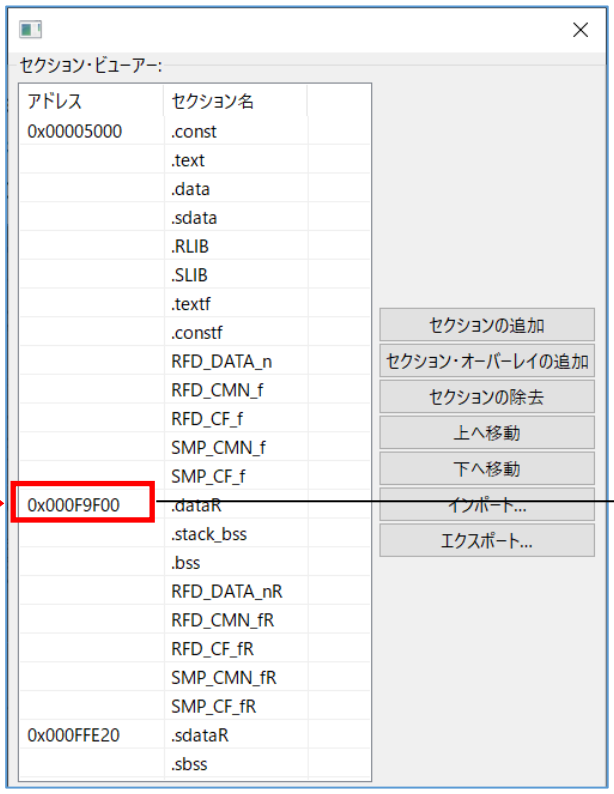
[R-1] →



・e²studio での CF/DF/EX 書き換え時のセクション設定（RAM の先頭アドレス）の変更箇所の例。

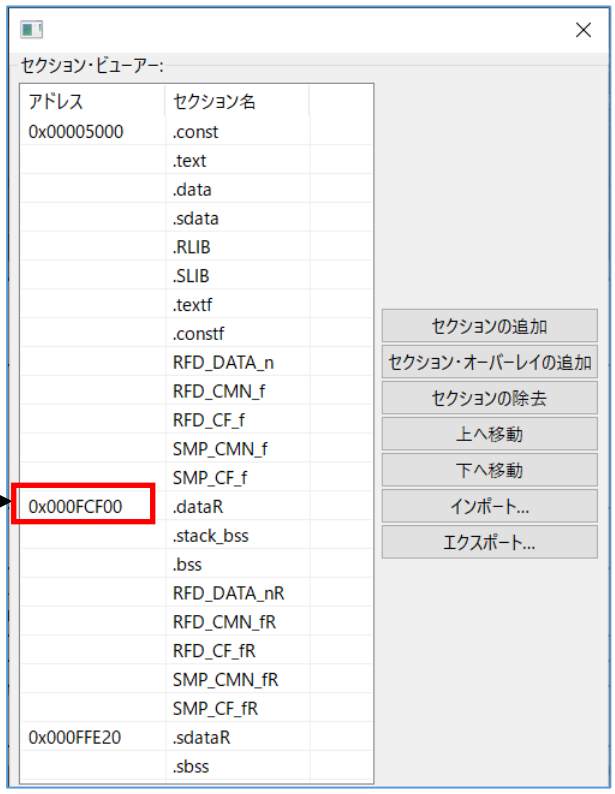
(1) コード・フラッシュ(CF)書き換え時

例）RL78/F24(R7F124FPJ)用設定 RAM: 24KB



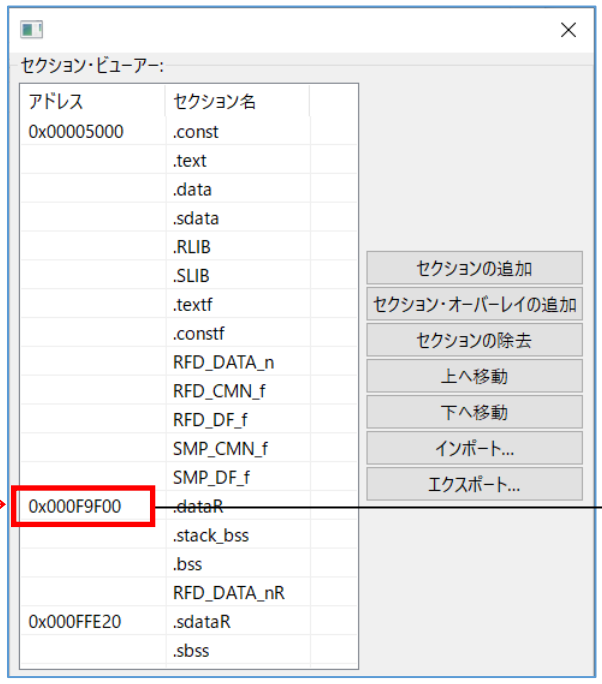
[R-1] →

例）RL78/F23(R7F123FxG)用設定 RAM: 12KB



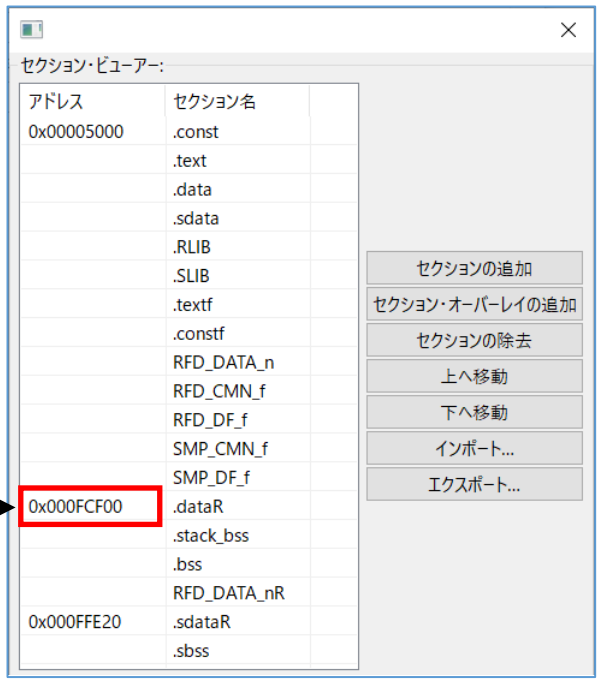
(2) データ・フラッシュ(DF)書き換え時

例）RL78/F24(R7F124FPJ)用設定 RAM: 24KB



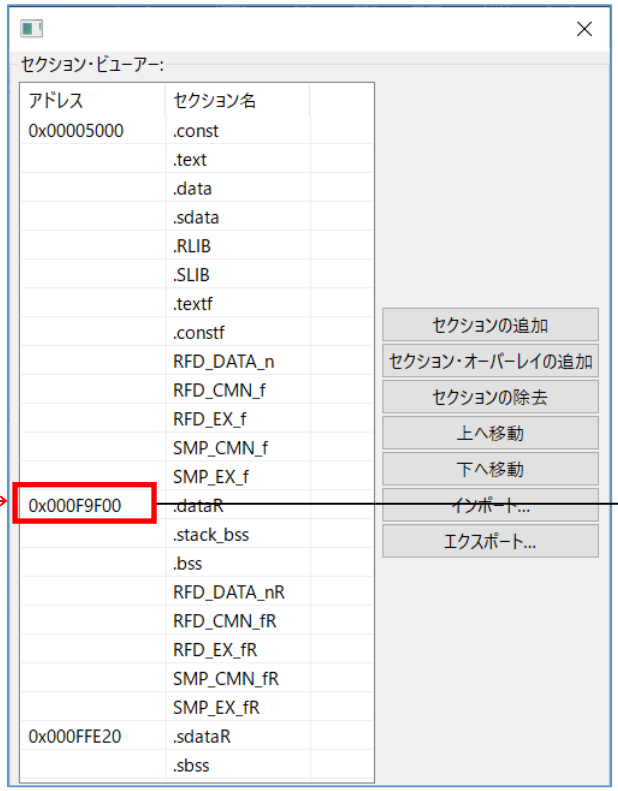
[R-1] →

例）RL78/F23(R7F123FxG)用設定 RAM: 12KB

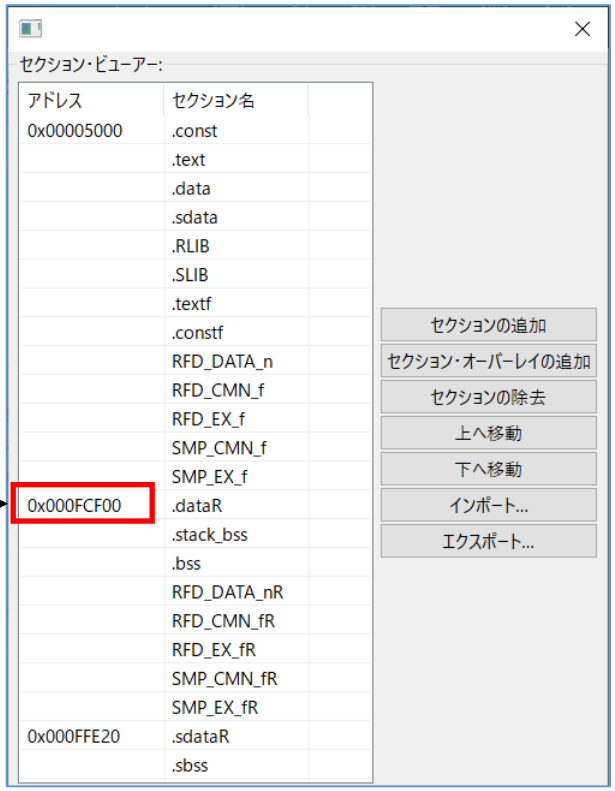


(3) エクストラ領域(EX)書き換え時

例）RL78/F24(R7F124FPJ)用設定 RAM: 24KB



例）RL78/F23(R7F123FxG)用設定 RAM: 12KB



6.3.1.4 デバッグ設定

RFD RL78 Type02 を RL78/F23 で使用する場合、デバッガ使用時のデバッグ・モニタ領域の範囲が異なります。

- デバッグ・モニタ領域の先頭アドレスは、ROM 領域の終了アドレスから"511byte(0x1FF)"を減算したアドレスを設定します。終了アドレスが"0x3FFFF"なら、"0x3FE00"を設定します。

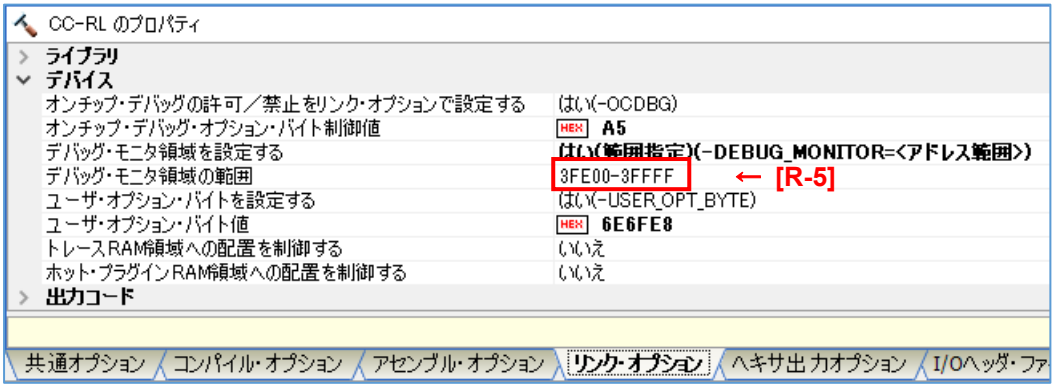
RL78/F24(R7F124FPJ)から RL78/F23(R7F123FMG)へ変更する場合を例として示します。

- RL78/F23 用にデバッグ・モニタ領域の範囲を[0x1FE00 - 0x1FFFF]に設定します。

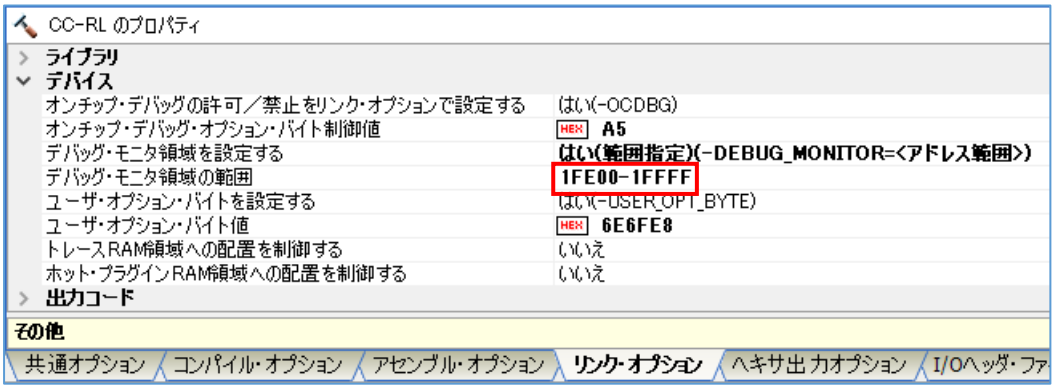
各製品のデバッグ・モニタ領域の先頭アドレスについては、対象 MCU リストの[R-5] 列をご確認ください。

・CS+でのデバッグ・モニタ領域の設定は、"リンク・オプション"タブで［デバイス］項目を選択します。

RL78/F24 用設定(ROM:256KB) R7F124FPJ の例



RL78/F23 用設定(ROM:128KB) R7F123FMG の例



・e²studio での OCD・モニタのメモリ領域の設定は、“Linker”から[デバイス]を選択します。

RL78/F24 用設定(ROM:256KB) R7F124FPJ の例

RL78/F23 用設定(ROM:128KB) R7F123FMG の例

6.3.2 IAR Embedded Workbench（IAR コンパイラ）を使用する場合の変更箇所

以降、IAR Embedded Workbench (IAR コンパイラ)を使用する場合の変更箇所と変更例を記載します。

6.3.2.1 デバイス用ヘッダ・ファイルの設定

RFD RL78 Type02 V1.00 で用意している main.c, low_level_init.c では、RL78/F24(R7F124FPJ)用のヘッダ・ファイルをインクルードしています。その他の RL78/F24 製品や RL78/F23 製品を使用する場合は、インクルードするヘッダ・ファイルを使用するデバイス用のヘッダ・ファイルに変更する必要があります。

ここでは、RL78/F23(R7F123FMG)を使用する場合の例を記載します。

対象ファイル名：main.c, low_level_init.c

- RL78/F24(R7F124FPJ)用:

```
< main.c >
  #include "ior7f124fpj.h"
< low_level_init.c >
  #include "ior7f124fpj.h"
  #include "ior7f124fpj_ext.h"
```

- RL78/F23(R7F123FMG)を使用する場合の例:

```
< main.c >
  #include "ior7f123fmg.h"
< low_level_init.c >
  #include "ior7f123fmg.h"
  #include "ior7f123fmg_ext.h"
```

※ 製品のデバイス型名については、対象 MCU リストの"Target MCU name" 列をご確認ください。

6.3.2.2 リンカ設定ファイルの設定

RFD RL78 Type02 で提供しているサンプル・プログラムでは、RL78/F24(R7F124FPJ)のセクション（ROM, RAM, Data flash の範囲）が設定されています。RL78/F24(R7F124FPJ)以外のデバイスを使用する場合は、セクション設定や、デバッガ使用時の TraceRAM 領域、デバッガ・モニタ領域の範囲、ホットプラグイン用 RAM 領域が異なるため、RFD RL78 Type02 V1.00 の RL78/F24 用に提供されているリンカ設定ファイル"sample_linker_file_xx.icf (xx = CF or DF or EX_FSW)"の内容を変更します。下記に変更箇所を赤文字で示していますので、対象 MCU リストを参照し、設定値を対象デバイス用に変更します。

対象ファイル名：sample_linker_file_xx.icf (xx = CF or DF or EX_FSW)

RL78/F24(R7F124FPJ)から RL78/F23(R7F123FMG)へ変更する場合を例として示します。

- ROM 領域を 128KB[0x00000 － 0x1FFFF]の範囲に設定します。
- RAM 領域が 12KB[0x0FCF00 － 0x0FFEFF]のため、開始アドレスを"0xFCF00"に変更します。
- Data flash 領域が 8KB[0x0F1000 － 0x0F2FFF]のため、終了アドレスを"0xF2FFF"に変更します。

(1) セクション設定

- ROM, RAM, Data Flash のサイズの変更箇所 xx = CF or EX_FSW の場合

例）RL78/F24(R7F124FPJ)用設定 ROM: 256KB, RAM: 24KB, Data Flash: 16KB

```
define region ROM_near = mem:[from 0x00132 to 0x0FFFF];   ← [R-2]
define region ROM_far = mem:[from 0x00132 to 0x0FFFF] | mem:[from 0x10000 to 0x1FFFF ]
                      | mem:[from 0x20000 to 0x2FFFF ] | mem:[from 0x30000 to 0x3FFFF ];  ← [R-2], [R-3] 注 1
define region ROM_huge = mem:[from 0x00132 to 0x3FFFF ];   ← [R-2] or [R-3] 注 2
define region SADDR = mem:[from 0xFFE20 to 0xFFEDF];
define region RAM_near = mem:[from 0xF9F00 to 0xFFE1F];   ← [R-1]
define region RAM_far = mem:[from 0xF9F00 to 0xFFE1F];    ← [R-1]
define region RAM_code = mem:[from 0xF9F00 to 0xFFE1F];   ← [R-1]
define region RAM_huge = mem:[from 0xF9F00 to 0xFFE1F];   ← [R-1]
define region VECTOR = mem:[from 0x00000 to 0x0007F];
define region CALLT = mem:[from 0x00080 to 0x000BF];
define region EEPROM = mem:[from 0xF1000 to 0xF4FFF ];    ← [R-4]
```

注 1　ROM サイズが 64KB よりも大きい場合、ROM サイズが増加するごとに記載を変更する必要があります。

注 2　対象 MCU リストの[R-3] にアドレス値が入力されている場合は[R-3] の値を使用し、"ー"の場合は[R-2] の値を設定してください。

例）RL78/F23(R7F123FMG)用設定 ROM: 128KB, RAM: 12KB, Data Flash: 8KB

```
define region ROM_near = mem:[from 0x00132 to 0x0FFFF];
define region ROM_far = mem:[from 0x00132 to 0x0FFFF] | mem:[from 0x10000 to 0x1FFFF ];
define region ROM_huge = mem:[from 0x00132 to 0x1FFFF ];
define region SADDR = mem:[from 0xFFE20 to 0xFFEDF];
define region RAM_near = mem:[from 0xFCF00 to 0xFFE1F];
define region RAM_far = mem:[from 0xFCF00 to 0xFFE1F];
define region RAM_code = mem:[from 0xFCF00 to 0xFFE1F];
define region RAM_huge = mem:[from 0xFCF00 to 0xFFE1F];
define region VECTOR = mem:[from 0x00000 to 0x0007F];
define region CALLT = mem:[from 0x00080 to 0x000BF];
define region EEPROM = mem:[from 0xF1000 to 0xF2FFF ];
```

- ROM, RAM, Data Flash のサイズの変更箇所　xx = DF の場合

例）RL78/F24(R7F124FPJ)用設定 ROM: 256KB, RAM: 24KB, Data Flash: 16KB

```
define region ROM_near = mem:[from 0x00132 to 0x0FFFF];   ← [R-2]
define region ROM_far = mem:[from 0x00132 to 0x0FFFF] | mem:[from 0x10000 to 0x1FFFF ]
                      | mem:[from 0x20000 to 0x2FFFF ] | mem:[from 0x30000 to 0x3FFFF ];  ← [R-2], [R-3] 注 1
define region ROM_huge = mem:[from 0x00132 to 0x3FFFF ];   ← [R-2] or [R-3] 注 2
define region SADDR = mem:[from 0xFFE20 to 0xFFEDF];
define region RAM_near = mem:[from 0xF9F00 to 0xFFE1F];   ← [R-1]
define region RAM_far = mem:[from 0xF9F00 to 0xFFE1F];    ← [R-1]
define region RAM_huge = mem:[from 0xF9F00 to 0xFFE1F];   ← [R-1]
define region VECTOR = mem:[from 0x00000 to 0x0007F];
define region CALLT = mem:[from 0x00080 to 0x000BF];
define region EEPROM = mem:[from 0xF1000 to 0xF4FFF ];    ← [R-4]
```

注 1　ROM サイズが 64KB よりも大きい場合、ROM サイズが増加するごとに記載を変更する必要があります。

注 2　対象 MCU リストの[R-3] にアドレス値が入力されている場合は[R-3] の値を使用し、"ー"の場合は[R-2] の値を設定してください。

例）RL78/F23(R7F123FMG)用設定 ROM: 128KB, RAM: 12KB, Data Flash: 8KB

```
define region ROM_near = mem:[from 0x00132 to 0x0FFFF];
define region ROM_far = mem:[from 0x00132 to 0x0FFFF] | mem:[from 0x10000 to 0x1FFFF ];
define region ROM_huge = mem:[from 0x00132 to 0x1FFFF ];
define region SADDR = mem:[from 0xFFE20 to 0xFFEDF];
define region RAM_near = mem:[from 0xFCF00 to 0xFFE1F];
define region RAM_far = mem:[from 0xFCF00 to 0xFFE1F];
define region RAM_huge = mem:[from 0xFCF00 to 0xFFE1F];
define region VECTOR = mem:[from 0x00000 to 0x0007F];
define region CALLT = mem:[from 0x00080 to 0x000BF];
define region EEPROM = mem:[from 0xF1000 to 0xF2FFF ];
```

(2) デバッグ設定

- デバッグ・モニタ領域の先頭アドレスは、ROM 領域の終了アドレスから"511byte(0x1FF)"を減算したアドレスを設定します。終了アドレスが"0x3FFFF"なら、"0x3FE00"を設定します。
- TraceRAM 領域の先頭アドレスは、RAM 領域の先頭アドレスに"1KB(0x400)"を加算したアドレスを設定します。 先頭アドレスが"0xF9F00"なら、"0xFA300"を設定します。
- ホットプラグイン用 RAM 領域の先頭アドレスは、RAM 領域の先頭アドレスに"0x600"を加算したアドレスを設定します。 先頭アドレスが"0xF9F00"なら、"0xFA500"を設定します。

RL78/F24(R7F124FPJ)から RL78/F23(R7F123FMG)へ変更する場合を例として示します。

- デバッグ・モニタ領域の範囲を[from 0x1FE00 size 0x0200]に設定します。
- TraceRAM 領域の範囲を[from 0xFD300 size 0x0200]に設定します。
- ホットプラグイン用 RAM 領域の範囲を[from 0xFD500 size 0x0030]に設定します。

- デバッガ使用時の TraceRAM 領域、デバッグ・モニタ領域、ホットプラグイン RAM 領域の変更箇所

例）RL78/F24(R7F124FPJ)用設定 ROM: 256KB, RAM: 24KB

```
if (isdefinedsymbol(__RESERVE_OCD_ROM))
{
        if (__RESERVE_OCD_ROM == 1)
        {
        reserve region "OCD ROM area" = mem:[from 0x3FE00 size 0x0200];  ← [R-5]
        }
}
（一部省略）
if (isdefinedsymbol(__RESERVE_OCD_TRACE_RAM))
{
        if (__RESERVE_OCD_TRACE_RAM == 1)
        {
        reserve region "OCD Trace RAM" = mem:[from 0xFA300 size 0x0200]; ← [R-6]
        }
}
（一部省略）
if (isdefinedsymbol(__RESERVE_HOTPLUGIN_RAM))
{
   if (__RESERVE_HOTPLUGIN_RAM == 1)
   {
      reserve region "Hot Plugin RAM" = mem:[from 0xFA500 size 0x0030];    ← [R-7]
   }
}
```

例）RL78/F23(R7F123FMG)用設定 ROM: 128KB, RAM: 12KB

```
if (isdefinedsymbol(__RESERVE_OCD_ROM))
{
        if (__RESERVE_OCD_ROM == 1)
        {
        reserve region "OCD ROM area" = mem:[from 0x1FE00 size 0x0200];
        }
}
（一部省略）
if (isdefinedsymbol(__RESERVE_OCD_TRACE_RAM))
{
        if (__RESERVE_OCD_TRACE_RAM == 1)
        {
        reserve region "OCD Trace RAM" = mem:[from 0xFD300 size 0x0200];
        }
}
（一部省略）
if (isdefinedsymbol(__RESERVE_HOTPLUGIN_RAM))
{
   if (__RESERVE_HOTPLUGIN_RAM == 1)
   {
      reserve region "Hot Plugin RAM" = mem:[from 0xFD500 size 0x0030];
   }
}
```

(3) RAM の開始アドレス設定

RAM 領域の開始アドレスを設定します。

"sample_linker_file_xx.icf (xx = CF or DF or EX_FSW)"は、RL78/F24(R7F124FPJ)を RAM サイズ 24KB で使用する設定のため、RAM サイズを 24KB 以外で使用する場合は設定を変更する必要があります。

ここでは、RL78/F23(R7F123FMG)を RAM サイズ 12KB で使用するための設定変更例を記載します、

RL78/F23(R7F123FMG)を RAM サイズ 12KB で使用するためには、RAM 開始アドレス設定レジスタ(RAMSAR)の値を"0x9F"から"0xCF"に変更します。

RAM 開始アドレス設定レジスタ(RAMSAR)の詳細は、デバイスのハードウェアマニュアルでご確認ください。

RL78/F24 用設定(RAM:24KB) R7F124FPJ の例

```
define exported symbol _RAMSAR_ADDR = 0xF0076;
if (!isdefinedsymbol(__RAMSAR_VAL))
{
  define exported symbol _RAMSAR_VAL = 0x9F;
}
else
{
  define exported symbol _RAMSAR_VAL = __RAMSAR_VAL;
}
```

RL78/F23 用設定(RAM:12KB) R7F123FMG の例

```
define exported symbol _RAMSAR_ADDR = 0xF0076;
if (!isdefinedsymbol(__RAMSAR_VAL))
{
  define exported symbol _RAMSAR_VAL = 0xCF;
}
else
{
  define exported symbol _RAMSAR_VAL = __RAMSAR_VAL;
}
```

6.3.3 サンプル・プログラムの変更箇所（コンパイラ共通）

6.3.3.1 Extra 領域［FSW 範囲］書き換えサンプル・プログラムの変更

RFD RL78 Type02 のサンプル・プログラムで対象としている RL78/F24 と RL78/F23 では、コード・フラッシュ・メモリのブロック数が異なります。そのため、Extra 領域用"sample_config.h" の FSW 範囲エンド・ブロック用マクロ[END_BLOCK]の設定値を変更します。[END_BLOCK]は、FSW 範囲の"エンド・ブロック番号+1"を示しています([R-8])。また、コメント内では、FSW 範囲のエンド・ブロック番号([R-8]-1）と、FSW 範囲の"エンド・ブロック番号+1"([R-8]) を示しています。

例）RL78/F24(R7F124FPJ)の場合、[R-8] が 256 で、[R-8] -1 は 255 です。

対象ファイル名：sample_config.h

ファイルパス：\sample\RL78_F24\EX_FSW\IAR\include

例）RL78/F24(R7F124FPJ)用設定 ROM: 256KB

```
/***********************************************************************************************************************
   User configurable parameters
***********************************************************************************************************************/

/**** CPU frequency (MHz) ****/
/* It must be rounded up digits after the decimal point to form an integer (MHz). */
#define CPU_FREQUENCY (40u)

/**** Block numbers for FSW ****/
/* Start block number for FSW */
#define START_BLOCK     (0u)
/* End block number for FSW */
/* It must be the block number points one block past the end of range for FSW. */
/* If the block number 255 is the end of range for FSW, please specify 256. */   ← [R-8]-1, [R-8]
#define END_BLOCK       (256u)   ← [R-8]
```

例）RL78/F23(R7F123FMG)用設定 ROM: 128KB

```
/***********************************************************************************************************************
   User configurable parameters
***********************************************************************************************************************/

/**** CPU frequency (MHz) ****/
/* It must be rounded up digits after the decimal point to form an integer (MHz). */
#define CPU_FREQUENCY (40u)

/**** Block numbers for FSW ****/
/* Start block number for FSW */
#define START_BLOCK     (0u)
/* End block number for FSW */
/* It must be the block number points one block past the end of range for FSW. */
/* If the block number 127 is the end of range for FSW, please specify 128. */
#define END_BLOCK       (128u)
```

7 改定記録

7.1 本版で改定された主な箇所

Rev.	発行日	改定内容	
		Page	概要
1.00	2022.7.6	-	新規作成
1.10	2022.12.28	-	RL78/F23 追加サポート

Renesas Flash Driver RL78 Type02 ユーザーズマニュアル

発行年月日　2022年12月28日　　Rev.1.10

発行　　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）

Renesas Flash Driver
RL78 Type02

RENESAS

ルネサス エレクトロニクス株式会社

R01UT5009JJ0110