User's Manual

32

RH850 ファミリ

コードフラッシュライブラリ Type01

対象インストーラ名：RENESAS_FCL_RH850_T01_Vx.xx※

※ x.xx ： 2.01以上

32 ビット・シングルチップ・マイクロコントローラ

ルネサス エレクトロニクス
www.renesas.com

Rev.1.02　2019.05

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器・システムの設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品、本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。
   標準水準：　コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット等
   高品質水準：輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、金融端末基幹システム、各種安全制御装置等
   当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
6. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
8. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
9. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
10. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
12. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

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

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

(Rev.4.0-1　2017.11)

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

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の商標です。すべての商標および登録商標は、それぞれの所有者に帰属します。

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、RH850ファミリのコードフラッシュライブラリ Type01の機能を理解し、それを用いたアプリケーションシステムを設計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、RH850ファミリのコードフラッシュの書き換えを行うために使用するRH850 コードフラッシュライブラリ Type01の使用方法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。

・はじめに
・アーキテクチャ
・機能仕様
・ユーザインタフェース（API）
・FCLの設定方法と使用方法
・注意事項

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の一般知識を必要とします。

□一通りの機能を理解しようとするとき
→目次に従って読んでください。
□関数の機能の詳細を知りたいとき
→このユーザーズ·マニュアルの**第4章　ユーザインタフェース**を参照してください。

なお、初版以降において本文欄外の★印は、本版で改訂された主な箇所を示しています。

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…××××
	16進数…××××Hまたは0x××××

目次

第 1 章 はじめに

本ユーザマニュアルでは、RH850 ファミリ用コードフラッシュライブラリ Type 01（以降 FCL と呼称する）の機能、アプリケーションプログラミング・インタフェース（API）について説明します。

備考：

- 本 FCL の対象デバイス以外には本 FCL を使用しないでください。あらゆる動作不良につながる恐れがあります。本 FCL はソースコードで供給されます。意図しない動作不良やプログラミング不良を生じる恐れがあるので、ご採用の際には細心の注意を払い、十分に動作確認してご使用ください。
- 本 FCL は、GHS 社製コンパイラ用 FCL とルネサスエレクトロニクス社製コンパイラ用 FCL が含まれます。コンパイラとアセンブラの機能は多様で、特にアセンブラファイルは環境によって異なります。したがって、本 FCL とアプリケーションプログラムは、適切な環境を選択できるようにインストーラツールを使用して提供されます。
- 本 FCL は、デバイス依存のアプリケーション･プログラムと一緒に提供されます。
- 本 FCL は無償製品であり、サンプルプログラムとしてソースコード形態で提供しています。
- 本 FCL、本ユーザマニュアルは、常に最新バージョンのご使用を推奨します。なお、対応デバイスにつきましては本 FCL パッケージに含まれている support.txt にてご確認ください。
- 本ユーザマニュアルのすべての章をよくお読みください。本 FCL の誤使用に起因する誤動作を回避するために、本ユーザーズマニュアルが定める手順と推奨事項に従ってください。また、本ユーザーズマニュアルは、必ず、本 FCL パッケージに添付されているリリースノート、および対象デバイスのユーザーズマニュアルと合わせてご使用ください。

1.1 フラッシュメモリの基本構造

RH850 ファミリの多くには、コードフラッシュに加えてデータフラッシュという独立したフラッシュメモリ領域が搭載されています。このフラッシュメモリ領域はデータ格納用に設計されています。命令実行（コードフェッチ）に用いることはできません。

・デバイスによってはコードフラッシュに関する仕様（1ブロックサイズ、読み出し単位など）が異なる場合や注意事項がある場合があります。

≪RH850/F1L コードフラッシュメモリマッピング仕様の一例≫

ユーザ領域は8 K バイトまたは32 K バイトのブロックに分割されており、各ブロック単位で消去可能です。また、32K バイトの拡張ユーザ領域を1 ブロック搭載しています。ユーザ領域と拡張ユーザ領域は、ユーザプログラムの格納領域として利用可能です。

・デバイスによってはFCLが書き換え可能なコードフラッシュの領域が異なる場合があります。

≪例 RH850/F1Lの場合≫

拡張ユーザ領域は FCL で書き換えが可能です。

1.1.1 BGO（Back Ground Operation）機能

データフラッシュ、またはコードフラッシュに対してコマンド実行中にコードフラッシュの実行が継続可能な機能です。書き換え対象がコードフラッシュ（特定領域）で、読み出し対象がコードフラッシュ（特定領域）に対応しているデバイスの場合、コードフラッシュに対しコマンド実行中もコードフラッシュ領域（特定領域）にアクセスすることが可能です。よって、デバイスによってはコードフラッシュ上でFCLを実行することが可能です。

書き換え対象領域と読み出し対象領域の組み合わせなどの詳細に関しましては対象デバイスのユーザーズマニュアルで必ずご確認ください。

なお、BGO 機能に対応していないデバイスの場合、コマンド実行中は、コードフラッシュにアクセスすることができません。コマンド実行中にコードフラッシュにアクセスした場合、システムがハングアップします。

注意：コードフラッシュとデータフラッシュを同時に書き換えることはできません。

1.1.2 フラッシュの動作単位

RH850 ファミリのコードフラッシュはブロック構造になっており、このブロックが FCL における消去操作単位です。消去は必ずブロック単位で行うことになります。また、書き込みなどは開始アドレスとサイズを指定して実行します。なお、対象デバイスのコードフラッシュのハードウェア仕様（コードフラッシュサイズ、1 ブロックサイズ、バンクなど）は、必ず対象デバイスのユーザーズマニュアルでご確認ください。

1) データは、ブロック間の境界に合わせて 256 バイト単位で書き込むことができます。
2) 消去直後のコードフラッシュを読み出すと、ECC エラーが発生するため、読み出さないでください。
3) RH850 デバイスのコードフラッシュは、次の 2 つのユーザ領域に分割されています。
   - ユーザ領域は、ユーザプログラムのプログラミングを行うコードフラッシュメモリで、0x00 を先頭として番号が付けられた多数のブロックで構成されています。これらのブロックの書き込みアドレスは 0x00000000 から始まり、256 ずつ増加します。
   - 拡張ユーザ領域は、ブートローダプログラムなどの特別なアプリケーション向けに設計されています。この領域の先頭のブロック番号は 0x80000000 です。この領域の書き込みアドレスは 0x01000000 から始まり、256 ずつ増加します。なお、デバイスによっては FCL が書き換え可能なコードフラッシュの領域が異なる場合があります。詳細に関しましては、対象デバイスのユーザーズマニュアルでご確認ください。

第 2 章 アーキテクチャ

この章では、ユーザが FCL を使用してコードフラッシュの書き換えを行う上で、必要な FCL のアーキテクチャについて説明します。

FCL は､コードフラッシュを操作するハードウェアにアクセスする為のインタフェースです。「図 1　メモリ構成例 RH850/F1L」の矢印が操作の流れとなります。ユーザアプリケーションから FCL 関数を実行することにより、コードフラッシュを制御するハードウェアが動作し、コードフラッシュを操作します。

- ユーザアプリケーション：

  このブロックはユーザのアプリケーション（起動プログラムがある場合はそれを含めて）を表します。

- フラッシュハードウェア：

  この機能ブロックは､FCL で制御するフラッシュプログラミングハードウェア(フラッシュシーケンサ)を表します。

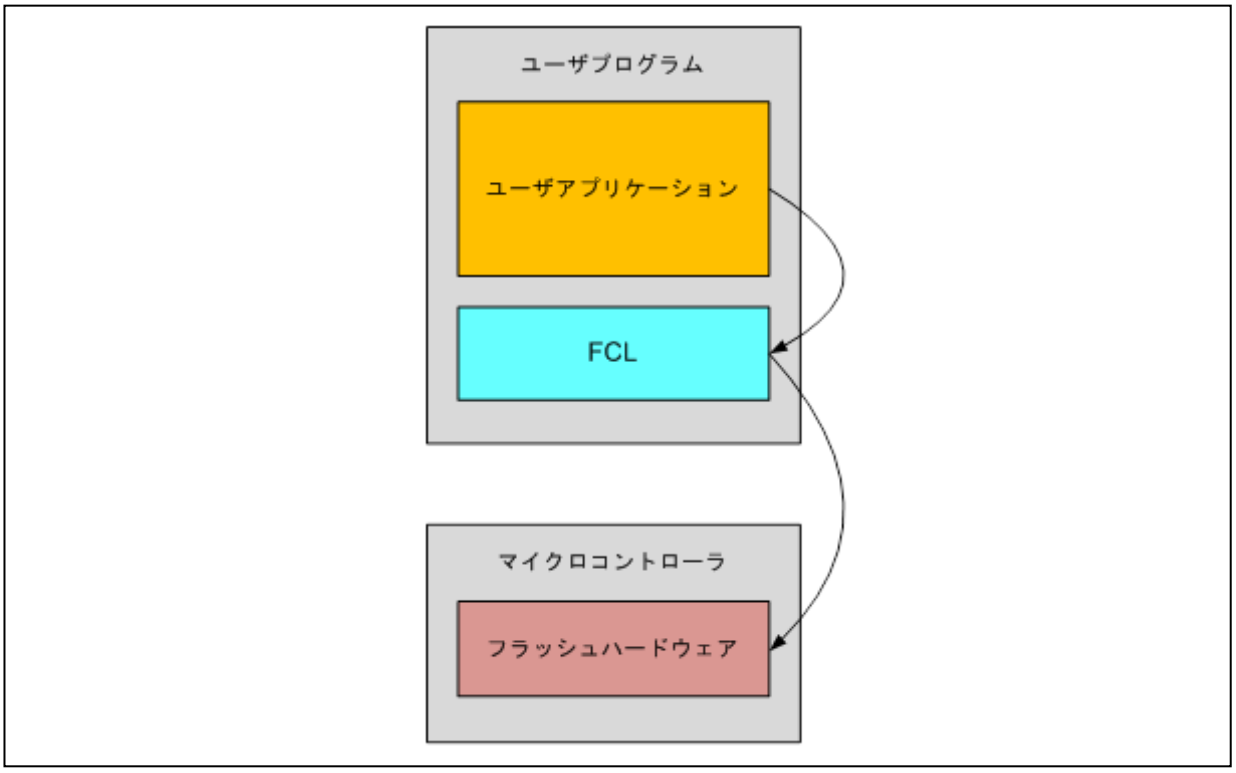

図 1　メモリ構成例 RH850/F1L

2.1 FCLの説明

本 FCL は次の形式で提供されています。

- ソースコード（インストーラパッケージ内にサンプルアプリケーションと同梱）

第 3 章 FCLの機能仕様

3.1 内蔵RAMでのコード実行

初期状態ではセルフプログラミングを行うユーザアプリケーションと FCL はコードフラッシュ内に配置されます。コマンド実行中はコードフラッシュにアクセスできない為、ユーザアプリケーションと FCL の一部を内蔵 RAM にコピーして実行する必要があります。

なお、本 FCL の対象デバイスにおいて、RAM に対して ECC によるエラー検出・訂正機能を有効にしてアクセスする場合、使用する RAM を初期化してください。

内蔵 RAM にコピーが必要なのは、コマンド実行を行う部分だけで、全てをコピーする必要はありませんが、コピーする箇所は、慎重に選択してください。FCL 実行中にコピーしなかったコードへのアクセス、または、割り込み、例外、ウォッチドッグリセットなどに起因するコードフラッシュへのアクセスが無いよう、十分にご注意ください。

必要な部分を使用可能な内蔵 RAM にコピーする方法には次の 3 種類があります。

なお、インターナルモードの場合、必ず R_FCL_CopySections 関数を使用してコピーしてください。

1. **ROM 化**

   コードをコピー先のアドレスにリンクさせます。コンパイラ固有のルーチン（CC-RH コンパイラの場合、INITSCT_RH 関数）を呼び出すことで、指定したコードがコードフラッシュ内から内蔵 RAM にコピーされます。 なお、RAM 上で C 言語デバッグする為には、ROM 化が必要です。

2. **R_FCL_CopySections**

   FCL は、FCL で必要なすべてのセクションを、使用する定義済みアドレスにコピーする R_FCL_CopySections 関数を提供しています。

3. **ユーザ固有**

   ユーザが作成したコピー関数などを使用する場合は、ユーザがセクションを正しく配置する必要があります。

設定したモード（インターナルモードまたはユーザモード。「3.2 動作モード」参照）によっては、以下に示すリンカセクションを内蔵 RAM にコピーする必要があります。

表 1　内蔵RAMにコピーするリンカセクション

プリコンパイル設定	内蔵 RAM にコピーするセクション
R_FCL_HANDLER_CALL_USER	R_FCL_CODE_RAM_USRINT R_FCL_CODE_RAM_USR R_FCL_CODE_RAM R_FCL_CODE_ROMRAM
R_FCL_HANDLER_CALL_INTERNAL	R_FCL_CODE_RAM_USRINT R_FCL_CODE_RAM_USR R_FCL_CODE_RAM

リンカセクションの詳細については、「5.3リンカセクション」を参照してください。

3.2 動作モード

セルフプログラミングには、以下に示す 2 つのモードがあります。

ユーザモード：

本 FCL の大部分の関数は、セルフプログラミング中に実行する FCL を制御するユーザ関数やその他のユーザコードと共に内蔵 RAM で実行されます。FCL 関数はコマンド動作を開始した後、ユーザコードに戻りますので、セルフプログラミング中にユーザアプリケーションを実行することができます。この間、コマンド動作はバックグラウンドで実行され、ユーザは R_FCL_Handler 関数を使って動作が完了するまでポーリングする必要があります（「4.4.2.3 R_FCL_Handler」参照）。割り込みルーチンとユーザコードは、関連する関数がすべて内蔵 RAM 内にあれば実行できます。

このモードを有効にするには、ユーザモードを使用するように本 FCL を設定しなければなりません（「4.1　プリコンパイル設定」参照）。

ユーザモード時の動作例を次の図に示します。

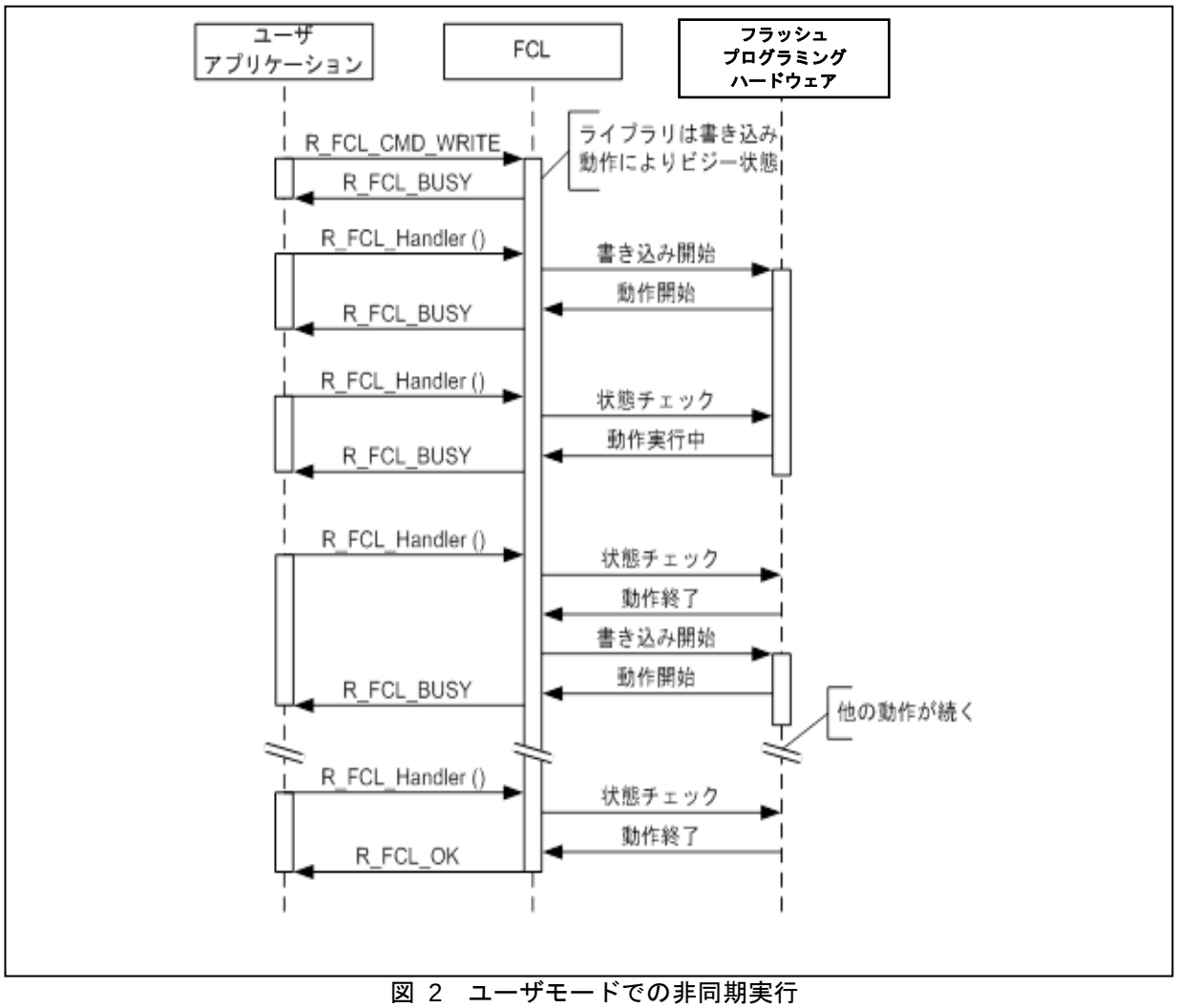

図 2　ユーザモードでの非同期実行

R_FCL_Handler 関数を繰り返し実行することによってコマンド実行を進行させ、最終的に終了を確認します。

インターナルモード：

FCL の一部だけを内蔵 RAM で実行し、残りをコードフラッシュで実行します。セルフプログラミング中は一連のコードフラッシュ制御動作が終了するまでユーザアプリケーションに戻りませんので通常のユーザコードは実行できません。したがって、セルフプログラミング中に実行できるのは割り込みだけです。

このモードを有効にするには、インターナルモードを使用するように本 FCL を設定しなければなりません（「4.1　プリコンパイル設定」参照）。

3.3 リクエスト/レスポンス型アーキテクチャ

FCL はコマンドの起動にリクエスト/レスポンス型アーキテクチャを利用しています。各コマンドは関数で実行し、リクエスト構造体経由でコマンドやデータサイズなどを FCL へ受け渡します。FCL はリクエスト構造体の内容を解釈し、その構造体のメンバの整合性をチェックして実行を開始します。また、逆に FCL の状態、エラー情報などをリクエスト構造体経由で取得します。

リクエスト構造体メンバ status_enu は、本 FCL の動作モードに応じて更新のされ方が異なります。インターナルモードでは、R_FCL_Execute 関数から処理が戻ると、その結果により状態が更新されます。一方、ユーザモードでは状態が R_FCL_BUSY の場合、直ちに R_FCL_Execute 関数から処理が戻ります。リクエストの状態が"R_FCL_BUSY"である限り R_FCL_Handler を呼び出して、コマンドを完了させます。ユーザアプリケーションは、フラッシュのリソースを使用しない限り、残りの時間他の動作を自由に実行できます。

次の図に示すように、リクエスト構造体は FCL 動作の中心に位置しています。

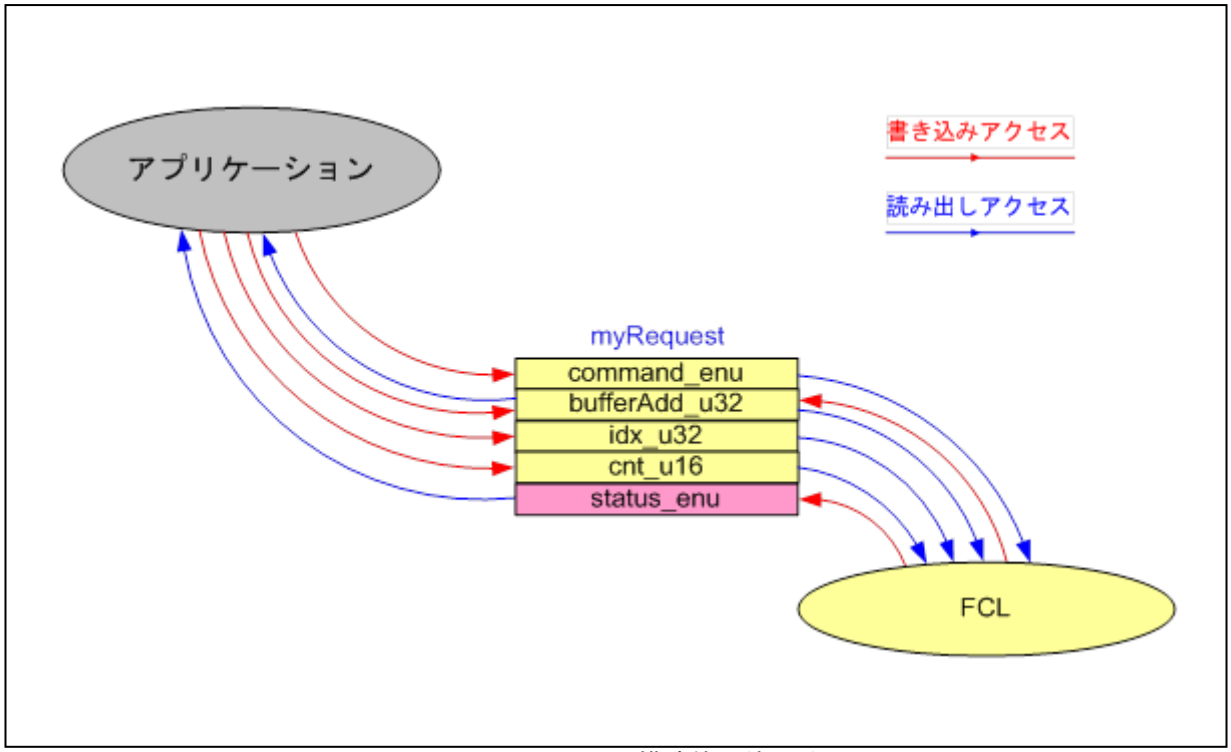

図 3　リクエスト構造体の使用法

3.4 サスペンド/レジュームメカニズム

消去または書き込み動作は長く続くこともあり、終了を待たず、途中で中断したい場合があります。FCL には、その動作をサスペンドして、後からレジュームするオプション機能があります。

このサスペンドは R_FCL_CMD_WRITE コマンドと R_FCL_CMD_ERASE コマンドのみに使用することが可能です。コマンドのサスペンド後は、別のコマンドを起動したり、ユーザアプリケーションで FCL と無関係な別の動作を実行したりできます。

サスペンド/レジューム機能はユーザモードでのみ使用可能です。

消去コマンドをサスペンドし、書き込みを行ってから消去コマンドをレジュームする方法を図 4 に示します。

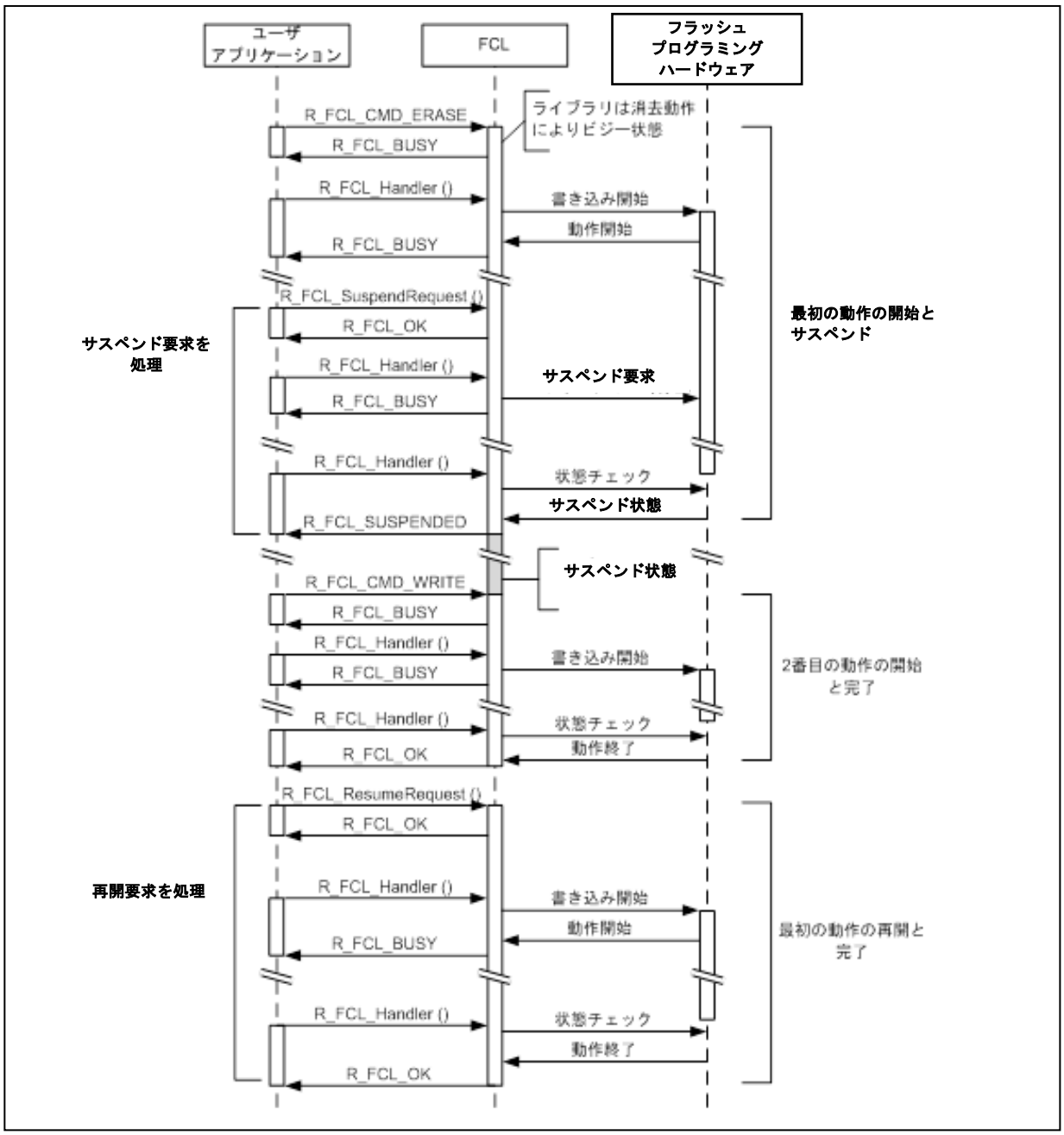

図 4　サスペンド/レジュームフロー例

消去中にサスペンドし、レジュームした場合、消去が再度実行されますが、これは中断された消去の再開の為、消去回数は追加されません。

サスペンドに関する注意事項がありますので「第 6 章　注意事項」をご参照ください。

3.5 キャンセルメカニズム

消去または書き込み動作は長く続くこともあり、終了を待たず、中止したい場合があります。FCL には、その動作をキャンセルする機能があります。 キャンセル機能はユーザモードでのみ使用可能です。消去コマンドをキャンセルする方法を次の図に示します。

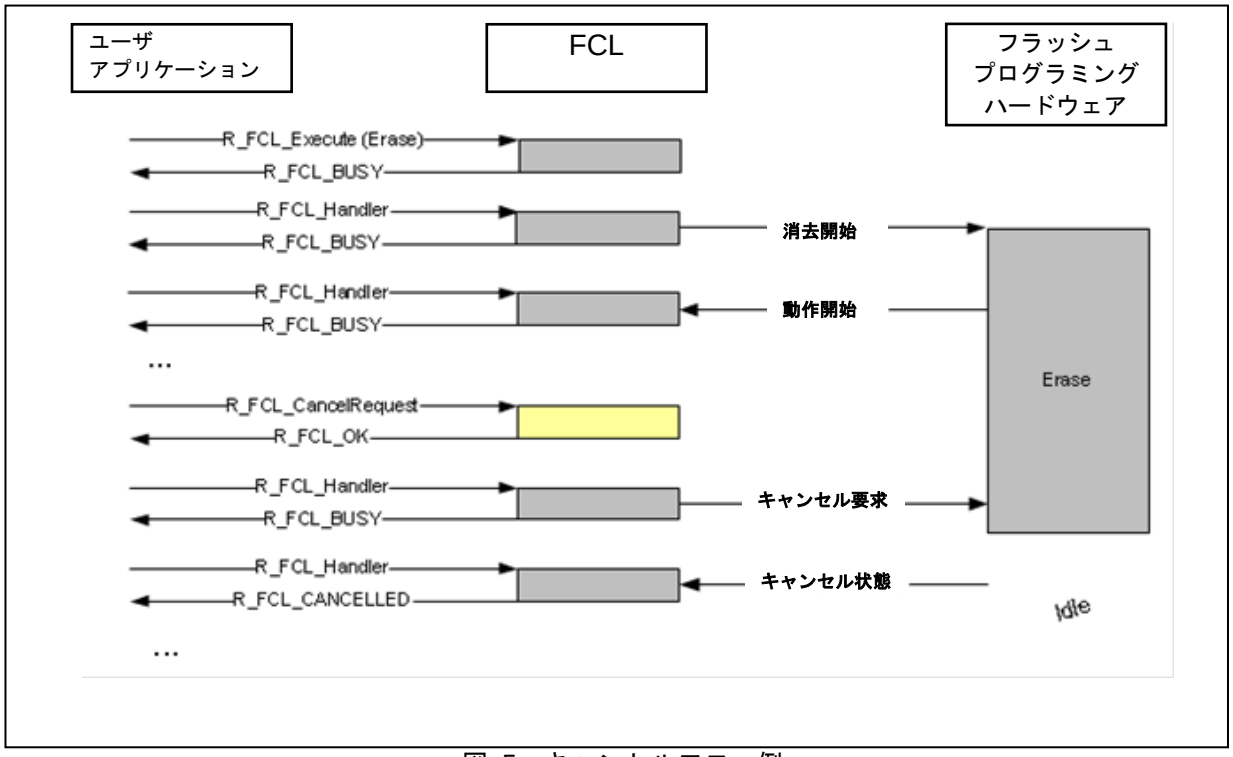

図 5　キャンセルフロー例

FCL サスペンド状態に対して R_FCL_CancelRequest 関数を実行した場合、FCL サスペンド状態の前に実行していた R_FCL_CMD_ERASE コマンド、R_FCL_CMD_WRITE コマンドもキャンセルされます。

3.6 タイムアウト処理

V2.11 以降、デバイスのレジスタを監視しているループ処理にタイムアウト機能を追加しました。

第 4 章 ユーザインタフェース（API）

4.1　プリコンパイル設定

FCL のプリコンパイル設定は fcl_cfg.h ファイル内にあります。そのヘッダファイル内のすべてのオプションを確認し、設定する必要があります。これらのオプションを次の表に記載します。

表 2　プリコンパイルオプション　(1/2)

オプション	説明
R_FCL_COMMAND_EXECUTION_MODE	状態チェックを FCL 内部で実行するか、またはユーザが実行して状態チェックの間、ユーザコードを実行できるようにするかを定義します。設定できる値は次のとおりです。 R_FCL_HANDLER_CALL_INTERNAL R_FCL_HANDLER_CALL_USER 詳細については下記をご覧ください。
R_FCL_SUPPORT_LOCKBIT	以下のコマンドを有効または無効にします。 R_FCL_CMD_GET_LOCKBIT R_FCL_CMD_SET_LOCKBIT R_FCL_CMD_ENABLE_LOCKBITS R_FCL_CMD_DISABLE_LOCKBITS
R_FCL_SUPPORT_OTP	以下のコマンドを有効または無効にします。 R_FCL_CMD_GET_OTP R_FCL_CMD_SET_OTP
R_FCL_SUPPORT_DEVICENAME	以下のコマンドを有効または無効にします。 R_FCL_CMD_GET_DEVICE_NAME
R_FCL_SUPPORT_BLOCKCNT	以下のコマンドを有効または無効にします。 R_FCL_CMD_GET_BLOCK_CNT
R_FCL_SUPPORT_BLOCKENDADDR	以下のコマンドを有効または無効にします。 R_FCL_CMD_GET_BLOCK_END_ADDR
R_FCL_SUPPORT_OPB	以下のコマンドを有効または無効にします。 R_FCL_CMD_GET_OPB R_FCL_CMD_SET_OPB
R_FCL_SUPPORT_ID	以下のコマンドを有効または無効にします。 R_FCL_CMD_GET_ID R_FCL_CMD_SET_ID
R_FCL_SUPPORT_RESETVECTOR	以下のコマンドを有効または無効にします。 R_FCL_CMD_GET_RESET_VECTOR R_FCL_CMD_SET_RESET_VECTOR
R_FCL_SUPPORT_SECURITYFLAGS	以下のコマンドを有効または無効にします。 R_FCL_CMD_SET_READ_PROTECT_FLAG R_FCL_CMD_SET_WRITE_PROTECT_FLAG R_FCL_CMD_SET_ERASE_PROTECT_FLAG R_FCL_CMD_SET_SERIAL_PROG_DISABLED R_FCL_CMD_SET_SERIAL_ID_ENABLED

表 2　プリコンパイルオプション　　(2/2)

オプション	説明
R_FCL_NO_BFA_SWITCH （対応バージョン：V2.11 以降）	下記の仕様を無効に設定し、RH850/F1K,F1KM,F1KH 専用にビルドします。 ・ユーザ領域と FCU ファームウェア格納領域の切り替え。 ・FCU ファームウェア転送。 本オプションを有効にした場合、RH850/F1K,F1KM,F1KH 以外では使用しないでください。なお、本オプションを有効にした場合、R_FCL_MIRROR_FCU_COPY オプションと R_FCL_NO_FCU_COPY オプションは有効にしないでください。
R_FCL_MIRROR_FCU_COPY （対応バージョン：V2.12 以降）	下記の仕様を無効にします。 ・ユーザ領域と FCU ファームウェア格納領域の切り替え。
R_FCL_NO_FCU_COPY （対応バージョン：V2.12 以降）	下記の仕様を無効に設定し、RH850/D1M1A,D1M1-V2,D1S1 専用にビルドします。 ・FCU ファームウェア転送。 本オプションを有効にした場合、RH850/D1M1A,D1M1-V2,D1S1 以外では使用しないでください。なお、本オプションを有効にした場合 R_FCL_MIRROR_FCU_COPY オプションと R_FCL_NO_BFA_SWITCH オプションは有効にしないでください。
R_FCL_CFL2_START_ADDR_2048K （対応バージョン：V2.12 以降）	コードフラッシュ バンク B の先頭アドレスを 0x00200000 に設定します。ご使用のデバイスにおけるコードフラッシュ バンク B の有無は、デバイスのハードウェアマニュアルをご参照ください。
R_FCL_CFL2_START_ADDR_4096K （対応バージョン：V2.12 以降）	コードフラッシュ バンク B の先頭アドレスを 0x00400000 に設定します。ご使用のデバイスにおけるコードフラッシュバンク B の有無は、デバイスのハードウェアマニュアルをご参照ください。

R_FCL_COMMAND_EXECUTION_MODE には以下に示す値を設定できます。

R_FCL_HANDLER_CALL_INTERNAL（インターナルモード）	
長所	ポーリングが不要
	内蔵 RAM の消費が少ない
短所	セルフプログラミング中はアプリケーションに戻らない
	セルフプログラミング中にユーザコードを実行できるのは、割り込みのみ

R_FCL_HANDLER_CALL_USER（ユーザモード）	
長所	CPU への負荷が小さい
	セルフプログラミングと同時にユーザコードを実行できる
短所	内蔵 RAM の消費が多い
	状態のポーリングが必要

動作モードの詳細については「3.2動作モード」を参照してください。

★　**注意事項**：RH850 FCL Type01 V2.12(FCL V2.12)以降のバージョンで、R_FCL_NO_FCU_COPY、R_FCL_MIRROR_FCU_COPY、および R_FCL_NO_BFA_SWITCH のプリコンパイル定義は各デバイスグループごとに有効/無効を設定する必要があります。

FCL V2.12 以降のプリコンパイル設定：

FCL V2.12 以降 プリコンパイル定義	F1L/F1M/ F1H	D1L/ D1M1/D1M1H D1M2/D1M2H	D1M1A/ D1M1-V2 D1S1	F1K/F1KM/ F1KH	Futute Product
R_FCL_NO_BFA_SWITCH	無効	無効	無効	有効	無効
R_FCL_MIRROR_FCU_COPY	無効	無効	無効	無効	有効
R_FCL_NO_FCU_COPY	無効	無効	有効	無効	無効

注意：バージョン毎の対象製品は、各ライブラリに付属のsupport.txtをご確認ください。

4.2　ランタイム設定

この設定には重要な本 FCL 関連の情報（CPU 周波数、ID 認証コードなど）が含まれています。ランタイム設定はディスクリプタ構造体（r_fcl_descriptor_t 参照）に格納されており、r_fcl_types.h で宣言されますが、ユーザアプリケーションで定義され、関数 R_FCL_Init によってライブラリに渡されます。ファイル fcl_descriptor.c はディスクリプタ構造体の定義および設定の例を示し、fcl_descriptor.h は構造体への設定に必要な定義の例を示します。

以下の定義の値は必ずご確認の上、ユーザによって設定してください。

(1) AUTHENTICATION_ID：

コードフラッシュ書き換えに必要なID認証の値を設定してください。また、ID認証の詳細に関しましては対象デバイスのユーザーズマニュアルをご参照ください。なお、初期値はすべて0xFFです。

(2) CPU_FREQUENCY_MHZ：

CPU動作周波数を設定してください。この周波数からFCL内部でタイミング計算に使用します。小数点が存在する場合、小数点以下を切り上げた値を設定してください。CPUの動作周波数については、対象デバイスのユーザーズマニュアルをご参照ください。

(3) FCL_RAM_ADDRESS

R_FCL_CopySections 関数でコピー先の先頭アドレスとして使用します。R_FCL_CopySections 関数を使用してセクションをコピーする場合は、コピー先の先頭アドレスを設定してください。また、R_FCL_CalcFctAddr 関数によるアドレス計算にも使用します。なお、R_FCL_CopySections 関数、および R_FCL_CalcFctAddr 関数を使用しない場合は、本定義の設定値を変更する必要はありません。

ランタイム設定例：

```
#define FCL_CPU_FREQUENCY_MHZ 160
#define FCL_AUTHENTICATION_ID {0xFFFFFFFF, ¥
                               0xFFFFFFFF, ¥
                               0xFFFFFFFF, ¥
                               0xFFFFFFFF }
#define FCL_RAM_ADDRESS         0xFEDF0000
```

4.3　データ型

本節では､FCL が使用および提供するすべてのデータ定義について説明します｡ユーザアプリケーションにおいて型の不一致が起こる可能性を減らすために、提供されている型を正確に使用してください。

★ **RH850 FCL Type01 V2.13(FCL V2.13)以降のバージョンでコンパイルをする場合**：

FCL V2.13 以降のバージョンでは､データ型の定義を ISO で定められた標準的な C 言語の規格(C99 規格以降)でコンパイルすることを前提としています。

GHS 社製コンパイラとルネサスエレクトロニクス社製コンパイラで､C99 以降の規格が指定されている場合、各コンパイラが提供するヘッダファイル”stdint.h”が使用されます。

C99 規格以降の規格が指定されていない場合は、"r_typedefs.h”内の定義（stdint.h と同等）が使用されます。

※C99 規格：ISO/IEC 9899:1999

RH850 FCL Type01 V2.12(FCL V2.12)以前のバージョンでコンパイルをする場合：

FCL V2.12 以前のバージョンで C99 規格のコンパイルをする場合、以下の 2 ファイルで定義されている「#include "r_typedefs.h"」を「#include "stdint.h”」に変更してください。

・r_fcl_hw_access.c

・r_fcl_user_if.c

≪変更例≫

```
#include "stdint.h"
/* #include "r_typedefs.h" */
```

また、添付サンプルを使用する場合も同様に、定義されている「#include "r_typedefs.h"」を「#include "stdint.h"」に変更してください。

RH850 FCL Type01 の対象サンプルファイルは、以下の通りです。

・main.c

・fcl_descriptor.c

・fcl_user.c

4.3.1　単純型定義

★　**コンパイラでC99以降の規格が指定されている場合の型定義**：

各コンパイラが提供するヘッダファイル”stdint.h”をご確認ください。

コンパイラでC99以降の規格が指定されていない場合の型定義("r_typedefs.h”内記述の抜粋)：

```
typedef signed char          int8_t;
typedef unsigned char        uint8_t;
typedef signed short         int16_t;
typedef unsigned short       uint16_t;
typedef signed long          int32_t;
typedef unsigned long        uint32_t;
```

説明：

これらの単純型は本FCLのAPI全体を通じて使用されます。

本FCL固有のすべての単純型定義については、本FCLのインストールパッケージに入っているファイルr_typedefs.hを参照してください。(C99規格でコンパイルする場合は、stdint.hを参照してください。)

4.3.2　r_fcl_command_t

型定義：

```
typedef enum R_FCL_COMMAND_T
{
      R_FCL_CMD_PREPARE_ENV,
      R_FCL_CMD_ERASE,
      R_FCL_CMD_WRITE,
      R_FCL_CMD_SET_LOCKBIT,
      R_FCL_CMD_GET_LOCKBIT,
      R_FCL_CMD_ENABLE_LOCKBITS,
      R_FCL_CMD_DISABLE_LOCKBITS,
      R_FCL_CMD_SET_OTP,
      R_FCL_CMD_GET_OTP,
      R_FCL_CMD_SET_OPB,
      R_FCL_CMD_GET_OPB,
      R_FCL_CMD_SET_ID,
      R_FCL_CMD_GET_ID,
      R_FCL_CMD_SET_READ_PROTECT_FLAG,
      R_FCL_CMD_GET_READ_PROTECT_FLAG,
      R_FCL_CMD_SET_WRITE_PROTECT_FLAG,
      R_FCL_CMD_GET_WRITE_PROTECT_FLAG,
      R_FCL_CMD_SET_ERASE_PROTECT_FLAG,
      R_FCL_CMD_GET_ERASE_PROTECT_FLAG,
      R_FCL_CMD_SET_SERIAL_PROG_DISABLED,
      R_FCL_CMD_GET_SERIAL_PROG_DISABLED,
      R_FCL_CMD_SET_SERIAL_ID_ENABLED,
      R_FCL_CMD_GET_SERIAL_ID_ENABLED,
      R_FCL_CMD_SET_RESET_VECTOR,
      R_FCL_CMD_GET_RESET_VECTOR,
      R_FCL_CMD_GET_BLOCK_CNT,
      R_FCL_CMD_GET_BLOCK_END_ADDR,
      R_FCL_CMD_GET_DEVICE_NAME
} r_fcl_command_t;
```

説明：

コマンドは1つの関数、R_FCL_Execute関数により開始され、その後、R_FCL_Handler関数により制御されます。使用可能なコマンドの詳細については「4.5 FCLのコマンド」を参照してください。

メンバ：

次の表をご参照ください。なお、コマンド名から接頭辞のR_FCL_CMD_を外してあることにご注意ください。

表 3　使用可能なコマンド一覧

★

メンバ	説明
PREPARE_ENV	FCLの内部関数を内蔵RAMにコピーし、初期化します **注意**: RH850/F1K,F1KM,F1KH、およびD1M1A,D1M1-V2,D1S1は、FCLの内部関数を内蔵RAMにコピーしません
ERASE	指定したコードフラッシュのブロックを消去します
WRITE	256バイト単位でコードフラッシュに書き込みます
SET_LOCKBIT	ロックビットを設定します なお、ロックビットを設定した場合、初期値では設定が有効になっています ロックビットの設定を有効・無効に変更するには R_FCL_CMD_ENABLE_LOCKBITSコマンド、 R_FCL_CMD_DISABLE_LOCKBITSコマンドの実行が必要です
GET_LOCKBIT	指定したブロックのロックビットの設定値を読み出します
ENABLE_LOCKBITS	設定しているロックビットを有効にします
DISABLE_LOCKBITS	設定しているロックビットを無効にします
SET_OTP	OTP（One Time Programming）ビットを設定します
GET_OTP	設定しているOTPビットの設定値を読み出します
SET_OPB	オプションバイトを設定します
GET_OPB	設定しているオプションバイトの設定値を読み出します
SET_ID	ID認証のIDコードを設定します
GET_ID	設定しているID認証のIDコードを読み出します
SET_READ_PROTECT_FLAG	リードコマンド禁止フラグを禁止に設定します
GET_READ_PROTECT_FLAG	設定しているリードコマンド禁止フラグを読み出します
SET_WRITE_PROTECT_FLAG	プログラムコマンド禁止フラグを禁止に設定します
GET_WRITE_PROTECT_FLAG	設定しているプログラムコマンド禁止フラグを読み出します
SET_ERASE_PROTECT_FLAG	ブロック消去コマンド禁止フラグを禁止に設定します
GET_ERASE_PROTECT_FLAG	設定しているブロック消去コマンド禁止フラグを読み出します
SET_SERIAL_PROG_DISABLED	シリアルプログラマ接続禁止設定を有効に設定します
GET_SERIAL_PROG_DISABLED	設定しているシリアルプログラマ接続禁止設定の設定状態（有効 ／ 無効）を読み出します
SET_SERIAL_ID_ENABLED	シリアルIDを有効に設定します この設定によりID認証のIDをシリアルIDとしても使用します
GET_SERIAL_ID_ENABLED	設定しているシリアルIDの設定状態（有効 ／ 無効）を読み出します
SET_RESET_VECTOR	可変リセットベクタの値を設定します
GET_RESET_VECTOR	設定している可変リセットベクタの設定値を読み出します
GET_BLOCK_CNT	コードフラッシュの総ブロック数を取得します
GET_BLOCK_END_ADDR	指定したブロックの最終アドレスを取得します
GET_DEVICE_NAME	使用しているデバイスのデバイス名を取得します

4.3.3　r_fcl_status_t

型定義：

```
typedef enum R_FCL_STATUS_T
{
      R_FCL_OK,
      R_FCL_BUSY,
      R_FCL_SUSPENDED,
      R_FCL_ERR_FLMD0,
      R_FCL_ERR_PARAMETER,
      R_FCL_ERR_PROTECTION,
      R_FCL_ERR_REJECTED,
      R_FCL_ERR_FLOW,
      R_FCL_ERR_WRITE,
      R_FCL_ERR_ERASE,
      R_FCL_ERR_COMMAND,
      R_FCL_CANCELLED,
      R_FCL_ERR_INTERNAL
} r_fcl_status_t;
```

説明：

列挙型r_fcl_status_tはFCLの状態を定義しています。FCLコマンドの現在の状態を示すために、上記の状態/エラーコードが本FCLによって返されます。初期化とサスペンド/レジューム、キャンセルに関する他のAPI関数も、この状態コードを返します。すべての状態コードとエラーコードの根本的な原因と解釈は、実行される動作または呼び出される関数に依存します。動作を示すコードの意味を次の表で説明します。

メンバ：

メンバ	説明
R_FCL_OK	要求された動作が正常終了しました。
R_FCL_BUSY	要求された動作が正常に起動し、進行中です。
R_FCL_SUSPENDED	現在の動作がサスペンドしています。
R_FCL_ERR_FLMD0	FLMD0 のハードウェア保護が有効であるため、要求されたコマンドが実行されませんでした。
R_FCL_ERR_PARAMETER	入力データが不正なため、要求されたコマンドが実行されませんでした。
R_FCL_ERR_PROTECTION	セキュリティ機能が有効である（消去保護が有効、ロックビットがセット済みなど）ため、要求されたコマンドが実行されませんでした。
R_FCL_ERR_REJECTED	別のコマンドが操作中のため、要求されたコマンドが実行されませんでした。
R_FCL_ERR_FLOW	初期化またはサスペンド/レジュームのシーケンスが不正なため、現在の要求が実行されませんでした。
R_FCL_ERR_WRITE	未消去領域への書き込みまたはハードウェア故障または FLMD 端子の状態が不安定であったため、要求された書き込みコマンドにエラーが生じました。
R_FCL_ERR_ERASE	ハードウェアの消去エラーにより、要求された消去コマンドにエラーが生じました。
R_FCL_ERR_COMMAND	要求されたコマンドが存在しない、またはプリコンパイルオプションにより要求されたコマンドが無効になっています。
R_FCL_CANCELLED	現在のコマンド操作がキャンセルされました。
R_FCL_ERR_INTERNAL	予期せぬエラーです。

4.3.4　r_fcl_request_t

型定義：

```
typedef volatile struct R_FCL_REQUEST_T
{
      r_fcl_command_t       command_enu;
      uint32_t              bufferAdd_u32;
      uint32_t              idx_u32;
      uint16_t              cnt_u16;
      r_fcl_status_t        status_enu;
} r_fcl_request_t;
```

説明：

すべてのユーザ動作は、主にR_FCL_Execute関数により開始されます。実行に必要なすべての情報はリクエスト構造体によってFCLに渡されます。また、そのエラーは同じ構造体によって返されます。この構造体の詳細については「3.3 リクエスト/レスポンス型アーキテクチャ」を参照してください。目的のコマンド毎にこの構造体に値を設定する方法と結果を確認する方法については、「4.5 FCLのコマンド」に詳説します。

メンバ：

メンバ	説明
command_enu	R_FCL_Execute 関数へ設定するコマンド
bufferAdd_u32	データバッファの先頭アドレス 1 ）書き込みコマンド（R_FCL_CMD_WRITE）の場合 ユーザの書き込むデータを格納しているデータバッファ 2 ）すべての取得機能コマンド（R_FCL_CMD_GET_xコマンド）※の場合 取得するデータを格納するデータバッファ 3 ）すべての設定機能コマンド（R_FCL_CMD_SET_xコマンド）※の場合 設定するデータを格納しているデータバッファ（必要に応じて） 4 ）その他の機能 設定無効
idx_u32	インデックス 1 ）書き込みコマンド（R_FCL_CMD_WRITE）の場合 書き込み先アドレス 2 ）消去コマンド（R_FCL_CMD_ERASE）の場合 消去を実行する先頭ブロック 3 ）R_FCL_CMD_SET_LOCKBIT，R_FCL_CMD_SET_OTPの場合 設定するブロック番号 4 ）R_FCL_CMD_GET_BLOCK_END_ADDR, R_FCL_CMD_GET_LOCKBIT, R_FCL_CMD_GET_OTPの場合 設定値を読み出すブロック番号 5 ）その他の機能 設定無効
cnt_u16	カウント 1 ）書き込みコマンド（R_FCL_CMD_WRITE）の場合 書き込みデータ数 2 ）消去コマンド（R_FCL_CMD_ERASE）の場合 消去を実行するブロック数 3 ）その他の機能 設定無効
status_enu	FCLの状態、エラー情報。 なお、R_FCL_Execute 関数、R_FCL_Handler 関数以外に R_FCL_SuspendRequest 関数、R_FCL_ResumeRequest 関数、 R_FCL_CancelRequest 関数実行時もリクエスト構造体メンバ status_enu を参照します。

※　R_FCL_CMD_SET_，R_FCL_CMD_GET_以降の文字列をxで省略しています。

4.3.5　r_fcl_descriptor_t

型定義：

```
typedef struct R_FCL_DESCRIPTOR_T
{
      uint32_t      id_au32[4];
      uint32_t      addrRam_u32;
      uint16_t      frequencyCpuMHz_u16;
} r_fcl_descriptor_t;
```

説明：

ランタイム設定（「4.2　ランタイム設定」参照）は独立したデータ型で定義されます。データ型の変数は初期化フェーズで読み出され、設定に応じて内部変数がセットされます。

メンバ：

メンバ	説明
id_au32	コードフラッシュ書き換えに必要な ID 認証の値
addrRam_u32	R_FCL_CopySections 関数、R_FCL_CalcFctAddr 関数で使用するコピー先の先頭アドレス
frequencyCpuMHz_u16	CPU 動作周波数

4.4　関数

本資料に記載されているすべてのAPI関数を以下にまとめます。

R_FCL_Init
R_FCL_CopySections
R_FCL_CalcFctAddr
R_FCL_GetVersionString
R_FCL_Execute
R_FCL_Handler
R_FCL_SuspendRequest
R_FCL_ResumeRequest
R_FCL_CancelRequest

4.4.1　初期化

4.4.1.1 R_FCL_Init

概要：FCLの初期化。

インタフェース：Cインタフェース

```
r_fcl_status_t R_FCL_Init (const r_fcl_descriptor_t * descriptor_pstr)
```

引数：

引数	型	アクセス	説明
*descriptor_pstr	r_fcl_descriptor_t	r	ランタイム設定のポインタ

戻り値：

型	説明		
r_fcl_status_t	R_FCL_OK	説明	正常終了しました
		原因	正常動作です
		処置	無し
	R_FCL_ERR_PARAMETER	説明	今回のFCL関数の実行を拒否しました
		原因	引数のポインタが不正です
		処置	正しい引数のポインタを設定してください

事前条件：本関数は内蔵ROMから実行すること。

事後条件：なし

説明：この関数はFCLを初期化するもので、FCL関数を実行する前に呼び出す必要があります。内部変数を初期化して、パラメータをチェックします。

例：

```
/* Initialize Self-Programming Library */
r_fcl_status_t status_enu;

status_enu = R_FCL_Init (&RTConfig_enu);
/* Error treatment ... */
```

4.4.1.2 R_FCL_CopySections

概要：FCLのセクションを内蔵ROMから内蔵RAMにコピーします。

インタフェース：Cインタフェース

```
r_fcl_status_t R_FCL_CopySections (void)
```

引数： なし

戻り値：

型	説明		
r_fcl_status_t	R_FCL_OK	説明	正常終了しました
		原因	正常動作です
		処置	無し
	R_FCL_ERR_FLOW	説明	今回のFCL関数の実行を拒否しました
		原因	FCLは、初期化を実行していない状態、または不正な状態です
		処置	さらなるFCL操作を中止し、根本的な原因を調査してください
	R_FCL_ERR_INTERNAL	説明	今回の FCL 関数の実行を拒否しました
		原因	ランタイム設定のFCL_RAM_ADDRESSの値が不正です
		処置	正しいFCL_RAM_ADDRESSの値を設定してください

事前条件：・R_FCL_Init関数を実行し、R_FCL_Init関数の戻り値がR_FCL_OKとなっていること。
・本関数は内蔵ROMから実行すること。

事後条件：なし

説明：この関数を使用して、FCLの一部のコードセクションを指定された内蔵RAMのコピー先アドレスにコピーします。
≪コピーするセクション≫
R_FCL_CODE_USRINT, R_FCL_CODE_USR, R_FCL_CODE_RAM, R_FCL_CODE_ROMRAM, R_FCL_CODE_RAM_EX_PROT
セクションのコピーの詳細については、「5.3 リンカセクション」を参照してください。

例：

```
/* Copy FCL to internal RAM */
r_fcl_status_t status_enu;

status_enu = R_FCL_CopySections ();
/* Error treatment */
```

4.4.1.3 R_FCL_CalcFctAddr

概要：コピー処理後の新しいアドレスを算出する。

インタフェース：Cインタフェース

```
uint32_t R_FCL_CalcFctAddr (uint32_t addFct_u32)
```

引数：

引数	型	アクセス	説明
addFct_u32	uint32_t	r	コピー元の関数の内蔵 ROM アドレス

戻り値：

型	説明
uint32_t	関数の新しい内蔵 RAM アドレス値

事前条件：・R_FCL_Init関数を実行し、R_FCL_Init関数の戻り値がR_FCL_OKとなっていること。
・本関数は内蔵ROMから実行すること。

事後条件：なし

説明：この関数は、内蔵ROMから内蔵RAMにコピーされる関数の新しいアドレスを算出します。算出可能な関数は、FCLのリンカセクションのR_FCL_CODE_USRINTセクション、R_FCL_CODE_USRセクション、R_FCL_CODE_RAM セクション、R_FCL_CODE_ROMRAM セクションに配置された関数です。

例：

```
/* Calculate new address of user control function fctUserCtrl located in FCL
section R_FCL_CODE_RAM_USR */
uint32_t (*fpFct)( void );

fpFct = (uint32_t(*)())R_FCL_CalcFctAddr ((void *)fctUserCtrl);
```

4.4.2 動作

4.4.2.1 R_FCL_GetVersionString

概要：FCLバージョン情報のポインタを取得します。

インタフェース：Cインタフェース

```
const uint8_t *R_FCL_GetVersionString (void)
```

引数：なし

戻り値：

型	説明
const uint8_t *	アドレス値

事前条件：・R_FCL_CONSTセクションを消去後は本関数を実行しないこと。
・本関数は内蔵ROMから実行すること。

事後条件：なし

説明：バージョン文字列は、ゼロ終端文字列です。バージョン文字列はFCLのコードセクションに格納されています。バージョン文字列は次のように構成されています。

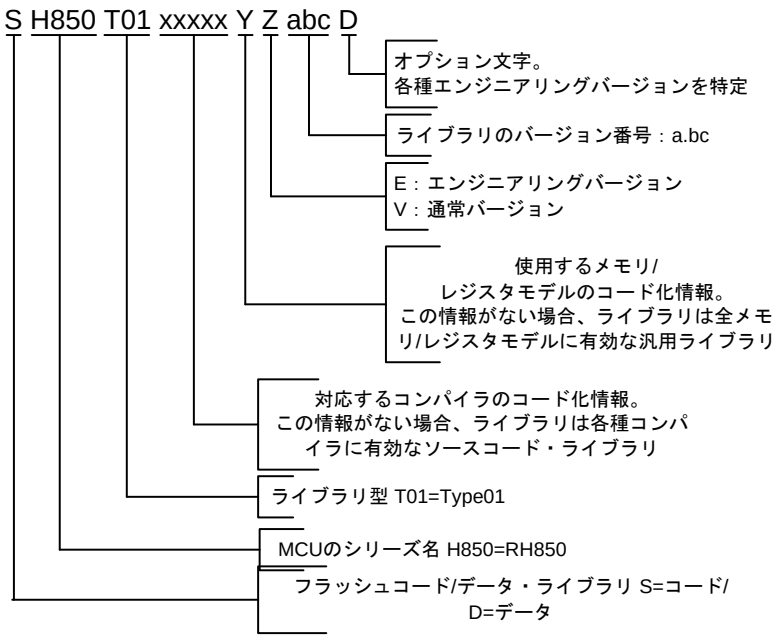

図 6　バージョン文字列

例：

```
/* Read library version */
const uint8_t *version_pu08;

version_pu08 = R_FCL_GetVersionString ();
```

4.4.2.2 R_FCL_Execute

概要：新しいコマンドを起動する。

インタフェース：Cインタフェース

```
void R_FCL_Execute (r_fcl_request_t * request_pstr)
```

引数：

引数	型	アクセス	説明
*request_pstr	r_fcl_request_t	rw	リクエスト構造体のポインタ。詳細は「4.3　データ型」参照。関数が戻ると、リクエスト構造体メンバ status_enu に FCL の状態が返される。（「4.5 FCLのコマンド」参照）

戻り値：なし

事前条件：・FLMD0端子の設定が必要なコマンドは、FLMD0端子を「1」に設定していること。
FLMD0端子の有無、設定に関しましては対象デバイスのユーザーズマニュアルをご参照ください。
・R_FCL_Init関数を実行し、R_FCL_Init関数の戻り値がR_FCL_OKとなっていること。
・FCLのセクションを内蔵RAMにコピーしておくこと
・R_FCL_CMD_PREPARE_ENVコマンド以外は、すでにR_FCL_CMD_PREPARE_ENVコマンドを実行し、R_FCL_Handler関数の戻り値がR_FCL_OKとなっていること。
・本関数は、使用するFCLのモードに応じて内蔵ROMまたは内蔵RAMから実行できます。本関数はR_FCL_CODE_ROMRAMセクション内にあります。

事後条件：・FLMD0端子の設定が必要なコマンドは、コマンド実行中はFLMD0端子を「0」に設定しないでください。
・R_FCL_Execute関数実行後、正常終了の場合、リクエスト構造体メンバstatus_enuはR_FCL_BUSY、またはR_FCL_OKになります。リクエスト構造体メンバstatus_enuがR_FCL_BUSY中は、繰り返しR_FCL_Handler関数を実行してください。
なお、以下のコマンドはR_FCL_Handler関数の実行が不要なコマンドです。

R_FCL_CMD_ENABLE_LOCKBITS	R_FCL_CMD_GET_ERASE_PROTECT_FLAG
R_FCL_CMD_DISABLE_LOCKBITS	R_FCL_CMD_GET_SERIAL_PROG_DISABLED
R_FCL_CMD_GET_OTP	R_FCL_CMD_GET_SERIAL_ID_ENABLED
R_FCL_CMD_GET_OPB	R_FCL_CMD_GET_RESET_VECTOR
R_FCL_CMD_GET_ID	R_FCL_CMD_GET_BLOCK_CNT
R_FCL_CMD_GET_READ_PROTECT_FLAG	R_FCL_CMD_GET_BLOCK_END_ADDR
R_FCL_CMD_GET_WRITE_PROTECT_FLAG	R_FCL_CMD_GET_DEVICE_NAME

説明：本関数はFCLの実行環境の準備とコードフラッシュに対し、消去、書き込みなどのコマンド処理を開始する関数です。コマンドに関するパラメータがリクエスト構造体によりFCLに渡され、コマンド操作の状態と結果も同じ構造体のメンバによりユーザアプリケーションに戻されます。

例：

```
/* Erase blocks 10, 11, 12 and 13 */
r_fcl_request_t myRequest;

myRequest.command_enu        = R_FCL_CMD_ERASE
myRequest.idx_u32            = 10
myRequest.cnt_u16            = 4

R_FCL_Execute (&myRequest);

#if R_FCL_COMMAND_EXECUTION_MODE == R_FCL_HANDLER_CALL_USER
      while (myRequest.status_enu == R_FCL_BUSY)
      {
            R_FCL_Handler ();
      }
#endif

if (R_FCL_OK != myRequest.status_enu)
{
      /* Error treatment ... */
}
```

4.4.2.3 R_FCL_Handler

注意事項：本関数はユーザモードでのみ使用可能。

概要：コマンドを実行し、終了を確認する関数です。

インタフェース：Cインタフェース

```
void R_FCL_Handler (void)
```

引数：なし

戻り値：なし

事前条件：・R_FCL_Execute関数を実行し、リクエスト構造体メンバstatus_enuがR_FCL_BUSYとなっていること。
・FLMD0端子の設定が必要なコマンドは、FLMD0端子を「1」に設定していること。
FLMD0端子の有無、設定に関しましては対象デバイスのユーザーズマニュアルをご参照ください。
・本関数は内蔵RAMから実行すること。本関数はR_FCL_CODE_RAMセクション内にあります。

事後条件：・status_enuがR_FCL_BUSY中は、繰り返しR_FCL_Handler関数を実行してください。
・FLMD0端子の設定が必要なコマンドは、コマンド実行中はFLMD0端子を「0」に設定しないでください。

説明：この関数はFCLのコマンド操作を行います。R_FCL_Executeにより動作を開始した後、この関数を繰り返し呼び出す必要があります。この関数は動作状態を確認し、動作が終了するとリクエスト構造体メンバstatus_enuを更新します。これにより、コマンド操作の終了をポーリングすることができます。

例：R_FCL_Execute（ユーザモード）を参照してください

4.4.2.4 R_FCL_SuspendRequest

注意事項：本関数はユーザモードでのみ使用可能。

概要：実行中の消去、書き込み動作のサスペンドを要求する（たとえば、コードフラッシュの読み出しを可能にするため）。

インタフェース：Cインタフェース

```
r_fcl_status_t R_FCL_SuspendRequest (void)
```

引数：なし

戻り値：

型	説明		
r_fcl_status_t	R_FCL_OK	説明	正常にFCLサスペンド要求を受け付けました
		原因	正常動作です
		処置	無し
	R_FCL_ERR_FLOW	説明	今回のFCL関数の実行を拒否しました
		原因	FCLは、初期化を実行していない状態、または不正な状態です
		処置	さらなるFCL操作を中止し、根本的な原因を調査してください
	R_FCL_ERR_REJECTED	説明	本関数の実行を拒否しました
		原因	サスペンドを実行できないFCL 関数が実行されています
		処置	さらなる FCL 操作を中止し、根本的な原因を調査してください

事前条件：・すでにリクエスト構造体メンバstatus_enuがR_FCL_CMD_ERASEコマンド、R_FCL_CMD_WRITEコマンドによってR_FCL_BUSYとなっていること。
・すでにサスペンド中でないこと。
・R_FCL_CancelRequest関数を実行中でないこと。
・本関数は内蔵RAMから実行すること。本関数はR_FCL_CODE_RAMセクション内にあります。

事後条件：・別のR_FCL_CMD_ERASEコマンド動作を実行するためにR_FCL_CMD_ERASEコマンドをサスペンドさせることはできない。
・サスペンドされたコマンドがR_FCL_CMD_WRITEコマンドの場合は、R_FCL_CMD_ERASEコマンドとR_FCL_CMD_WRITEコマンドのどちらも実行できない。
・リクエスト構造体メンバstatus_enuがR_FCL_BUSY中は、繰り返しR_FCL_Handler関数を実行してください。

説明：この関数は実行中のR_FCL_CMD_ERASEコマンド、またはR_FCL_CMD_WRITEコマンドをサスペンドさせます。サスペンドを要求できるのはこの関数だけです。サスペンド処理はR_FCL_Handler関数により実行されます。

例：

```
/* Erase blocks 0, 1, 2 and 3 */
r_fcl_request_t myRequest;
r_fcl_status_t    srRes_enu;
uint32_t             i;

myRequest.command_enu        = R_FCL_CMD_ERASE;
myRequest.idx_u32              = 0;
myRequest.cnt_u16              = 4;

R_FCL_Execute (&myRequest);

/* call the handler some time */
i = 0;
while ((myRequest.status_enu == R_FCL_BUSY) && (i < 10))
{
      R_FCL_Handler ();
      i++;
}

/* Suspend request and wait until suspended */
srRes_enu = R_FCL_SuspendRequest ();
if (srRes_enu != R_FCL_OK)
{
      /* Error treatment ... */
}

while (myRequest.status_enu != R_FCL_SUSPENDED)
{
      R_FCL_Handler ();
}

/* Now the FCL is suspended and we can read the Flash ... */

/* Erase resume */
srRes_enu = R_FCL_ResumeRequest ();
if (srRes_enu != R_FCL_OK)
{
      /* Error treatment ... */
}

/* Finish the erase */
while (myRequest.status_enu == R_FCL_SUSPENDED)
{
      R_FCL_Handler ();
}
while (myRequest.status_enu == R_FCL_BUSY)
{
      R_FCL_Handler ();
}
if (myRequest.status_enu != R_FCL_OK)
{
      /* Error treatment ... */
}
```

4.4.2.5 R_FCL_ResumeRequest

注意事項：本関数はユーザモードでのみ使用可能。

概要：事前にサスペンドされたコマンドのレジュームを要求する。

インタフェース：Cインタフェース

```
r_fcl_status_t R_FCL_ResumeRequest (void)
```

引数：なし

戻り値：

型	説明		
r_fcl_status_t	R_FCL_OK	説明	正常にFCLサスペンドからの復帰の要求を受け付けました
		原因	正常動作です
		処置	無し
	R_FCL_ERR_FLOW	説明	今回の FCL 関数の実行を拒否しました
		原因	FCLは、初期化を実行していない状態、または不正な状態です
		処置	さらなるFCL操作を中止し、根本的な原因を調査してください

事前条件：・コマンドが正常にサスペンドしていること。
（リクエスト構造体メンバstatus_enuがR_FCL_SUSPENDEDであること）
・リクエスト構造体の内容がサスペンド期間中に変更された場合、それを復元しておくこと
・本関数は内蔵RAMから実行すること。
本関数はR_FCL_CODE_RAMセクション内にあります。
・FLMD0端子を「1」に設定していること。
FLMD0端子の有無、設定に関しましては対象デバイスのユーザーズマニュアルをご参照ください。

事後条件：リクエスト構造体メンバstatus_enuがR_FCL_SUSPENDED中は、繰り返しR_FCL_Handler関数を実行してください。

説明：この関数は、事前にサスペンドされた FCL 動作をレジュームするよう要求します。レジュームを要求できるのはこの関数だけです。レジューム処理は R_FCL_Handler 関数により実行されます。

例：R_FCL_SuspendRequest を参照してください

4.4.2.6 R_FCL_CancelRequest

注意事項：本関数はユーザモードでのみ使用可能。

概要：実行中の消去、書き込み、サスペンド動作のキャンセルを要求する。

インタフェース：Cインタフェース

```
r_fcl_status_t R_FCL_CancelRequest (void)
```

引数：なし

戻り値：

型	説明		
r_fcl_status_t	R_FCL_OK	説明	正常にキャンセル要求を受け付けました
		原因	正常動作です
		処置	無し
	R_FCL_ERR_FLOW	説明	今回のFCL関数の実行を拒否しました
		原因	FCLは、初期化を実行していない状態、またはR_FCL_BUSYでない状態、FCLサスペンドではない状態です
		処置	さらなるFCL操作を中止し、根本的な原因を調査してください
	R_FCL_ERR_REJECTED	説明	今回のFCL関数の実行を拒否しました
		原因	R_FCL_CancelRequest関数が実行不可能なコマンドが実行中です
		処置	さらなるFCL操作を中止し、根本的な原因を調査してください

事前条件：・R_FCL_CMD_ERASEコマンド、またはR_FCL_CMD_WRITEコマンドが実行中(リクエスト構造体メンバstatus_enuがR_FCL_BUSY)かサスペンド中(リクエスト構造体メンバstatus_enuがR_FCL_SUSPENDED)であること。
・別のキャンセルリクエストが受け付けられていないこと。
・本関数は内蔵RAMから実行すること。本関数はR_FCL_CODE_RAMセクション内にあります。

事後条件：リクエスト構造体メンバstatus_enuがR_FCL_BUSY中は、繰り返しR_FCL_Handler関数を実行してください。

説明：この関数は、事前に実行されたFCL動作をキャンセルするよう要求します。キャンセルを要求できるのはこの関数だけです。キャンセル処理はR_FCL_Handler関数により実行されます。

例：

```
/* Erase block 0,1,2 and 3 */
r_fcl_request_t myRequest ;
r_fcl_status_t srRes_enu ;
uint32_t i ;

myRequest.command_enu = R_FCL_CMD_ERASE
myRequest.idx_u32 = 0
myRequest.cnt_u16 = 4

R_FCL_Execute(&myRequest);

/* call the handler some time */
i= 0;
while ((myRequest.status_enu == R_FCL_BUSY) && (i<10))
{
      R_FCL_Handler ();
      i++;
}

/* Cancel request and wait until cancelled */
srRes_enu = R_FCL_CancelRequest () ;
if (R_FCL_OK != srRes_enu)
{
      /* Error treatment */
      ...
)

while (R_FCL_CANCELLED != myRequest.status_enu)
{
      R_FCL_Handler ();
}
```

4.5 FCLのコマンド

FCL に対して使用可能なコマンドと各コマンドで使用されるリクエスト構造体の概要を次の表に示します。なお、コマンド名から接頭辞の R_FCL_CMD_を外してあることにご注意ください。

表 4　コマンドで使用される構造体

<table>
<tr><th>command_enu</th><th>idx_u32</th><th>cnt_u16</th><th>bufferAdd_u32</th><th>バッファサイズ[バイト]</th></tr>
<tr><td>PREPARE_ENV</td><td colspan="4">設定不要</td></tr>
<tr><td>ERASE</td><td>消去を開始する先頭ブロック番号</td><td>消去するブロックの数</td><td colspan="2">設定不要</td></tr>
<tr><td>WRITE</td><td>書き込みを開始する先頭アドレス[256 バイトアライン]</td><td>書き込むデータの数[256 バイト単位]</td><td>書き込むデータを格納しているデータバッファの先頭アドレス</td><td>256×n※</td></tr>
<tr><td>SET_LOCKBIT</td><td rowspan="2">設定するブロックの番号</td><td rowspan="2">設定不要</td><td colspan="2" rowspan="2">設定不要</td></tr>
<tr><td>SET_OTP</td></tr>
<tr><td>GET_BLOCK_END_ADDR</td><td>読み出しを実行するブロックの番号</td><td rowspan="3">設定不要</td><td rowspan="3">取得するデータを格納するデータバッファの先頭アドレス</td><td>4</td></tr>
<tr><td>GET_LOCKBIT</td><td rowspan="2">設定値を読み出すブロックの番号</td><td>4</td></tr>
<tr><td>GET_OTP</td><td>4</td></tr>
<tr><td>SET_OPB</td><td colspan="2" rowspan="3">設定不要</td><td rowspan="3">設定するデータを格納しているデータバッファの先頭アドレス</td><td>32</td></tr>
<tr><td>SET_ID</td><td>16</td></tr>
<tr><td>SET_RESET_VECTOR</td><td>16</td></tr>
<tr><td>GET_OPB</td><td colspan="2" rowspan="10">設定不要</td><td rowspan="10">取得するデータを格納するデータバッファの先頭アドレス</td><td>32</td></tr>
<tr><td>GET_ID</td><td>16</td></tr>
<tr><td>GET_READ_PROTECT_FLAG</td><td>4</td></tr>
<tr><td>GET_WRITE_PROTECT_FLAG</td><td>4</td></tr>
<tr><td>GET_ERASE_PROTECT_FLAG</td><td>4</td></tr>
<tr><td>GET_SERIAL_PROG_DISABLED</td><td>4</td></tr>
<tr><td>GET_SERIAL_ID_ENABLED</td><td>4</td></tr>
<tr><td>GET_RESET_VECTOR</td><td>16</td></tr>
<tr><td>GET_BLOCK_CNT</td><td>4</td></tr>
<tr><td>GET_DEVICE_NAME</td><td>16</td></tr>
<tr><td>SET_READ_PROTECT_FLAG</td><td colspan="4" rowspan="7">設定不要</td></tr>
<tr><td>SET_WRITE_PROTECT_FLAG</td></tr>
<tr><td>SET_ERASE_PROTECT_FLAG</td></tr>
<tr><td>SET_SERIAL_PROG_DISABLED</td></tr>
<tr><td>SET_SERIAL_ID_ENABLED</td></tr>
<tr><td>ENABLE_LOCKBITS</td></tr>
<tr><td>DISABLE_LOCKBITS</td></tr>
</table>

※n=1，2，3，…

注意：いずれの場合も、CPU のアラインメントが適用されます。

4.5.1 R_FCL_CMD_PREPARE_ENV

FCL の内部関数を内蔵 RAM にコピーし、初期化します。

ただし、RH850/F1K, F1KM, F1KH, D1M1A, D1M1-V2, D1S1 は内部関数を内蔵 RAM にコピーする必要が無いため、初期化のみを実行します。

なお、R_FCL_CMD_PREPARE_ENV コマンド実行中に FLERR 割り込み注が発生しないように、FLERR 割り込み[注]をマスクし、復帰時にマスクを解除しています。

注 FLERR 割り込みの詳細は対象デバイスのユーザーズマニュアルを参照してください。

以下にリクエスト構造体の設定方法を示します。

表 5　リクエスト構造体の設定

リクエスト構造体	値	説明
command_enu	R_FCL_CMD_PREPARE_ENV	コマンド
bufferAdd_u32	設定不要	
idx_u32	設定不要	
cnt_u16	設定不要	
status_enu	次表を参照	R_FCL_Execute および R_FCL_Handler により自動的に更新

ユーザモードの場合、コマンド操作中はリクエスト構造体メンバ status_enu が R_FCL_BUSY に設定されます。

表 6　本コマンドが取り得るリクエスト構造体メンバ status_enu の値

状態	背景と対応	
R_FCL_OK	説明	正常終了しました
	原因	正常動作です
	処置	無し
R_FCL_BUSY※	説明	（初回 R_FCL_Execute 実行時）コマンド操作は正常に開始しました。または実行中です
	原因	正常動作です
	処置	操作を続けてください
R_FCL_ERR_FLOW	説明	今回の FCL 関数の実行を拒否しました
	原因	FCL は、初期化を実行していない状態、または不正な状態です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_REJECTED	説明	今回の FCL 関数の実行を拒否しました
	原因	他の FCL 関数、コマンドが実行中です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_PARAMETER	説明	今回の FCL 関数の実行を拒否しました
	原因	設定しているパラメータ（CPU 動作周波数）が不正です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_INTERNAL	説明	通常発生しない予期せぬエラーです
	原因	1.FCL では原因を判別できません 2.ユーザプログラムによって、FCL が使用する内蔵 RAM を破壊している可能性があります 3.FCL が破壊されている可能性もあります
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_PROTECTION	説明	今回の FCL 関数の実行を拒否しました
	原因	ID 認証の値が不正です
	処置	正しい ID 認証の値を設定してください
R_FCL_ERR_FLMD0	説明	今回の FCL 関数の実行を拒否しました
	原因	FLMD0 端子を正しく設定していません
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください

※　ユーザモードでのみ使用可能

4.5.2 R_FCL_CMD_ERASE

指定したコードフラッシュのブロックを消去します。

以下にリクエスト構造体の設定方法を示します。

表 7　リクエスト構造体の設定

リクエスト構造体	値	説明
command_enu	R_FCL_CMD_ERASE	コマンド
bufferAdd_u32	設定不要	
idx_u32	「0」～ 「デバイスのブロック数－1」	ユーザ領域内の消去を開始する 先頭ブロック番号
	「0x80000000」～ 「0x80000000 + 拡張ユーザ領域のブロック数－1」	拡張ユーザ領域の消去を開始する 先頭ブロック番号
cnt_u16	「1」～ 「デバイスのブロック数－idx_u32」	消去するブロック数
status_enu	次表を参照	R_FCL_Execute および R_FCL_Handler により自動的に更新

ユーザモードの場合、コマンド操作中はリクエスト構造体メンバ status_enu が R_FCL_BUSY に設定されます。

表 8　本コマンドが取り得るリクエスト構造体メンバ status_enu の値

状態	背景と対応	
R_FCL_OK	説明	正常終了しました
	原因	正常動作です
	処置	無し
R_FCL_BUSY※	説明	（初回 R_FCL_Execute 実行時）コマンド操作は正常に開始しました。または実行中です
	原因	正常動作です
	処置	操作を続けてください
R_FCL_SUSPENDED※	説明	FCL サスペンド中です
	原因	正常動作です
	処置	無し
R_FCL_ERR_FLOW	説明	今回の FCL 関数の実行を拒否しました
	原因	FCL は、初期化を実行していない状態、または不正な状態です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_FLMD0	説明	今回の FCL 関数の実行を拒否しました
	原因	FLMD0 端子を正しく設定していません
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_REJECTED	説明	今回の FCL 関数の実行を拒否しました
	原因	他の FCL 関数、コマンドが実行中です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_PARAMETER	説明	今回の FCL 関数の実行を拒否しました
	原因	設定しているパラメータが不正です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_ERASE	説明	消去が失敗しました
	原因	1.コードフラッシュが故障している可能性があります 2.FLMD0 端子のレベルが安定した状態でない可能性があります
	処置	1.さらなる FCL 操作を中止し、原因を調査してください 2.FLMD0 端子のレベルが安定していなかった場合、FLMD0 端子を正しく設定し、再度今回実行しようとしたコマンドを実行してください
R_FCL_ERR_PROTECTION	説明	今回の FCL 関数の実行を拒否しました
	原因	保護（FHVE15 レジスタ/ FHVE3 レジスタ）している領域/設定に対して、禁止されているコマンドを実行しました
	処置	FHVE15 レジスタ/ FHVE3 レジスタの設定をご確認ください
R_FCL_ERR_INTERNAL	説明	通常発生しない予期せぬエラーです
	原因	1.FCL では原因を判別できません 2.ユーザプログラムによって、FCL が使用する内蔵 RAM を破壊している可能性があります 3.FCL が破壊されている可能性もあります 4.OTP を設定している領域に対して、禁止されているコマンドを実行しました
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_CANCELLED※	説明	実行中だった FCL 操作をキャンセルしました
	原因	正常動作です
	処置	無し

※　ユーザモードでのみ使用可能

4.5.3 R_FCL_CMD_WRITE

256 バイト単位でコードフラッシュに書き込みます。

以下にリクエスト構造体の設定方法を示します。

表 9　リクエスト構造体の設定

リクエスト構造体	値	説明
command_enu	R_FCL_CMD_WRITE	コマンド
bufferAdd_u32	例）データバッファ名を data の配列とした場合 &data[0]	書き込むデータを格納している データバッファの先頭アドレス
idx_u32	「0」～ 「コードフラッシュの最終アドレス－256」	ユーザ領域内の書き込みを開始する先頭アドレス （256 バイトアライン）
	「0x01000000」～ 「拡張ユーザ領域の最終アドレス－256」	拡張ユーザ領域内の書き込みを開始する先頭アドレス （256 バイトアライン）
cnt_u16	「1」～ 「コードフラッシュのサイズ/256」	書き込むデータの数 （256 バイト単位）
status_enu	次表を参照	R_FCL_Execute および R_FCL_Handler により自動的に更新

ユーザモードの場合、コマンド操作中はリクエスト構造体メンバ status_enu が R_FCL_BUSY に設定されます。

表 10　本コマンドが取り得るリクエスト構造体メンバ status_enu の値

状態	背景と対応	
R_FCL_OK	説明	正常終了しました
	原因	正常動作です
	処置	無し
R_FCL_BUSY※	説明	（初回 R_FCL_Execute 実行時）コマンド操作は正常に開始しました。または実行中です
	原因	正常動作です
	処置	操作を続けてください
R_FCL_SUSPENDED※	説明	FCL サスペンド中です
	原因	正常動作です
	処置	無し
R_FCL_ERR_FLOW	説明	今回の FCL 関数の実行を拒否しました
	原因	FCL は、初期化を実行していない状態、または不正な状態です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_FLMD0	説明	今回の FCL 関数の実行を拒否しました
	原因	FLMD0 端子を正しく設定していません
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_REJECTED	説明	今回の FCL 関数の実行を拒否しました
	原因	他の FCL 関数、コマンドが実行中です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_PARAMETER	説明	今回の FCL 関数の実行を拒否しました
	原因	設定しているパラメータが不正です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_WRITE	説明	書き込みが失敗しました
	原因	1.コードフラッシュが故障している可能性があります 2.書き込み先が消去した状態でない可能性があります 3.FLMD0 端子のレベルが安定した状態でない可能性があります
	処置	1.書き込み先が消去状態で無かった場合、消去を実行後、再度書き込みを実行してください 2.書き込み先が消去状態であった場合、さらなる FCL 操作を中止し、原因を調査してください 3.FLMD0 端子のレベルが安定していなかった場合、FLMD0 端子を正しく設定し、再度今回実行しようとしたコマンドを実行してください
R_FCL_ERR_PROTECTION	説明	今回の FCL 関数の実行を拒否しました
	原因	保護（FHVE15 レジスタ/ FHVE3 レジスタ）している領域/設定に対して、禁止されているコマンドを実行しました
	処置	FHVE15 レジスタ/ FHVE3 レジスタの設定をご確認ください
R_FCL_ERR_INTERNAL	説明	通常発生しない予期せぬエラーです
	原因	1.FCL では原因を判別できません 2.ユーザプログラムによって、FCL が使用する内蔵 RAM を破壊している可能性があります 3.FCL が破壊されている可能性もあります 4.OTP を設定している領域に対して、禁止されているコマンドを実行しました
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_CANCELLED※	説明	実行中だった FCL 操作をキャンセルしました
	原因	正常動作です
	処置	無し

※　ユーザモードでのみ使用

4.5.4 R_FCL_CMD_SET_LOCKBIT

ロックビットを設定します。なお、ロックビットを設定した場合、初期値では設定が有効になっています。ロックビットの設定を有効・無効に変更するにはR_FCL_CMD_ENABLE_LOCKBITSコマンド、R_FCL_CMD_DISABLE_LOCKBITS コマンドの実行が必要です。

注意：

1. R_FCL_CMD_ERASE コマンドが正常に実行された場合、ブロックの内容と一緒にロックビットも消去されます。
2. ロックビットの設定は、R_FCL_CMD_ENABLE_LOCKBITS コマンドまたは R_FCL_CMD_DISABLE_LOCKBITS コマンドで有効または無効にできます。

以下にリクエスト構造体の設定方法を示します。

表 11　リクエスト構造体の設定

リクエスト構造体	値	説明
command_enu	R_FCL_CMD_SET_LOCKBIT	コマンド
bufferAdd_u32	設定不要	
idx_u32	「0」～ 「デバイスのブロック数–1」	ユーザ領域の設定するブロック番号
	「0x80000000」～ 「0x80000000＋ 拡張ユーザ領域のブロック数－1」	拡張ユーザ領域の設定するブロック番号
cnt_u16	設定不要	
status_enu	次表を参照	R_FCL_Execute および R_FCL_Handler により自動的に更新

ユーザモードの場合、コマンド操作中はリクエスト構造体メンバ status_enu が R_FCL_BUSY に設定されます。

表 12　本コマンドが取り得るリクエスト構造体メンバ status_enu の値

状態	背景と対応	
R_FCL_OK	説明	正常終了しました
	原因	正常動作です
	処置	無し
R_FCL_BUSY※	説明	（初回 R_FCL_Execute 実行時）コマンド操作は正常に開始しました。または実行中です
	原因	正常動作です
	処置	操作を続けてください
R_FCL_ERR_FLOW	説明	今回の FCL 関数の実行を拒否しました
	原因	FCL は、初期化を実行していない状態、または不正な状態です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_FLMD0	説明	今回の FCL 関数の実行を拒否しました
	原因	FLMD0 端子を正しく設定していません
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_WRITE	説明	書き込みが失敗しました
	原因	1.コードフラッシュが故障している可能性があります 2.書き込み先が消去した状態でない可能性があります 3.FLMD0 端子のレベルが安定した状態でない可能性があります
	処置	1.書き込み先が消去状態で無かった場合、消去を実行後、再度書き込みを実行してください 2.書き込み先が消去状態であった場合、さらなる FCL 操作を中止し、原因を調査してください 3.FLMD0 端子のレベルが安定していなかった場合、FLMD0 端子を正しく設定し、再度今回実行しようとしたコマンドを実行してください
R_FCL_ERR_REJECTED	説明	今回の FCL 関数の実行を拒否しました
	原因	他の FCL 関数、コマンドが実行中です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_PARAMETER	説明	今回の FCL 関数の実行を拒否しました
	原因	設定しているパラメータが不正です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_PROTECTION	説明	今回の FCL 関数の実行を拒否しました
	原因	保護（FHVE15 レジスタ/ FHVE3 レジスタ）している領域/設定に対して、禁止されているコマンドを実行しました
	処置	FHVE15 レジスタ/ FHVE3 レジスタの設定をご確認ください
R_FCL_ERR_INTERNAL	説明	通常発生しない予期せぬエラーです
	原因	1.FCL では原因を判別できません 2.ユーザプログラムによって、FCL が使用する内蔵 RAM を破壊している可能性があります 3.FCL が破壊されている可能性もあります 4.OTP を設定している領域に対して、禁止されているコマンドを実行しました
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください

※　ユーザモードでのみ使用可能

4.5.5 R_FCL_CMD_GET_LOCKBIT

指定したブロックのロックビットの設定値を読み出します。

注意：

1. あるブロックで R_FCL_CMD_ERASE コマンドが正常に実行されると、ブロックの内容と一緒にロックビットも消去されます。
2. ロックビットの設定は、R_FCL_CMD_ENABLE_LOCKBITS コマンドまたは R_FCL_CMD_DISABLE_LOCKBITS コマンドで有効または無効にできます。

以下にリクエスト構造体の設定方法を示します。

表 13　リクエスト構造体の設定

リクエスト構造体	値	説明
command_enu	R_FCL_CMD_GET_LOCKBIT	コマンド
bufferAdd_u32	例）データバッファ名を data の配列とした場合 &data[0]	取得するデータを格納するデータバッファの先頭アドレス ≪取得値≫ 0：対象の機能を設定していません 1：対象の機能を設定しています
idx_u32	「0」～ 「デバイスのブロック数－1」	ユーザ領域の設定値を読み出すブロック番号
	「0x80000000」～ 「0x80000000＋ 拡張ユーザ領域のブロック数－1」	拡張ユーザ領域の設定値を読み出すブロック番号
cnt_u16	設定不要	
status_enu	次表を参照	R_FCL_Execute および R_FCL_Handler により自動的に更新

ユーザモードの場合、コマンド操作中はリクエスト構造体メンバ status_enu が R_FCL_BUSY に設定されます。

表 14　本コマンドが取り得るリクエスト構造体メンバ status_enu の値

状態	背景と対応	
R_FCL_OK	説明	正常終了しました
	原因	正常動作です
	処置	無し
R_FCL_BUSY※	説明	（初回 R_FCL_Execute 実行時）コマンド操作は正常に開始しました。または実行中です
	原因	正常動作です
	処置	操作を続けてください
R_FCL_ERR_FLOW	説明	今回の FCL 関数の実行を拒否しました
	原因	FCL は、初期化を実行していない状態、または不正な状態です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_REJECTED	説明	今回の FCL 関数の実行を拒否しました
	原因	他の FCL 関数、コマンドが実行中です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_INTERNAL	説明	通常発生しない予期せぬエラーです
	原因	1.FCL では原因を判別できません 2.ユーザプログラムによって、FCL が使用する内蔵 RAM を破壊している可能性があります 3.FCL が破壊されている可能性もあります 4.OTP を設定している領域に対して、禁止されているコマンドを実行しました
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_PARAMETER	説明	今回の FCL 関数の実行を拒否しました
	原因	設定しているパラメータが不正です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください

※　ユーザモードでのみ使用可能

4.5.6 R_FCL_CMD_ENABLE_LOCKBITS

設定しているロックビットを有効にします。

以下にリクエスト構造体の設定方法を示します。

表 15　リクエスト構造体の設定

リクエスト構造体	値	説明
command_enu	R_FCL_CMD_ENABLE_LOCKBITS	コマンド
bufferAdd_u32	設定不要	
idx_u32	設定不要	
cnt_u16	設定不要	
status_enu	次表を参照	R_FCL_Execute により自動的に更新

表 16　本コマンドが取り得るリクエスト構造体メンバ status_enu の値

状態	背景と対応	
R_FCL_OK	説明	正常終了しました
	原因	正常動作です
	処置	無し
R_FCL_ERR_FLOW	説明	今回の FCL 関数の実行を拒否しました
	原因	FCL は、初期化を実行していない状態、または不正な状態です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_REJECTED※	説明	今回の FCL 関数の実行を拒否しました
	原因	他の FCL 関数、コマンドが実行中です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください

※　ユーザモードでのみ使用可能

4.5.7 R_FCL_CMD_DISABLE_LOCKBITS

設定しているロックビットを無効にします。

ロックビット機能の無効時は、R_FCL_CMD_ERASE が正常に動作すると、消去されたブロックに付随するロックビットも消去されます。

以下にリクエスト構造体の設定方法を示します。

表 17　リクエスト構造体の設定

リクエスト構造体	値	説明
command_enu	R_FCL_CMD_DISABLE_LOCKBITS	コマンド
bufferAdd_u32	設定不要	
idx_u32	設定不要	
cnt_u16	設定不要	
status_enu	次表を参照	R_FCL_Execute により自動的に更新

表 18　本コマンドが取り得るリクエスト構造体メンバ status_enu の値

状態	背景と対応	
R_FCL_OK	説明	正常終了しました
	原因	正常動作です
	処置	無し
R_FCL_ERR_FLOW	説明	今回の FCL 関数の実行を拒否しました
	原因	FCL は、初期化を実行していない状態、または不正な状態です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_REJECTED※	説明	今回の FCL 関数の実行を拒否しました
	原因	他の FCL 関数、コマンドが実行中です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください

※　ユーザモードでのみ使用可能

4.5.8 R_FCL_CMD_SET_OTP

OTP（One Time Programming）ビットを設定します。

本設定はシリアルプログラミングによるコンフィグレーションクリア※も禁止となり、設定を解除できなくなりますのでご注意ください。

※　コンフィグレーションクリアとはシリアルプログラミングがサポートしている ID 認証、セキュリティ設定、プロテクション設定、オプションバイト設定を初期化する機能です。詳細は対象デバイスのユーザーズマニュアルをご参照ください。

以下にリクエスト構造体の設定方法を示します。

表 19　リクエスト構造体の設定

リクエスト構造体	値	説明
command_enu	R_FCL_CMD_SET_OTP	コマンド
bufferAdd_u32	設定不要	
idx_u32	「0」～ 「デバイスのブロック数－1」	ユーザ領域の設定するブロック番号
	「0x80000000」～ 「0x80000000 ＋ 拡張ユーザ領域のブロック数－1」	拡張ユーザ領域の設定するブロック番号
cnt_u16	設定不要	
status_enu	次表を参照	R_FCL_Execute および R_FCL_Handler により自動的に更新

ユーザモードの場合、コマンド操作中はリクエスト構造体メンバ status_enu が R_FCL_BUSY に設定されます。

表 20　本コマンドが取り得るリクエスト構造体メンバ status_enu の値

状態	背景と対応	
R_FCL_OK	説明	正常終了しました
	原因	正常動作です
	処置	無し
R_FCL_BUSY※	説明	（初回 R_FCL_Execute 実行時）コマンド操作は正常に開始しました。または実行中です
	原因	正常動作です
	処置	操作を続けてください
R_FCL_ERR_FLOW	説明	今回の FCL 関数の実行を拒否しました
	原因	FCL は、初期化を実行していない状態、または不正な状態です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_PARAMETER	説明	今回の FCL 関数の実行を拒否しました
	原因	設定しているパラメータが不正です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_FLMD0	説明	今回の FCL 関数の実行を拒否しました
	原因	FLMD0 端子を正しく設定していません
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_WRITE	説明	書き込みが失敗しました
	原因	1.コードフラッシュが故障している可能性があります 2.FLMD0 端子のレベルが安定した状態でない可能性があります
	処置	1.さらなる FCL 操作を中止し、原因を調査してください 2.FLMD0 端子のレベルが安定していなかった場合、FLMD0 端子を正しく設定し、再度今回実行しようとしたコマンドを実行してください
R_FCL_ERR_ERASE	説明	消去が失敗しました
	原因	1.コードフラッシュが故障している可能性があります 2.FLMD0 端子のレベルが安定した状態でない可能性があります
	処置	1.さらなる FCL 操作を中止し、原因を調査してください 2.FLMD0 端子のレベルが安定していなかった場合、FLMD0 端子を正しく設定し、再度今回実行しようとしたコマンドを実行してください
R_FCL_ERR_REJECTED	説明	今回の FCL 関数の実行を拒否しました
	原因	他の FCL 関数、コマンドが実行中です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_INTERNAL	説明	通常発生しない予期せぬエラーです
	原因	1 .FCL では原因を判別できません 2.ユーザプログラムによって、FCL が使用する内蔵 RAM を破壊している可能性があります 3.FCL が破壊されている可能性もあります 4.OTP を設定している領域に対して、禁止されているコマンドを実行しました
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください

※　ユーザモードでのみ使用可能

4.5.9 R_FCL_CMD_GET_OTP

設定している OTP ビットの設定値を読み出します。

以下にリクエスト構造体の設定方法を示します。

表 21　リクエスト構造体の設定

リクエスト構造体	値	説明
command_enu	R_FCL_CMD_GET_OTP	コマンド
bufferAdd_u32	例）データバッファ名を data の配列とした場合 &data[0]	取得するデータを格納するデータバッファの先頭アドレス ≪取得値≫ 0：対象の機能を設定していません 1：対象の機能を設定しています
idx_u32	「0」～ 「デバイスのブロック数－1」	ユーザ領域の設定値を読み出すブロック番号
	「0x80000000」～ 「0x80000000 ＋ 拡張ユーザ領域のブロック数－1」	拡張ユーザ領域の設定値を読み出すブロック番号
cnt_u16	設定不要	
status_enu	次表を参照	R_FCL_Execute により自動的に更新

表 22　本コマンドが取り得るリクエスト構造体メンバ status_enu の値

状態	背景と対応	
R_FCL_OK	説明	正常終了しました
	原因	正常動作です
	処置	無し
R_FCL_ERR_FLOW	説明	今回の FCL 関数の実行を拒否しました
	原因	FCL は、初期化を実行していない状態、または不正な状態です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_PARAMETER	説明	今回の FCL 関数の実行を拒否しました
	原因	設定しているパラメータが不正です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_REJECTED※	説明	今回の FCL 関数の実行を拒否しました
	原因	他の FCL 関数、コマンドが実行中です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください

※ユーザモードでのみ使用可能

4.5.10 R_FCL_CMD_SET_OPB

オプションバイトを設定します。

以下にリクエスト構造体の設定方法を示します。

表 23　リクエスト構造体の設定

リクエスト構造体	値	説明
command_enu	R_FCL_CMD_SET_OPB	コマンド
bufferAdd_u32	例）データバッファ名を data の配列とした場合 &data[0]	設定するデータを格納しているデータバッファの先頭アドレス
idx_u32	設定不要	
cnt_u16	設定不要	
status_enu	次表を参照	R_FCL_Execute および R_FCL_Handler により自動的に更新

ユーザモードの場合、コマンド操作中はリクエスト構造体メンバ status_enu が R_FCL_BUSY に設定されます。

≪参考例≫

<table>
<tr><td>設定値</td><td>unsigned long data[8] ={ 0x03020100, 0x07060504, 0x0B0A0908, 0x0F0E0D0C,
0x13121110, 0x17161514, 0x1B1A1918, 0x1F1E1D1C };
bufferAdd_u32=&data[0]</td></tr>
<tr><td>実行結果</td><td>下記の通り設定されます。

<table>
<tr><th>名称</th><th>値</th><th>名称</th><th>値</th></tr>
<tr><td>オプションバイト0</td><td>0x03020100</td><td>予約領域4</td><td>0x13121110</td></tr>
<tr><td>予約領域1</td><td>0x07060504</td><td>予約領域5</td><td>0x17161514</td></tr>
<tr><td>予約領域2</td><td>0x0B0A0908</td><td>予約領域6</td><td>0x1B1A1918</td></tr>
<tr><td>予約領域3</td><td>0x0F0E0D0C</td><td>予約領域7</td><td>0x1F1E1D1C</td></tr>
</table>

本FCLはオプションバイト0と予約領域1～7までの設定に対応しており、一度にオプションバイト0と予約領域1～7まで設定します。なお、予約領域に関しましては、対象デバイスのユーザーズマニュアルをご確認ください。

例）RH850/F1Lの場合
オプションバイト0のみに対応しています。予約領域はすべて初期値である0xFFFFFFFFで設定してください。</td></tr>
</table>

表 24　本コマンドが取り得るリクエスト構造体メンバ status_enu の値

★

状態	背景と対応	
R_FCL_OK	説明	正常終了しました
	原因	正常動作です
	処置	無し
R_FCL_BUSY※	説明	（初回 R_FCL_Execute 実行時）コマンド操作は正常に開始しました。または実行中です
	原因	正常動作です
	処置	操作を続けてください
R_FCL_ERR_FLOW	説明	今回の FCL 関数の実行を拒否しました
	原因	FCL は、初期化を実行していない状態、または不正な状態です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_PARAMETER	説明	今回の FCL 関数の実行を拒否しました
	原因	設定しているパラメータが不正です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_FLMD0	説明	今回の FCL 関数の実行を拒否しました
	原因	FLMD0 端子を正しく設定していません
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_WRITE	説明	書き込みが失敗しました
	原因	1.コードフラッシュが故障している可能性があります 2.FLMD0 端子のレベルが安定した状態でない可能性があります
	処置	1.さらなる FCL 操作を中止し、原因を調査してください 2.FLMD0 端子のレベルが安定していなかった場合、FLMD0 端子を正しく設定し、再度今回実行しようとしたコマンドを実行してください
R_FCL_ERR_ERASE	説明	消去が失敗しました
	原因	1.コードフラッシュが故障している可能性があります 2.FLMD0 端子のレベルが安定した状態でない可能性があります
	処置	1.さらなる FCL 操作を中止し、原因を調査してください 2.FLMD0 端子のレベルが安定していなかった場合、FLMD0 端子を正しく設定し、再度今回実行しようとしたコマンドを実行してください
R_FCL_ERR_REJECTED	説明	今回の FCL 関数の実行を拒否しました
	原因	他の FCL 関数、コマンドが実行中です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_PROTECTION	説明	今回の FCL 関数の実行を拒否しました
	原因	保護（FHVE15 レジスタ/ FHVE3 レジスタ）している領域/設定に対して、禁止されているコマンドを実行しました
	処置	FHVE15 レジスタ/ FHVE3 レジスタの設定をご確認ください
R_FCL_ERR_INTERNAL	説明	通常発生しない予期せぬエラーです
	原因	1.FCL では原因を判別できません 2.ユーザプログラムによって、FCL が使用する内蔵 RAM を破壊している可能性があります 3.FCL が破壊されている可能性もあります
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください

※ユーザモードでのみ使用可能

4.5.11 R_FCL_CMD_GET_OPB

設定しているオプションバイトの設定値を読み出します。

以下にリクエスト構造体の設定方法を示します。

表 25　リクエスト構造体の設定

リクエスト構造体	値	説明
command_enu	R_FCL_CMD_GET_OPB	コマンド
bufferAdd_u32	例）データバッファ名を data の配列とした場合 &data[0]	取得するデータを格納するデータバッファの先頭アドレス ≪取得値≫ 現在設定しているオプションバイトの値です
idx_u32	設定不要	
cnt_u16	設定不要	
status_enu	次表を参照	R_FCL_Execute および R_FCL_Handler により自動的に更新

ユーザモードの場合、コマンド操作中はリクエスト構造体メンバ status_enu が R_FCL_BUSY に設定されます。

設定値	bufferAdd_u32=&data[0]
実行結果	下記の値を取得します。 unsigned long data[8] ={ 0x03020100, 0x07060504, 0x0B0A0908, 0x0F0E0D0C, 0x13121110, 0x17161514, 0x1B1A1918, 0x1F1E1D1C }; ただし、デバイスに設定されている値を以下と仮定します。

名称	値	名称	値
オプションバイト0	0x03020100	予約領域4	0x13121110
予約領域1	0x07060504	予約領域5	0x17161514
予約領域2	0x0B0A0908	予約領域6	0x1B1A1918
予約領域3	0x0F0E0D0C	予約領域7	0x1F1E1D1C

表 26　本コマンドが取り得るリクエスト構造体メンバ status_enu の値

状態	背景と対応	
R_FCL_OK	説明	正常終了しました
	原因	正常動作です
	処置	無し
R_FCL_ERR_FLOW	説明	今回の FCL 関数の実行を拒否しました
	原因	FCL は、初期化を実行していない状態、または不正な状態です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_REJECTED※	説明	今回の FCL 関数の実行を拒否しました
	原因	他の FCL 関数、コマンドが実行中です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_PARAMETER	説明	今回の FCL 関数の実行を拒否しました
	原因	設定しているパラメータが不正です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください

※ユーザモードでのみ使用可能

4.5.12 R_FCL_CMD_SET_ID

ID 認証の ID コードを設定します。

この ID は、セルフプログラミング時とシリアルプログラミング時に使用されます。

以下にリクエスト構造体の設定方法を示します。

表 27　リクエスト構造体の設定

リクエスト構造体	値	説明
command_enu	R_FCL_CMD_SET_ID	コマンド
bufferAdd_u32	例）データバッファ名を data の配列とした場合 &data[0]	設定するデータを格納しているデータバッファの先頭アドレス
idx_u32	設定不要	
cnt_u16	設定不要	
status_enu	次表を参照	R_FCL_Execute および R_FCL_Handler により自動的に更新

ユーザモードの場合、コマンド操作中はリクエスト構造体メンバ status_enu が R_FCL_BUSY に設定されます。

設定値	unsigned long data[4] ={ 0x03020100, 0x07060504, 0x0B0A0908, 0x0F0E0D0C }; bufferAdd_u32=&data[0]
実行結果	下記の通り設定されます。 ≪各ツールによるGUI表示例≫

Renesas Flash Programmer	IDコード　上位8バイト：0001020304050607
	IDコード　下位8バイト：08090A0B0C0D0E0F
CS+（旧CubeSuite+）	セキュリティID　：000102030405060708090A0B0C0D0E0F
MULTI（GHS社製）	Registry ID code：000102030405060708090A0B0C0D0E0F

表 28　本コマンドが取り得るリクエスト構造体メンバ status_enu の値

状態	背景と対応	
R_FCL_OK	説明	正常終了しました
	原因	正常動作です
	処置	設定した ID を有効にするには、リセットが必要です
R_FCL_BUSY※	説明	（初回 R_FCL_Execute 実行時）コマンド操作は正常に開始しました。または実行中です
	原因	正常動作です
	処置	操作を続けてください
R_FCL_ERR_FLOW	説明	今回の FCL 関数の実行を拒否しました
	原因	FCL は、初期化を実行していない状態、または不正な状態です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_PARAMETER	説明	今回の FCL 関数の実行を拒否しました
	原因	設定しているパラメータが不正です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_FLMD0	説明	今回の FCL 関数の実行を拒否しました
	原因	FLMD0 端子を正しく設定していません
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_WRITE	説明	書き込みが失敗しました
	原因	1.コードフラッシュが故障している可能性があります 2.FLMD0 端子のレベルが安定した状態でない可能性があります
	処置	1.さらなる FCL 操作を中止し、原因を調査してください 2.FLMD0 端子のレベルが安定していなかった場合、FLMD0 端子を正しく設定し、再度今回実行しようとしたコマンドを実行してください
R_FCL_ERR_ERASE	説明	消去が失敗しました
	原因	1.コードフラッシュが故障している可能性があります 2.FLMD0 端子のレベルが安定した状態でない可能性があります
	処置	1.さらなる FCL 操作を中止し、原因を調査してください 2.FLMD0 端子のレベルが安定していなかった場合、FLMD0 端子を正しく設定し、再度今回実行しようとしたコマンドを実行してください
R_FCL_ERR_REJECTED	説明	今回の FCL 関数の実行を拒否しました
	原因	他の FCL 関数、コマンドが実行中です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_PROTECTION	説明	今回の FCL 関数の実行を拒否しました
	原因	保護（FHVE15 レジスタ/ FHVE3 レジスタ）している領域/設定に対して、禁止されているコマンドを実行しました
	処置	FHVE15 レジスタ/ FHVE3 レジスタの設定をご確認ください
R_FCL_ERR_INTERNAL	説明	通常発生しない予期せぬエラーです
	原因	1.FCL では原因を判別できません 2.ユーザプログラムによって、FCL が使用する内蔵 RAM を破壊している可能性があります 3.FCL が破壊されている可能性もあります
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください

※ユーザモードでのみ使用可能

4.5.13 R_FCL_CMD_GET_ID

設定している ID 認証の ID コードを読み出します。

以下にリクエスト構造体の設定方法を示します。

表 29　リクエスト構造体の設定

リクエスト構造体	値	説明
command_enu	R_FCL_CMD_GET_ID	コマンド
bufferAdd_u32	例）データバッファ名を data の配列とした場合 &data[0]	取得するデータを格納するデータバッファの先頭アドレス ≪取得値≫ 現在設定している ID 認証の ID コードです
idx_u32	設定不要	
cnt_u16	設定不要	
status_enu	次表を参照	R_FCL_Execute により自動的に更新

設定値	bufferAdd_u32=&data[0]
実行結果	下記の値を取得します。 unsigned long data[4] ={ 0x03020100, 0x07060504, 0x0B0A0908, 0x0F0E0D0C }; ただし、デバイスに設定されている値を以下と仮定します。 ≪各ツールによるGUI表示例≫

Renesas Flash Programmer	IDコード　上位8バイト：0001020304050607
	IDコード　下位8バイト：08090A0B0C0D0E0F
CS+（旧CubeSuite+）	セキュリティID　：000102030405060708090A0B0C0D0E0F
MULTI（GHS社製）	Registry ID code：000102030405060708090A0B0C0D0E0F

表 30　本コマンドが取り得るリクエスト構造体メンバ status_enu の値

状態	背景と対応	
R_FCL_OK	説明	正常終了しました
	原因	正常動作です
	処置	無し
R_FCL_ERR_FLOW	説明	今回の FCL 関数の実行を拒否しました
	原因	FCL は、初期化を実行していない状態、または不正な状態です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_PARAMETER	説明	今回の FCL 関数の実行を拒否しました
	原因	設定しているパラメータが不正です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_REJECTED※	説明	今回の FCL 関数の実行を拒否しました
	原因	他の FCL 関数、コマンドが実行中です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください

※ユーザモードでのみ使用可能

4.5.14 R_FCL_CMD_SET_READ_PROTECT_FLAG

リードコマンド禁止フラグを禁止に設定します。

禁止状態に設定したセキュリティを FCL 関数で許可状態へと変更することはできません。セキュリティ情報を許可状態にする場合は、専用フラッシュ・プログラマを使用してください。

以下にリクエスト構造体の設定方法を示します。

表 31　リクエスト構造体の設定

リクエスト構造体	値	説明
command_enu	R_FCL_CMD_SET_READ_PROTECT_FLAG	コマンド
bufferAdd_u32	設定不要	
idx_u32	設定不要	
cnt_u16	設定不要	
status_enu	次表を参照	R_FCL_Execute および R_FCL_Handler により自動的に更新

ユーザモードの場合、コマンド操作中はリクエスト構造体メンバ status_enu が R_FCL_BUSY に設定されます。

表 32　本コマンドが取り得るリクエスト構造体メンバ status_enu の値

状態	背景と対応	
R_FCL_OK	説明	正常終了しました
	原因	正常動作です
	処置	無し
R_FCL_BUSY※	説明	（初回 R_FCL_Execute 実行時）コマンド操作は正常に開始しました。または実行中です
	原因	正常動作です
	処置	操作を続けてください
R_FCL_ERR_FLOW	説明	今回の FCL 関数の実行を拒否しました
	原因	FCL は、初期化を実行していない状態、または不正な状態です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_FLMD0	説明	今回の FCL 関数の実行を拒否しました
	原因	FLMD0 端子を正しく設定していません
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_WRITE	説明	書き込みが失敗しました
	原因	1.コードフラッシュが故障している可能性があります 2.FLMD0 端子のレベルが安定した状態でない可能性があります
	処置	1.さらなる FCL 操作を中止し、原因を調査してください 2.FLMD0 端子のレベルが安定していなかった場合、FLMD0 端子を正しく設定し、再度今回実行しようとしたコマンドを実行してください
R_FCL_ERR_ERASE	説明	消去が失敗しました
	原因	1.コードフラッシュが故障している可能性があります 2.FLMD0 端子のレベルが安定した状態でない可能性があります
	処置	1.さらなる FCL 操作を中止し、原因を調査してください 2.FLMD0 端子のレベルが安定していなかった場合、FLMD0 端子を正しく設定し、再度今回実行しようとしたコマンドを実行してください
R_FCL_ERR_REJECTED	説明	今回の FCL 関数の実行を拒否しました
	原因	他の FCL 関数、コマンドが実行中です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_INTERNAL	説明	通常発生しない予期せぬエラーです
	原因	1.FCL では原因を判別できません 2.ユーザプログラムによって、FCL が使用する内蔵 RAM を破壊している可能性があります 3.FCL が破壊されている可能性もあります
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください

※ユーザモードでのみ使用可能

4.5.15 R_FCL_CMD_GET_READ_PROTECT_FLAG

設定しているリードコマンド禁止フラグを読み出します。

以下にリクエスト構造体の設定方法を示します。

表 33　リクエスト構造体の設定

リクエスト構造体	値	説明
command_enu	R_FCL_CMD_GET_READ_PROTECT_FLAG	コマンド
bufferAdd_u32	例）データバッファ名を data の配列とした場合 &data[0]	取得するデータを格納する データバッファの先頭アドレス ≪取得値≫ 0：対象の機能を設定していません 1：対象の機能を設定しています
idx_u32	設定不要	
cnt_u16	設定不要	
status_enu	次表を参照	R_FCL_Execute により自動的に更新

表 34　本コマンドが取り得るリクエスト構造体メンバ status_enu の値

状態	背景と対応	
R_FCL_OK	説明	正常終了しました
	原因	正常動作です
	処置	無し
R_FCL_ERR_FLOW	説明	今回の FCL 関数の実行を拒否しました
	原因	FCL は、初期化を実行していない状態、または不正な状態です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_PARAMETER	説明	今回の FCL 関数の実行を拒否しました
	原因	設定しているパラメータが不正です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_REJECTED※	説明	今回の FCL 関数の実行を拒否しました
	原因	他の FCL 関数、コマンドが実行中です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください

※ユーザモードでのみ使用可能

4.5.16 R_FCL_CMD_SET_WRITE_PROTECT_FLAG

プログラムコマンド禁止フラグを禁止に設定します。

禁止状態に設定したセキュリティを FCL 関数で許可状態へと変更することはできません。セキュリティ情報を許可状態にする場合は、専用フラッシュ・プログラマを使用してください。

以下にリクエスト構造体の設定方法を示します。

表 35　リクエスト構造体の設定

リクエスト構造体	値	説明
command_enu	R_FCL_CMD_SET_WRITE_PROTECT_FLAG	コマンド
bufferAdd_u32	設定不要	
idx_u32	設定不要	
cnt_u16	設定不要	
status_enu	次表を参照	R_FCL_Execute および R_FCL_Handler により自動的に更新

ユーザモードの場合、コマンド操作中はリクエスト構造体メンバ status_enu が R_FCL_BUSY に設定されます。

表 36　本コマンドが取り得るリクエスト構造体メンバ status_enu の値

状態	背景と対応	
R_FCL_OK	説明	正常終了しました
	原因	正常動作です
	処置	無し
R_FCL_BUSY※	説明	（初回 R_FCL_Execute 実行時）コマンド操作は正常に開始しました。または実行中です
	原因	正常動作です
	処置	操作を続けてください
R_FCL_ERR_FLOW	説明	今回の FCL 関数の実行を拒否しました
	原因	FCL は、初期化を実行していない状態、または不正な状態です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_FLMD0	説明	今回の FCL 関数の実行を拒否しました
	原因	FLMD0 端子を正しく設定していません
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_WRITE	説明	書き込みが失敗しました
	原因	1.コードフラッシュが故障している可能性があります 2.FLMD0 端子のレベルが安定した状態でない可能性があります
	処置	1.さらなる FCL 操作を中止し、原因を調査してください 2.FLMD0 端子のレベルが安定していなかった場合、FLMD0 端子を正しく設定し、再度今回実行しようとしたコマンドを実行してください
R_FCL_ERR_ERASE	説明	消去が失敗しました
	原因	1.コードフラッシュが故障している可能性があります 2.FLMD0 端子のレベルが安定した状態でない可能性があります
	処置	1.さらなる FCL 操作を中止し、原因を調査してください 2.FLMD0 端子のレベルが安定していなかった場合、FLMD0 端子を正しく設定し、再度今回実行しようとしたコマンドを実行してください
R_FCL_ERR_REJECTED	説明	今回の FCL 関数の実行を拒否しました
	原因	他の FCL 関数、コマンドが実行中です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_INTERNAL	説明	通常発生しない予期せぬエラーです
	原因	1.FCL では原因を判別できません 2.ユーザプログラムによって、FCL が使用する内蔵 RAM を破壊している可能性があります 3.FCL が破壊されている可能性もあります
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください

※ユーザモードでのみ使用可能

4.5.17 R_FCL_CMD_GET_WRITE_PROTECT_FLAG

設定しているプログラムコマンド禁止フラグを読み出します。

以下にリクエスト構造体の設定方法を示します。

表 37　リクエスト構造体の設定

リクエスト構造体	値	説明
command_enu	R_FCL_CMD_GET_WRITE_PROTECT_FLAG	コマンド
bufferAdd_u32	例）データバッファ名を data の配列とした場合 &data[0]	取得するデータを格納する データバッファの先頭アドレス ≪取得値≫ 0：対象の機能を設定していません 1：対象の機能を設定しています
idx_u32	設定不要	
cnt_u16	設定不要	
status_enu	次表を参照	R_FCL_Execute により自動的に更新

ユーザモードの場合、コマンド操作中はリクエスト構造体メンバ status_enu が R_FCL_BUSY に設定されます。

表 38　本コマンドが取り得るリクエスト構造体メンバ status_enu の値

状態	背景と対応	
R_FCL_OK	説明	正常終了しました
	原因	正常動作です
	処置	無し
R_FCL_ERR_FLOW	説明	今回の FCL 関数の実行を拒否しました
	原因	FCL は、初期化を実行していない状態、または不正な状態です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_PARAMETER	説明	今回の FCL 関数の実行を拒否しました
	原因	設定しているパラメータが不正です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_REJECTED※	説明	今回の FCL 関数の実行を拒否しました
	原因	他の FCL 関数、コマンドが実行中です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください

※ユーザモードでのみ使用可能

4.5.18 R_FCL_CMD_SET_ERASE_PROTECT_FLAG

ブロック消去コマンド禁止フラグを禁止に設定します。

本設定はシリアルプログラミングによるコンフィグレーションクリア※も禁止となり、設定を解除できなくなりますのでご注意ください。

※　コンフィグレーションクリアとはシリアルプログラミングがサポートしている ID 認証、セキュリティ設定、プロテクション設定、オプションバイト設定を初期化する機能です。詳細は対象デバイスのユーザーズマニュアルをご参照ください。

禁止状態に設定したセキュリティを FCL 関数で許可状態へと変更することはできません。セキュリティ情報を許可状態にする場合は、専用フラッシュ・プログラマを使用してください。

以下にリクエスト構造体の設定方法を示します。

表 39　リクエスト構造体の設定

リクエスト構造体	値	説明
command_enu	R_FCL_CMD_SET_ERASE_PROTECT_FLAG	コマンド
bufferAdd_u32	設定不要	
idx_u32	設定不要	
cnt_u16	設定不要	
status_enu	次表を参照	R_FCL_Execute により自動的に更新

ユーザモードの場合、コマンド操作中はリクエスト構造体メンバ status_enu が R_FCL_BUSY に設定されます。

表 40　本コマンドが取り得るリクエスト構造体メンバ status_enu の値

状態	背景と対応	
R_FCL_OK	説明	正常終了しました
	原因	正常動作です
	処置	無し
R_FCL_BUSY※	説明	（初回 R_FCL_Execute 実行時）コマンド操作は正常に開始しました。または実行中です
	原因	正常動作です
	処置	操作を続けてください
R_FCL_ERR_FLOW	説明	今回の FCL 関数の実行を拒否しました
	原因	FCL は、初期化を実行していない状態、または不正な状態です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_FLMD0	説明	今回の FCL 関数の実行を拒否しました
	原因	FLMD0 端子を正しく設定していません
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_WRITE	説明	書き込みが失敗しました
	原因	1.コードフラッシュが故障している可能性があります 2.FLMD0 端子のレベルが安定した状態でない可能性があります
	処置	1.さらなる FCL 操作を中止し、原因を調査してください 2.FLMD0 端子のレベルが安定していなかった場合、FLMD0 端子を正しく設定し、再度今回実行しようとしたコマンドを実行してください
R_FCL_ERR_ERASE	説明	消去が失敗しました
	原因	1.コードフラッシュが故障している可能性があります 2.FLMD0 端子のレベルが安定した状態でない可能性があります
	処置	1.さらなる FCL 操作を中止し、原因を調査してください 2.FLMD0 端子のレベルが安定していなかった場合、FLMD0 端子を正しく設定し、再度今回実行しようとしたコマンドを実行してください
R_FCL_ERR_REJECTED	説明	今回の FCL 関数の実行を拒否しました
	原因	他の FCL 関数、コマンドが実行中です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_INTERNAL	説明	通常発生しない予期せぬエラーです
	原因	1.FCL では原因を判別できません 2.ユーザプログラムによって、FCL が使用する内蔵 RAM を破壊している可能性があります 3.FCL が破壊されている可能性もあります
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください

※ユーザモードでのみ使用可能

4.5.19 R_FCL_CMD_GET_ERASE_PROTECT_FLAG

設定しているブロック消去コマンド禁止フラグを読み出します。

以下にリクエスト構造体の設定方法を示します。

表 41　リクエスト構造体の設定

リクエスト構造体	値	説明
command_enu	R_FCL_CMD_GET_ERASE_PROTECT_FLAG	コマンド
bufferAdd_u32	例）データバッファ名を data の配列とした場合 &data[0]	取得するデータを格納する データバッファの先頭アドレス ≪取得値≫ 0：対象の機能を設定していません 1：対象の機能を設定しています
idx_u32	設定不要	
cnt_u16	設定不要	
status_enu	次表を参照	R_FCL_Execute により自動的に更新

表 42　本コマンドが取り得るリクエスト構造体メンバ status_enu の値

状態	背景と対応	
R_FCL_OK	説明	正常終了しました
	原因	正常動作です
	処置	無し
R_FCL_ERR_FLOW	説明	今回の FCL 関数の実行を拒否しました
	原因	FCL は、初期化を実行していない状態、または不正な状態です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_PARAMETER	説明	今回の FCL 関数の実行を拒否しました
	原因	設定しているパラメータが不正です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_REJECTED※	説明	今回の FCL 関数の実行を拒否しました
	原因	他の FCL 関数、コマンドが実行中です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください

※ユーザモードでのみ使用可能

4.5.20 R_FCL_CMD_SET_SERIAL_PROG_DISABLED

シリアルプログラマ接続禁止設定を有効に設定します。

本設定はシリアルプログラミングによるコンフィグレーションクリア※も禁止となり、設定を解除できなくなりますのでご注意ください。

※　コンフィグレーションクリアとはシリアルプログラミングがサポートしている ID 認証、セキュリティ設定、プロテクション設定、オプションバイト設定を初期化する機能です。詳細は対象デバイスのユーザーズマニュアルをご参照ください。

以下にリクエスト構造体の設定方法を示します。

表 43　リクエスト構造体の設定

リクエスト構造体	値	説明
command_enu	R_FCL_CMD_SET_SERIAL_PROG_DISABLED	コマンド
bufferAdd_u32	設定不要	
idx_u32	設定不要	
cnt_u16	設定不要	
status_enu	次表を参照	R_FCL_Execute および R_FCL_Handler により自動的に更新

ユーザモードの場合、コマンド操作中はリクエスト構造体メンバ status_enu が R_FCL_BUSY に設定されます。

表 44　本コマンドが取り得るリクエスト構造体メンバ status_enu の値

状態	背景と対応	
R_FCL_OK	説明	正常終了しました
	原因	正常動作です
	処置	無し
R_FCL_BUSY※	説明	（初回 R_FCL_Execute 実行時）コマンド操作は正常に開始しました。 または実行中です
	原因	正常動作です
	処置	操作を続けてください
R_FCL_ERR_FLOW	説明	今回の FCL 関数の実行を拒否しました
	原因	FCL は、初期化を実行していない状態、または不正な状態です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_FLMD0	説明	今回の FCL 関数の実行を拒否しました
	原因	FLMD0 端子を正しく設定していません
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_WRITE	説明	書き込みが失敗しました
	原因	1.コードフラッシュが故障している可能性があります 2.書き込み先が消去した状態でない可能性があります 3.FLMD0 端子のレベルが安定した状態でない可能性があります
	処置	1.書き込み先が消去状態で無かった場合、消去を実行後、再度書き込みを実行してください 2.書き込み先が消去状態であった場合、さらなる FCL 操作を中止し、原因を調査してください 3.FLMD0 端子のレベルが安定していなかった場合、FLMD0 端子を正しく設定し、再度今回実行しようとしたコマンドを実行してください
R_FCL_ERR_ERASE	説明	消去が失敗しました
	原因	1.コードフラッシュが故障している可能性があります 2.FLMD0 端子のレベルが安定した状態でない可能性があります
	処置	1.さらなる FCL 操作を中止し、原因を調査してください 2.FLMD0 端子のレベルが安定していなかった場合、FLMD0 端子を正しく設定し、再度今回実行しようとしたコマンドを実行してください
R_FCL_ERR_REJECTED	説明	今回の FCL 関数の実行を拒否しました
	原因	他の FCL 関数、コマンドが実行中です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_INTERNAL	説明	通常発生しない予期せぬエラーです
	原因	1.FCL では原因を判別できません 2.ユーザプログラムによって、FCL が使用する内蔵 RAM を破壊している可能性があります 3.FCL が破壊されている可能性もあります
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください

※ユーザモードでのみ使用可能

4.5.21 R_FCL_CMD_GET_SERIAL_PROG_DISABLED

設定しているシリアルプログラマ接続禁止設定の設定状態（有効 ／ 無効）を読み出します。

以下にリクエスト構造体の設定方法を示します。

表 45　リクエスト構造体の設定

リクエスト構造体	値	説明
command_enu	R_FCL_CMD_GET_SERIAL_PROG_DISABLED	コマンド
bufferAdd_u32	例）データバッファ名を data の配列とした場合 &data[0]	取得するデータを格納する データバッファの先頭アドレス ≪取得値≫ 0： 対象の機能を設定していません 1： 対象の機能を設定しています
idx_u32	設定不要	
cnt_u16	設定不要	
status_enu	次表を参照	R_FCL_Execute により自動的に更新

表 46　本コマンドが取り得るリクエスト構造体メンバ status_enu の値

状態	背景と対応	
R_FCL_OK	説明	正常終了しました
	原因	正常動作です
	処置	無し
R_FCL_ERR_FLOW	説明	今回の FCL 関数の実行を拒否しました
	原因	FCL は、初期化を実行していない状態、または不正な状態です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_PARAMETER	説明	今回の FCL 関数の実行を拒否しました
	原因	設定しているパラメータが不正です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_REJECTED※	説明	今回の FCL 関数の実行を拒否しました
	原因	他の FCL 関数、コマンドが実行中です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください

※ユーザモードでのみ使用可能

4.5.22 R_FCL_CMD_SET_SERIAL_ID_ENABLED

シリアルIDを有効に設定します。この設定によりID認証のIDをシリアルIDとしても使用します。FCL関数でシリアルIDを無効状態へと変更することはできません。シリアルIDを無効状態にする場合は、専用フラッシュ・プログラマを使用してください。

以下にリクエスト構造体の設定方法を示します。

表 47　リクエスト構造体の設定

リクエスト構造体	値	説明
command_enu	R_FCL_CMD_SET_SERIAL_ID_ENABLED	コマンド
bufferAdd_u32	設定不要	
idx_u32	設定不要	
cnt_u16	設定不要	
status_enu	次表を参照	R_FCL_Execute および R_FCL_Handler により自動的に更新

ユーザモードの場合、コマンド操作中はリクエスト構造体メンバ status_enu が R_FCL_BUSY に設定されます。

表 48　本コマンドが取り得るリクエスト構造体メンバ status_enu の値

状態	背景と対応	
R_FCL_OK	説明	正常終了しました
	原因	正常動作です
	処置	無し
R_FCL_BUSY※	説明	（初回 R_FCL_Execute 実行時）コマンド操作は正常に開始しました。または実行中です
	原因	正常動作です
	処置	操作を続けてください
R_FCL_ERR_FLOW	説明	今回の FCL 関数の実行を拒否しました
	原因	FCL は、初期化を実行していない状態、または不正な状態です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_FLMD0	説明	今回の FCL 関数の実行を拒否しました
	原因	FLMD0 端子を正しく設定していません
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_WRITE	説明	書き込みが失敗しました
	原因	1.コードフラッシュが故障している可能性があります 2.書き込み先が消去した状態でない可能性があります 3.FLMD0 端子のレベルが安定した状態でない可能性があります
	処置	1.書き込み先が消去状態で無かった場合、消去を実行後、再度書き込みを実行してください 2.書き込み先が消去状態であった場合、さらなる FCL 操作を中止し、原因を調査してください 3.FLMD0 端子のレベルが安定していなかった場合、FLMD0 端子を正しく設定し、再度今回実行しようとしたコマンドを実行してください
R_FCL_ERR_ERASE	説明	消去が失敗しました
	原因	1.コードフラッシュが故障している可能性があります 2.FLMD0 端子のレベルが安定した状態でない可能性があります
	処置	1.さらなる FCL 操作を中止し、原因を調査してください 2.FLMD0 端子のレベルが安定していなかった場合、FLMD0 端子を正しく設定し、再度今回実行しようとしたコマンドを実行してください
R_FCL_ERR_REJECTED	説明	今回の FCL 関数の実行を拒否しました
	原因	他の FCL 関数、コマンドが実行中です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_INTERNAL	説明	通常発生しない予期せぬエラーです
	原因	1.FCL では原因を判別できません 2.ユーザプログラムによって、FCL が使用する内蔵 RAM を破壊している可能性があります 3.FCL が破壊されている可能性もあります
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください

※ユーザモードでのみ使用可能

4.5.23 R_FCL_CMD_GET_SERIAL_ID_ENABLED

設定しているシリアル ID の設定状態（有効 ／ 無効）を読み出します。

以下にリクエスト構造体の設定方法を示します。

表 49　リクエスト構造体の設定

リクエスト構造体	値	説明
command_enu	R_FCL_CMD_GET_SERIAL_ID_ENABLED	コマンド
bufferAdd_u32	例）データバッファ名を data の配列とした場合 &data[0]	取得するデータを格納する データバッファの先頭アドレス ≪取得値≫ 0：対象の機能を設定していません 1：対象の機能を設定しています
idx_u32	設定不要	
cnt_u16	設定不要	
status_enu	次表を参照	R_FCL_Execute により自動的に更新

表 50　本コマンドが取り得るリクエスト構造体メンバ status_enu の値

状態	背景と対応	
R_FCL_OK	説明	正常終了しました
	原因	正常動作です
	処置	無し
R_FCL_ERR_FLOW	説明	今回の FCL 関数の実行を拒否しました
	原因	FCL は、初期化を実行していない状態、または不正な状態です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_PARAMETER	説明	今回の FCL 関数の実行を拒否しました
	原因	設定しているパラメータが不正です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_REJECTED※	説明	今回の FCL 関数の実行を拒否しました
	原因	他の FCL 関数、コマンドが実行中です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください

※ユーザモードでのみ使用可能

4.5.24 R_FCL_CMD_SET_RESET_VECTOR

可変リセットベクタの値を設定します。デバイスによっては可変リセットベクタをサポートしていない場合があります。必ず、対象デバイスのユーザーズマニュアルでご確認ください。

注意：いずれかの OTP フラグがセットされると、リセットベクタを変更することはできません。

以下にリクエスト構造体の設定方法を示します。

表 51　リクエスト構造体の設定

リクエスト構造体	値	説明
command_enu	R_FCL_CMD_SET_RESET_VECTOR	コマンド
bufferAdd_u32	例）データバッファ名を data の配列とした場合 &data[0]	設定するデータを格納しているデータバッファの先頭アドレス
idx_u32	設定不要	
cnt_u16	設定不要	
status_enu	次表を参照	R_FCL_Execute および R_FCL_Handler により自動的に更新

ユーザモードの場合、コマンド操作中はリクエスト構造体メンバ status_enu が R_FCL_BUSY に設定されます。

<table>
<tr><td>設定値</td><td>unsigned long data[4] ={ 0x03020100, 0x07060504, 0x0B0A0908, 0x0F0E0D0C };
bufferAdd_u32=&data[0]</td></tr>
<tr><td>実行結果</td><td>下記の通り設定されます。

<table><tr><th>名称</th><th>値</th></tr><tr><td>リセットベクタ</td><td>0x03020100</td></tr><tr><td>予約領域1</td><td>0x07060504</td></tr><tr><td>予約領域2</td><td>0x0B0A0908</td></tr><tr><td>予約領域3</td><td>0x0F0E0D0C</td></tr></table>
本FCLはリセットベクタと予約領域1～3までの設定に対応しており、一度にリセットベクタと予約領域1～3まで設定します。なお、予約領域に関しましては、対象デバイスのユーザーズマニュアルをご確認ください。

例）RH850/F1Lの場合
リセットベクタのみに対応しています。予約領域はすべて0xFFFFFFFFで設定してください。</td></tr>
</table>

表 52　本コマンドが取り得るリクエスト構造体メンバ status_enu の値

状態	背景と対応	
R_FCL_OK	説明	正常終了しました
	原因	正常動作です
	処置	無し
R_FCL_BUSY※	説明	（初回 R_FCL_Execute 実行時）コマンド操作は正常に開始しました。または実行中です
	原因	正常動作です
	処置	操作を続けてください
R_FCL_ERR_FLOW	説明	今回の FCL 関数の実行を拒否しました
	原因	FCL は、初期化を実行していない状態、または不正な状態です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_FLMD0	説明	今回の FCL 関数の実行を拒否しました
	原因	FLMD0 端子を正しく設定していません
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_WRITE	説明	書き込みが失敗しました
	原因	1.コードフラッシュが故障している可能性があります 2.FLMD0 端子のレベルが安定した状態でない可能性があります
	処置	1.さらなる FCL 操作を中止し、原因を調査してください 2.FLMD0 端子のレベルが安定していなかった場合、FLMD0 端子を正しく設定し、再度今回実行しようとしたコマンドを実行してください
R_FCL_ERR_ERASE	説明	消去が失敗しました
	原因	1.コードフラッシュが故障している可能性があります 2.FLMD0 端子のレベルが安定した状態でない可能性があります
	処置	1.さらなる FCL 操作を中止し、原因を調査してください 2.FLMD0 端子のレベルが安定していなかった場合、FLMD0 端子を正しく設定し、再度今回実行しようとしたコマンドを実行してください
R_FCL_ERR_REJECTED	説明	今回の FCL 関数の実行を拒否しました
	原因	他の FCL 関数、コマンドが実行中です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_PROTECTION	説明	今回の FCL 関数の実行を拒否しました
	原因	保護（FHVE15 レジスタ/ FHVE3 レジスタ）している領域/設定に対して、禁止されているコマンドを実行しました
	処置	FHVE15 レジスタ/ FHVE3 レジスタの設定をご確認ください
R_FCL_ERR_INTERNAL	説明	通常発生しない予期せぬエラーです
	原因	1.FCL では原因を判別できません 2.ユーザプログラムによって、FCL が使用する内蔵 RAM を破壊している可能性があります 3.FCL が破壊されている可能性もあります 4.OTP を設定している領域に対して、禁止されているコマンドを実行しました
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください

※ユーザモードでのみ使用可能

4.5.25 R_FCL_CMD_GET_RESET_VECTOR

設定している可変リセットベクタの設定値を読み出します。

以下にリクエスト構造体の設定方法を示します。

表 53　リクエスト構造体の設定

リクエスト構造体	値	説明
command_enu	R_FCL_CMD_GET_RESET_VECTOR	コマンド
bufferAdd_u32	例）データバッファ名を data の配列とした場合 &data[0]	取得するデータを格納する データバッファの先頭アドレス ≪取得値≫ 現在設定しているリセットベクタのアドレスです
idx_u32	設定不要	
cnt_u16	設定不要	
status_enu	次表を参照	R_FCL_Execute により自動的に更新

設定値	bufferAdd_u32=&data[0]
実行結果	下記の値を取得します。 unsigned long data[4] ={ 0x03020100, 0x07060504, 0x0B0A0908, 0x0F0E0D0C }; ただし、デバイスに設定されている値を以下と仮定します。

名称	値
リセットベクタ	0x03020100
予約領域1	0x07060504
予約領域2	0x0B0A0908
予約領域3	0x0F0E0D0C

表 54　本コマンドが取り得るリクエスト構造体メンバ status_enu の値

状態	背景と対応	
R_FCL_OK	説明	正常終了しました
	原因	正常動作です
	処置	無し
R_FCL_ERR_FLOW	説明	今回の FCL 関数の実行を拒否しました
	原因	FCL は、初期化を実行していない状態、または不正な状態です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_PARAMETER	説明	今回の FCL 関数の実行を拒否しました
	原因	設定しているパラメータが不正です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_REJECTED※	説明	今回の FCL 関数の実行を拒否しました
	原因	他の FCL 関数、コマンドが実行中です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください

※ユーザモードでのみ使用可能

4.5.26 R_FCL_CMD_GET_BLOCK_CNT

コードフラッシュの総ブロック数を取得します。

以下にリクエスト構造体の設定方法を示します。

表 55　リクエスト構造体の設定

リクエスト構造体	値	説明
command_enu	R_FCL_CMD_GET_BLOCK_CNT	コマンド
bufferAdd_u32	例）データバッファ名を data の配列とした場合 &data[0]	取得するデータを格納する データバッファの先頭アドレス ≪取得値≫ 対象デバイスのコードフラッシュのブロック総数です
idx_u32	設定不要	
cnt_u16	設定不要	
status_enu	次表を参照	R_FCL_Execute により自動的に更新

表 56　本コマンドが取り得るリクエスト構造体メンバ status_enu の値

状態	背景と対応	
R_FCL_OK	説明	正常終了しました
	原因	正常動作です
	処置	無し
R_FCL_ERR_FLOW	説明	今回の FCL 関数の実行を拒否しました
	原因	FCL は、初期化を実行していない状態、または不正な状態です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_PARAMETER	説明	今回の FCL 関数の実行を拒否しました
	原因	設定しているパラメータが不正です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_REJECTED※	説明	今回の FCL 関数の実行を拒否しました
	原因	他の FCL 関数、コマンドが実行中です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください

※ユーザモードでのみ使用可能

4.5.27 R_FCL_CMD_GET_BLOCK_END_ADDR

指定したブロックの最終アドレスを取得します。

以下にリクエスト構造体の設定方法を示します。

表 57　リクエスト構造体の設定

リクエスト構造体	値	説明
command_enu	R_FCL_CMD_GET_BLOCK_END_ADDR	コマンド
bufferAdd_u32	例）データバッファ名を data の配列とした場合 &data[0]	取得するデータを格納する データバッファの先頭アドレス ≪取得値≫ 指定したブロックの最終アドレスです
idx_u32	「0」～ 「デバイスのブロック数－1」	ユーザ領域のブロック番号
	「0x80000000」～ 「0x80000000 ＋ 拡張ユーザ領域のブロック数－1」	拡張ユーザ領域のブロック番号
cnt_u16	設定不要	
status_enu	次表を参照	R_FCL_Execute により自動的に更新

表 58　本コマンドが取り得るリクエスト構造体メンバ status_enu の値

状態	背景と対応	
R_FCL_OK	説明	正常終了しました
	原因	正常動作です
	処置	無し
R_FCL_ERR_FLOW	説明	今回の FCL 関数の実行を拒否しました
	原因	FCL は、初期化を実行していない状態、または不正な状態です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_PARAMETER	説明	今回の FCL 関数の実行を拒否しました
	原因	設定しているパラメータが不正です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_REJECTED※	説明	今回の FCL 関数の実行を拒否しました
	原因	他の FCL 関数、コマンドが実行中です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください

※ユーザモードでのみ使用可能

4.5.28 R_FCL_CMD_GET_DEVICE_NAME

使用しているデバイスのデバイス名を取得します。

以下にリクエスト構造体の設定方法を示します。

表 59　リクエスト構造体の設定

リクエスト構造体	値	説明
command_enu	R_FCL_CMD_GET_DEVICE_NAME	コマンド
bufferAdd_u32	例）データバッファ名を data の配列とした場合 &data[0]	取得するデータを格納する データバッファの先頭アドレス ≪取得値≫ デバイス名です
idx_u32	設定不要	
cnt_u16	設定不要	
status_enu	次表を参照	R_FCL_Execute により自動的に更新

表 60　本コマンドが取り得るリクエスト構造体メンバ status_enu の値

状態	背景と対応	
R_FCL_OK	説明	正常終了しました
	原因	正常動作です
	処置	無し
R_FCL_ERR_FLOW	説明	今回の FCL 関数の実行を拒否しました
	原因	FCL は、初期化を実行していない状態、または不正な状態です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_PARAMETER	説明	今回の FCL 関数の実行を拒否しました
	原因	設定しているパラメータが不正です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください
R_FCL_ERR_REJECTED※	説明	今回の FCL 関数の実行を拒否しました
	原因	他の FCL 関数、コマンドが実行中です
	処置	さらなる FCL 操作を中止し、根本的な原因を調査してください

※ユーザモードでのみ使用可能

第 5 章 FCLの設定方法と使用方法

本章には、FCL を動作させる方法と、ユーザのアプリケーションに組み込む方法に関する重要な情報を記載しています。ライブラリの問題や誤動作を回避するために、本章および「6 章 注意事項」をよくお読みください。ライブラリをユーザのプロジェクトに組み込む前に、FCL 使用方法など必ず読んで理解してください（2 章と 3 章参照）

5.1 FCLの入手

本 FCL、本ユーザマニュアルは、常に最新バージョンのご使用を推奨します。

5.2 ファイル構成

対象デバイスでの FCL の実装方法と使用方法を紹介するサンプルアプリケーションと FCL を含むコンパイル可能なサンプルプロジェクトとして、本 FCL は供給されます。

5.2.1 概要

FCL とサンプルアプリケーションに関するファイルを次の図に示します。

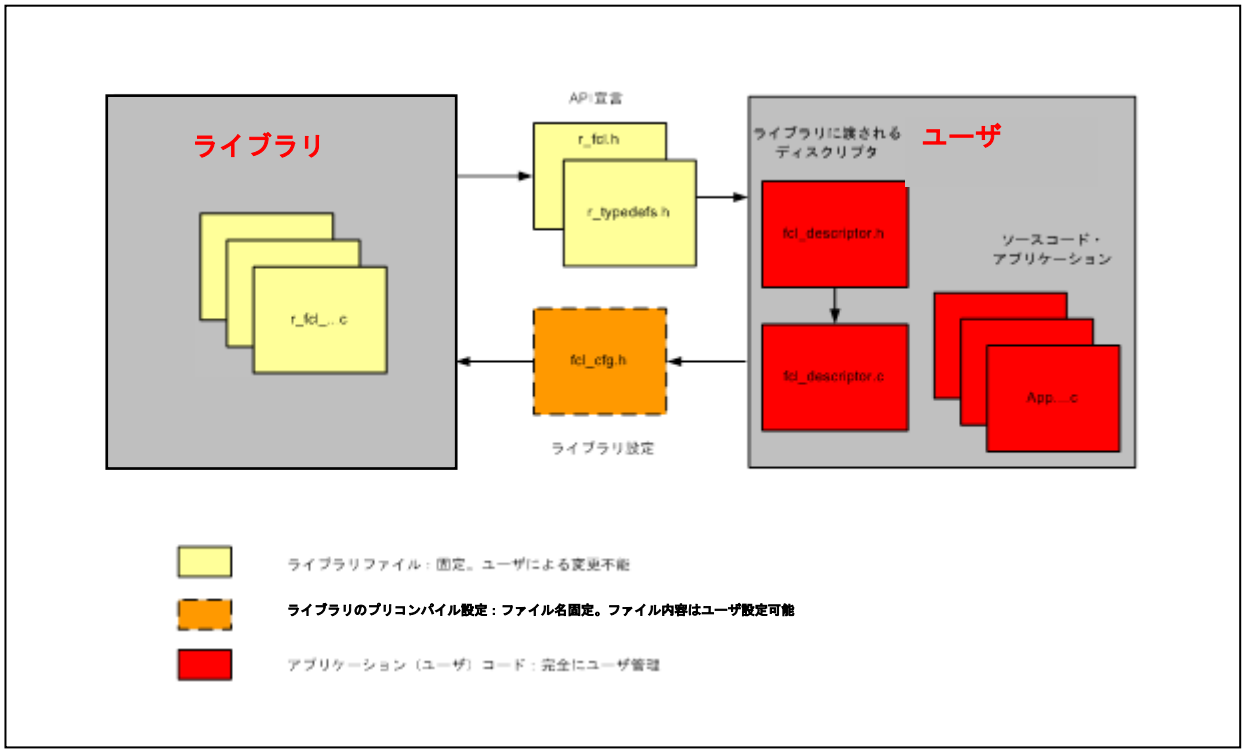

図 7　FCL とアプリケーションのファイル構成

FCL コードは r_fcl_...で始まる各種ソースファイルで構成されています。これらのファイルをユーザが自由に変更することはできません。

ライブラリはソースコードで提供されているため、ライブラリをコンパイルのための設定をする必要があります。ファイル fcl_cfg.h にはそのための定義が含まれています。このファイルはライブラリのソースファイルの 1 つなので、ファイル内容はユーザにより変更可能ですが、ファイル名は変更できません。

5.2.2 パッケージのファイルシステム構成

FCL のインストーラを使ってインストールされる全ファイルを表 61 に示します。

- 赤字のファイルはビルド環境関連のファイルで、コンパイル、リンク、ターゲットのビルド処理を制御します。
- 青字のファイルはサンプルアプリケーションに関連のファイルです。
- 緑字のファイルは説明のみです。
- 黒字のファイルは FCL 関連のファイルです。

表 61　FCL パッケージのファイル構成

★

ファイル名		内容
<installation_folder>/FCL		
Release.txt		リリースノート
support.txt		サポートデバイス一覧
<installation_folder>/FCL /<compiler>/<device_name>		
Build.bat		FCL のサンプルアプリケーションをビルドするためのバッチファイル
Clean.bat		FCL のサンプルアプリケーションを削除するためのバッチファイル
Makefile		ビルド処理と削除処理を制御する Make ファイル
<installation_folder>/FCL /<compiler>/<device_name>/Sample(1)		
dr7f70xxxx_startup.850(1)	<GHS 用>	デバイスとコンパイラに固有のスタートアップコード
cstart.asm	<CC-RH 用>	
dr7f70xxxx.ld(1),	<GHS 用>	コンパイラ固有のリンクディレクティブ
dr7f70xxxx_pic.dir(1),	<CC-RH 用>	
dr7f70xxxx.dvf.h(1) dr7f70xxxx_irq.h(1)	<GHS 用>	デバイス固有のヘッダファイル <GHS 用>は"dr7f70xxxx.dvf.h(1) "、または"dr7f70xxxx_0.h(1), "と"io_macros_v2.h"を使用します。 <CC-RH 用>は"boot.asm"、または"vecttbl.asm".を使用します。
iodefine.h boot.asm	<CC-RH 用>	
main.c fcl_ctrl.c(2)		サンプルアプリケーションのコード
target.h		ターゲットマイクロコントローラの初期化コード
fcl_prefetch.850	<GHS 用>	プリフェッチ領域の初期化(3)
fcl_cfg.h		FCL のユーザ定義の設定
fcl_descriptor.c		サンプルアプリケーションで使用する FCL のディスクリプタ

ファイル名		内容
fcl_descriptor.h		サンプルアプリケーションで使用する FCL のディスクリプタ
fcl_user.c fcl_user.h		サンプルアプリケーションで使用するユーザファイル
<installation_folder>/FCL /<compiler>/FCL		
r_fcl.h		FCL の API の定義
r_fcl_types.h		ユーザインタフェースによる型定義、およびセルフプログラミング中に使用されるすべてのエラーコードと状態コード
<installation_folder>/FCL /<compiler>/FCL/lib		
r_typedefs.h		FCL で使用される C 言語型定義
r_fcl_env.h		FCL の内部定義
r_fcl_global.h		セルフプログラミング中に使用されるグローバル変数と設定
r_fcl_hw_access.c r_fcl_user_if.c		FCL の主要ソースコード
r_fcl_hw_access_asm.850	<GHS 用>	コンパイラ固有のアセンブラコード
r_fcl_hw_access_asm.asm	<CC-RH 用>	

(1) ファイル名は、選択したデバイス型名に依存します。ここで示すファイル名は、RH850/F1L の例です。

例) デバイスが R7F701007 の場合、dr7f70xxxx_startup.850 は、dr7f701007_startup.850 です。

(2) アプリケーション制御のファイル(fcl_ctrl.c)は、FCL V2.13 以降のサンプルで使用するファイルです。FCL V2.12 以前のバージョンでは、この処理を main.c で行っているため、本ファイルは不要です。

(3) プリフェッチ領域の初期化(fcl_prefetch.850)は、FCL V2.12 以前のバージョンで使用する GHS コンパイラ(リンカ)専用のファイルです。FCL V2.13 以降、プリフェッチ領域の初期化を r_fcl_hw_access_asm.850 で行っているため、本ファイルは不要です。CC-RH コンパイラ(リンカ)では、プリフェッチ領域の初期化を r_fcl_hw_access_asm.asm で行っているため、本ファイルは不要です。

(4) RH850 コードフラッシュライブラリ(FCL) Type01 に同梱のバッチファイルに記載の make.exe ファイルは外部ツールで、make.exe を提供しているサイトからダウンロードする必要があります。添付 release.txt に記述の通り、サンプルアプリケーションは、GNU Make を使用して動作確認をしています。同等の環境でご使用いただく場合は、GNU の Web サイトから make.exe をダウンロード、インストールし、添付のバッチファイルを実行してください。

5.3 リンカセクション

以下に示すセクションは FCL に関連しており、リンカファイルで定義される必要があります（例については、サンプルアプリケーション内のリンクディレクティブ・ファイルを参照してください）。

FCL のデータセクション	
R_FCL_DATA	FCL の内部変数です。内蔵 RAM に配置できます。

FCL のコンストセクションとコードセクション	
R_FCL_CONST	FCL の内部定数データです。
R_FCL_CODE_ROM	セルフプログラミングの最初に実行されるコードが入っています。このコードはリンク先の位置から実行されます。主に初期化コードです。
R_FCL_CODE_USRINT	コードフラッシュが使用できないときに FCL 動作と平行して実行可能なユーザの割り込みルーチンが入っています。
R_FCL_CODE_USR	コードフラッシュが使用できないときに FCL 動作と平行して実行する必要のあるユーザコードが入っています。
R_FCL_CODE_RAM	コマンド操作を扱う FCL コードの一部が入っており、このためコードフラッシュ領域の外に置く必要があります。
R_FCL_CODE_ROMRAM	ユーザインタフェースが入っています。FCL の設定（状態チェックのモード）に応じて、このセクションのコードは内蔵 RAM（状態チェックをユーザモードで行う場合）またはコードフラッシュ（状態チェックをインターナルモードで行う場合）で実行されます。
R_FCL_CODE_RAM_EX_PROT	この小セクションを内蔵 RAM にコピーします。このセクションには、CPU が直前のセクションの終わり付近のコードを実行しているときの ECC によるプリフェッチ例外を避ける目的があります。

サンプルアプリケーションのセクション	
R_FCL_RESERVE	FCL のセクションがコピーされるための内蔵 RAM 空間を確保する方法の例です。次のセクションは、サンプルアプリケーションでこのセクションにコピーされます 。 R_FCL_CODE_USRINT， R_FCL_CODE_USR， R_FCL_CODE_RAM，R_FCL_CODE_ROMRAM，R_FCL_CODE_RAM_EX_PROT の各セクションが、このセクションにコピーされます。

注意：

1. セクションの順序を変更すること、または FCL セクションの間に他のセクションを置くことはできません。そのようにした場合、FCL ライブラリが正常に動作しません。アラインメントのためにセクション間に空き領域を設けることは可能です。但し、本 FCL の対象デバイスによっては、プリフェッチにより空き領域にアクセスし ECC 例外が発生する場合がありますので、適切な処置をしてください。
2. セクションが空であっても、定義する必要があります。
3. セクション R_FCL_CODE_USRINT および R_FCL_CODE_USR はユーザコード専用です。ユーザは他の FCL セクションにコードまたはデータを配置してはいけません。

5.4 サンプルアプリケーション

FCL をユーザアプリケーションに正しく実装するには、コードフラッシュと FCL を理解することがとても重要です。したがって、このユーザマニュアルを FCL ご使用前に読むことが大切です。最良の方法は、ユーザマニュアルをお読みになられた後、FCL のサンプルアプリケーションを試してみることです。

FCL の動作確認時に使用したビルドオプションは Make ファイル内に記述されている内容です。但し、本計測値は、参考として必要な容量の目安としてお使いいただくことを目的とし、より使用量の多い状態での計測行うために、一部、レジスタモードを 32 に変更して計測している場合があります。お客様アプリケーションでの使用量は、アプリケーションに組み込んだ状態での各セクションの値をご確認ください。

また、下記に動作確認時に計測した FCL V2.13 が使用する ROM/RAM 容量を参考に記載します。

★

項目名	サイズ　[バイト]	
	GHS （22 レジスタ・モード）	CC-RH （22 レジスタ・モード）
R_FCL_CODE_ROMセクション	740	696
R_FCL_CODE_ROMRAMセクション	3072	3172
R_FCL_CODE_RAMセクション	5368	4264
R_FCL_CONSTセクション	20	20
R_FCL_DATAセクション	172	172
R_FCL_CODE_RAM_USRINTセクション	ユーザプログラム依存※	
R_FCL_CODE_RAM_USRセクション		
FCLスタック最大使用容量	96	68

注意：

1. ユーザプログラムを配置するセクションの為、サイズはユーザプログラムに依存します。

   ライブラリに付属のサンプルプログラムは、"R_FCL_SUPPORT_OTP"(プリコンパイル設定)が未定義としており、上記[R_FCL_CODE_RAM セクション]サイズとは異なります(小さくなります)のでご注意ください。

2. FCL V2.13 以降、CC-RH コンパイラは V2.xx を採用しました。これにより、関数・定数・変数をリンク時とは異なる任意のアドレスに配置できる PIC/PID 機能をサポートした PIC オプションを設定、CopySection 関数でライブラリの一部を RAM に転送することを前提に付属のサンプルプログラムを作成しています。RAM へ転送する必要のあるユーザプログラムもこの CopySection 関数で転送可能ですが、転送されるユーザプログラム内で定数データを参照している場合、その定数データも RAM へ転送する必要があります。定数データは、参照する関数との相対アドレスを維持して、ユーザ処理にて RAM へ転送する必要がありますのでご注意ください。

5.5 セルフプログラミング・シーケンス

次のフローチャートに、デバイス動作時の標準的な FCL シーケンスを使用される API 関数とともに示します。簡略化するため、このフローチャートではエラー処理について詳述していません。

5.5.1 標準フローチャート（ユーザモード）

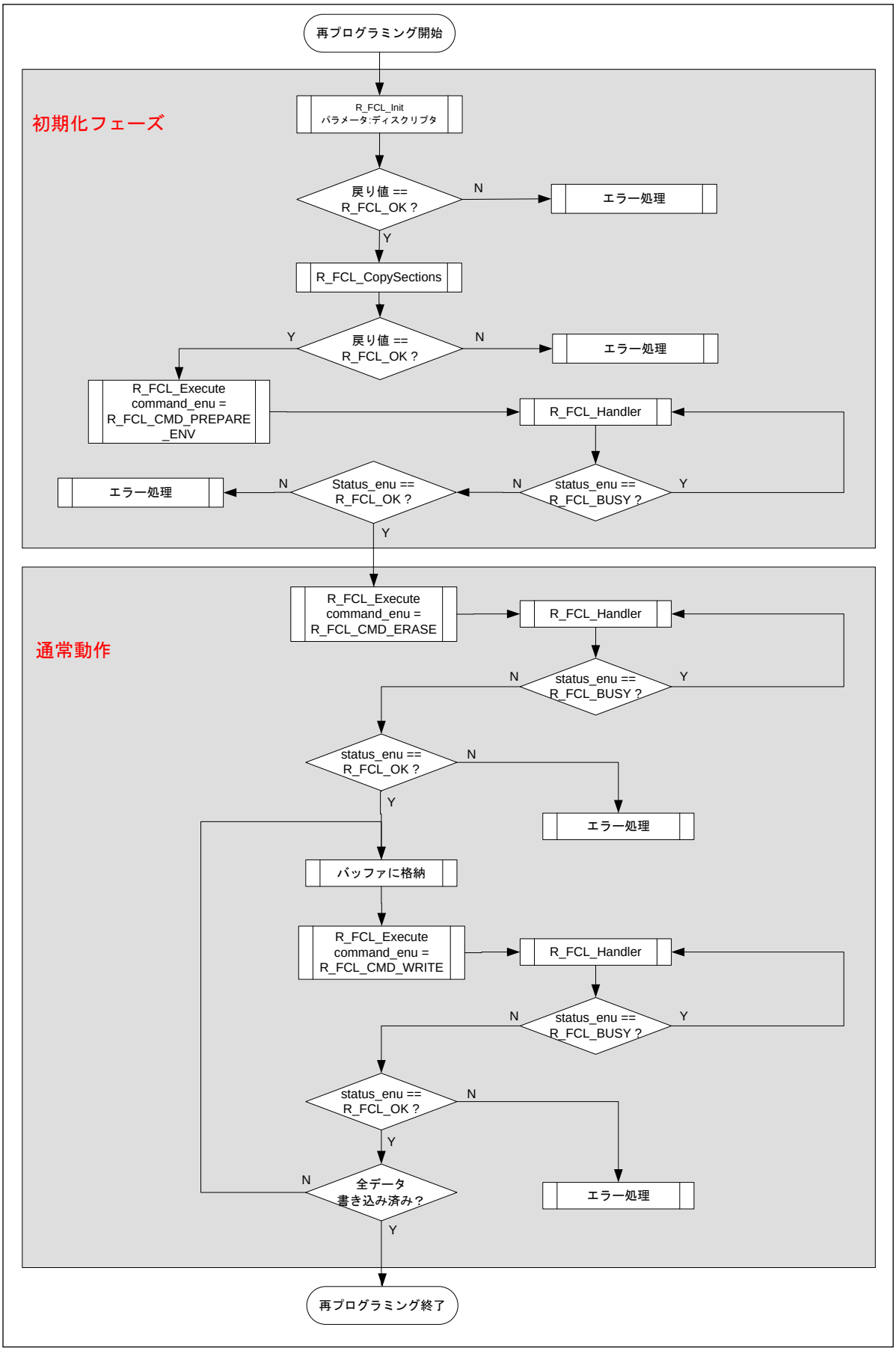

図 8　標準フローチャート（ユーザモード）

5.5.2 標準フローチャート（インターナルモード）

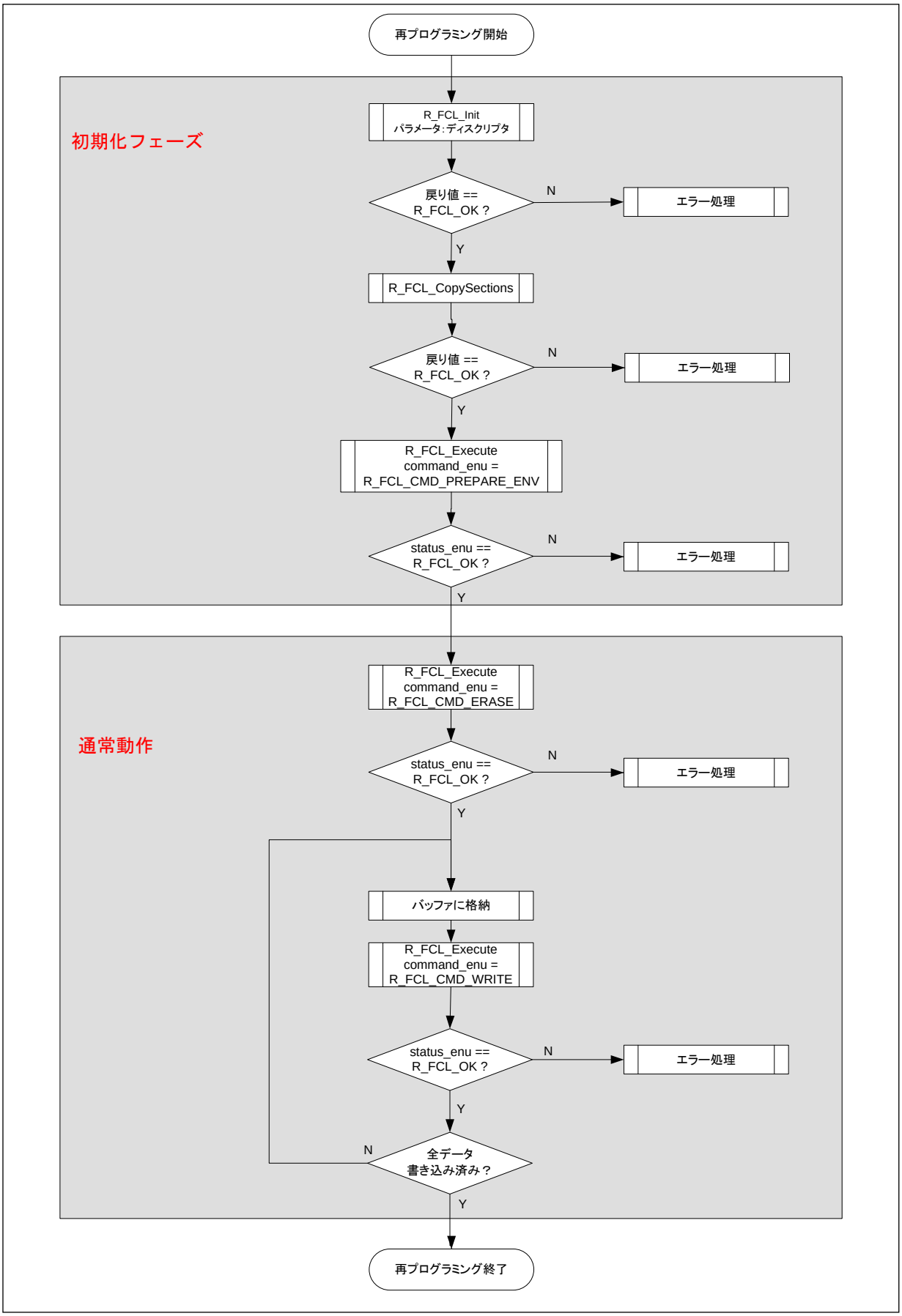

図 9　標準フローチャート（インターナルモード）

5.6 MISRA C対応

本 FCL コードは MISRA C に対応しております。
使用したツールは、MISRA C 2004 標準規則の検査用の QAC ソースコードアナライザです。

注：
"MISRA", "MISRA C"は、MISRA Consortium を代表して HORIBA MIRA 社が保有する登録商標です。 "QA C"は Programming Research 社の登録商標です。

第 6 章 注意事項

本 FCL を使用する前に必ず以下の注意事項をご参照ください。

1. 動作周波数の設定：

・FCL を使用する場合、CPU を PLL クロックで動作させてください。FCL では指定した CPU 動作周波数が RH850 製品毎の PLL クロックの下限・上限内に収まっているかをチェックしています。なお、デバイスによってはコードフラッシュに対する操作に対し、CPU 動作周波数に制限がある場合があります。また、FCL 実行中に CPU の周波数を変更しないでください。周波数の変更が必要な場合は、CPU 周波数を変更後、再度初期化から実行してください。

★ ・RH850/F1KM-S4, RH850/F1KH-D8 使用時、FCL で設定しているフラッシュシーケンサ動作周波数は、初期値として CPU 動作周波数(Max 240MHz で使用する CPU クロック)の 1/8 の周波数を設定しています。

fPCLK = 1/8 fCPUCLK_H（CKDIVMD=1 : CPU 動作周波数 : Max 240MHz）

RH850/F1KM-S4, RH850/F1KH-D8 のオプションバイトを CKDIVMD=0（Max 120MHz）に設定して使用する場合は、フラッシュシーケンサ動作周波数を CPU 動作周波数の 1/4 の周波数に切り替えて使用する必要があります。

fPCLK = 1/4 fCPUCLK_H（CKDIVMD=0 : CPU 動作周波数 : Max 120MHz）

FCL V2.13 以降、フラッシュシーケンサ動作周波数を切り替えて、Max 120MHz に対応することが可能です。

切り替え方法は、FCL の”r_fcl_hw_access.c”ファイルの”R_FCL_FCUFct_SetFrequency” 関数内で設定している周波数比の分母を 1/2 にする行を有効にします。

具体的には、”Changing CKDIVMD” のキーワードの後のコメント行に続く、”#if 0”の行を”#if 1”に変更します。

<キーワード>

```
/***********************************************************************
* SAMPLE: Changing CKDIVMD
***********************************************************************/
```

```
#if 0
ffcu = ( (ffcu + (g_fcl_data_str.deviceInfo_str.fDivider_u8 / 2u)) - 1u) /
        (g_fcl_data_str.deviceInfo_str.fDivider_u8 / 2u);
#else
```

⇩

```
#if 1
ffcu = ( (ffcu + (g_fcl_data_str.deviceInfo_str.fDivider_u8 / 2u)) - 1u) /
        (g_fcl_data_str.deviceInfo_str.fDivider_u8 / 2u);
#else
```

RH850/F1KM-S4, F1KH-D8をCKDIVMD=0(Max 120MHz)で使用する場合に限り、**”#if 1”**に変更する。

※FCL V2.12 以前は CKDIVMD=0（Max 120MHz）の設定には対応していません。

2. CPU：

・FCL はスーパバイザモード（CPU 動作モード）でのみ使用できます。ユーザモードでは使用できません。

・RH850 デュアル CPU の製品では、メイン CPU（CPU1）でのみ本製品の使用が可能です。パフォーマンス CPU(CPU2)での本製品の使用は禁止です。また、メイン CPU（CPU1）で FCL のコマンドを使用する時、パフォーマンス CPU(CPU2)を停止させてください。CPU1 で FCL のコマンドを実行中、CPU1 からコードフラッシュにアクセスできないだけでなく、CPU2 からもコードフラッシュにアクセスできません。

3. 関数のリエントラント：

本 FCL のすべての関数はリエントラントではありません。したがって、いずれの FCL 関数に対してもリエントラントな呼び出しを避ける必要があります。

4. 関数の同時実行：

本 FCL は多重実行に対応していません。

・FCL 関数を割り込み処理内で実行しないでください。

・OS 上で FCL 関数を実行する場合は、複数のタスクから FCL 関数を実行しないでください。

5. セルフプログラミング中のコードフラッシュおよびデータフラッシュへのアクセス：

・セルフプログラミングが実行中はコードフラッシュおよびデータフラッシュにアクセスすることはできません。コードフラッシュおよびデータフラッシュにアクセスすると、不正な動作となります。

・FCL コマンドの動作中はコードフラッシュを使用できないため、FCL の実行に関係するすべてのコードを内蔵 RAM 内に配置してください。また、標準ライブラリは使用しないでください。標準ライブラリが内蔵 ROM に配置されている場合、FCL 実行中に内蔵 ROM にアクセスしてしまうことが考えられます。標準ライブラリの使用に関しましては必ずビルドオプションの設定をご確認ください。オプションのデフォルト設定が標準ライブラリの使用を許可に設定している場合があります。

6. セルフプログラミング中の割り込み：

・コマンド実行中でも、割り込み処理を使用することが可能です。ただしコマンド実行中は、内蔵 ROM 上のユーザプログラムと割り込みベクタは使用できません。デバイスの「例外ハンドラ・アドレス切り替え機能」により、割り込みベクタテーブルとプログラムを内蔵 RAM に設定する必要があります。設定後は、通常の割り込み処理と同じ動作です。なお、割り込みテーブルの変更方法の詳細は対象デバイスのユーザーズマニュアルをご参照ください。

・ユーザプログラムにおいて「例外ハンドラ・アドレス切り替え機能」により割り込みテーブルを変更する際は、切り替え手順の開始から完了までの間、割り込みが発生しない、または発生しても問題が無いように考慮してください。

7. コマンド操作中の中断：

・書き込み処理中、または消去処理中にリセット/電源瞬断が発生、もしくは R_FCL_CancelRequest 関数を実行した場合、処理は中断され、書き込み処理、または消去処理を実行していたブロックの内容は不定となります。

上記の処理中断により不定となった領域を再度ご使用になる場合は、必ず再度消去処理を実行し、ご使用ください。

・コマンド実行中に、HALTモード以外の低消費電力で動作するモードへの移行することも禁止です。低消費電力で動作するモードについては、対象デバイスのユーザーズマニュアルでご確認ください。

8. 書き込み動作：

対象ブロックに対する書き込みは、事前に対象ブロックの内容を消去し、実行してください。書き込み済みの領域に対する上書きは禁止です。

9. ウォッチドッグタイマ：

ウォッチドッグタイマは、FCL の実行中は停止しません。 なお、FCL においてはタイムアウト処理を行っていません。FCL 関数のタイムアウト処理が必要な場合は、ユーザプログラムでウォッチドッグタイマなど使用して処理を行ってください。

10. FCL 動作の前提条件：

あらゆる FCL 動作（R_FCL_CMD_PREPARE_ENV を除くあらゆるコマンド）の開始前に、以下の初期化シーケンスを実行する必要があります。

・FCL の初期化（ディスクリプタに正しいパラメータを設定して R_FCL_Init 関数の実行）

・プログラムの内蔵 RAM へのコピー（R_FCL_CopySections 関数の実行など）

・FLMD0 端子を「1」に設定

・FCL の実行環境の準備

（R_FCL_Execute 関数による R_FCL_CMD_PREPARE_ENV コマンドの実行）

11. デュアルオペレーション：

FCL と FDL の同時実行はできません。

12. コマンド実行中のリクエスト構造体：

コマンド実行中にリクエスト構造体の内容を変更した場合、FCLは誤動作します。

13. データアライメント：

・書き込みを開始するコードフラッシュの先頭アドレスは256バイトアラインです。

・ユーザアプリケーションのデータデータバッファのアドレスは4バイトアラインの必要があります。

14. サスペンド動作中のキャンセル：

すでに前の動作がサスペンドしている状態で書き込み、および消去を実行中にキャンセルコマンドを実行した場合、現在実行中の書き込み、および消去とサスペンド以前の動作の両方がキャンセルされます。

15. 新しい ID の設定：

R_FCL_CMD_SET_ID コマンドによって新しいIDを設定した場合、設定した新しいIDを有効にするには、リセットが必要です

16. サスペンドのネスト動作：

サスペンド機能はサスペンド状態中にサスペンドを実行することと、以下のシーケンスを禁止しています。
≪禁止シーケンス≫
・R_FCL_CMD_ERASEコマンド --> サスペンド状態 --> R_FCL_CMD_ERASEコマンド
・R_FCL_CMD_WRITEコマンド --> サスペンド状態 --> R_FCL_CMD_ERASEコマンド
・R_FCL_CMD_WRITEコマンド --> サスペンド状態 --> R_FCL_CMD_WRITEコマンド

以下のシーケンスのみ許可されます。
≪許可シーケンス≫
・R_FCL_CMD_ERASEコマンド --> サスペンド状態 --> R_FCL_CMD_WRITEコマンド
（ただし、R_FCL_CMD_WRITEコマンドは消去中のブロック以外に対してのみ許可されます。）

できる限りネスト動作を行わないことを推奨します。

17. プリコンパイルオプション設定時のアクセス領域：

本FCLはプリコンパイルオプション設定によって、初期化時に以下の領域にアクセスします。
R_FCL_ExecuteのPrepareEnvコマンド実行時は、以下の領域のリードアクセスを許可してください。

FCL V2.12 以降 プリコンパイル定義	アクセス領域
R_FCL_NO_BFA_SWITCH	0x01030000～0x0103029F
R_FCL_MIRROR_FCU_COPY	0x01030000～0x0103029F, 0x01037000～0x01037FFF
R_FCL_NO_FCU_COPY	0x00010000～0x0001029F
上記プリコンパイル定義なし	0x00010000～0x0001029F, 0x00017000～0x00017FFF

★ 18. RH850/F1KM,F1KH のセルフプログラミング ID 認証モードの対応：

本FCLは、RH850/F1KM,F1KHのオプションバイト1(OPBT1)のSIDAMビット=”1” (セルフプログラミングID 認証要)のみに対応しており、SIDAMビット= “0”(セルフプログラミングID 認証不要)のモードには対応しておりません。

付録A　改訂記録

A. 1　本版で改訂された主な箇所

Rev.	発行日	改訂内容	
		ページ	ポイント
1.02	2019.05.31	全体	R_FCL_CMD_PREPARE_ENV コマンドの注意に該当する新サポートデバイスを追加(F1KH, D1M1-V2, D1S1)。
		17	「4. 1　プリコンパイル設定」の注意事項で、プリコンパイル定義毎に対象となる新サポートデバイスを追加。
		18	「4. 3　データ型」に V2.13 以降のバージョンを C99 でコンパイルする場合の注意を追加。
		19	「4. 3. 1　単純型定義」に C99 以降の規格が指定されている場合の説明を追加。
		86	「5.2.2　パッケージのファイルシステム構成」補足(2),(3)で一部のファイル構成変更内容を追加。
		88	「5. 4　サンプルアプリケーション」の ROM/RAM 容量を対象バージョン FDL V2.13 用に変更。
		88	「5. 4　サンプルアプリケーション」の「注意」に CopySection 関数使用時のユーザ定数データ転送に関する注意事項を追加。
		92	「第 6 章　注意事項」に「動作周波数の設定」項目を追加、内容に RH850/F1KM-S4, または、RH850/F1KH-D8 のオプションバイトで CKDIVMD=0（Max 120MHz）に設定して使用する場合の注意事項を追加。
		95	「第 6 章　注意事項」に「18.セルフプログラミング ID 認証モードの対応」を追加。

A. 2　前版までの改版履歴

これまでの改版履歴を次に示します。

Rev.	発行日	改訂内容 ポイント
1.00	2015.03.31	初版発行
1.01	2017.12.25	「第 1 章　はじめに」の「備考」に文言を追加。
		「3.1 内蔵 RAM でのコード実行」に補足を追加。
		「3.6 タイムアウト処理」を追加。
		「表 2　プリコンパイルオプション」に新機能を追加。
		「4. 3　データ型」に C99 でコンパイルする場合の注意を追加。
		「4.3.2　r_fcl_command_t」の使用可能なコマンド一覧で、PREPARE_ENV コマンドの説明に追加デバイスでの「注意」を追加
		「4.5.1 R_FCL_CMD_PREPARE_ENV」にデバイス仕様差分と新機能を追加。
		「4.5.1 R_FCL_CMD_PREPARE_ENV」 「4.5.4 R_FCL_CMD_SET_LOCKBIT」 「4.5.5 R_FCL_CMD_GET_LOCKBIT」 「4.5.8 R_FCL_CMD_SET_OTP」 「4.5.10 R_FCL_CMD_SET_OPB」 「4.5.12 R_FCL_CMD_SET_ID」 「4.5.14 R_FCL_CMD_SET_READ_PROTECT_FLAG」 「4.5.16 R_FCL_CMD_SET_WRITE_PROTECT_FLAG」 「4.5.18 R_FCL_CMD_SET_ERASE_PROTECT_FLAG」 「4.5.20 R_FCL_CMD_SET_SERIAL_PROG_DISABLED」 「4.5.22 R_FCL_CMD_SET_SERIAL_ID_ENABLED」 「4.5.24 R_FCL_CMD_SET_RESET_VECTOR」 で status_enu の取り得る値から R_FCL_ERR_REJECTED の"※"を削除。
		「4.5.6 R_FCL_CMD_ENABLE_LOCKBITS」 「4.5.11 R_FCL_CMD_GET_OPB」 で status_enu の取り得る値から R_FCL_BUSY を削除。
		「表 61　FCL パッケージのファイル構成」のファイル名、内容を変更。(3)に make.exe に関する注意書きを追加。
		「5.4 サンプルアプリケーション」の ROM/RAM 容量、処理時間に関する記述を変更。
		「第 6 章 注意事項」の「1.CPU」に新デバイス RH850/F1KM-S4 の注意事項追加。
		「第 6 章 注意事項」に「16.プリコンパイルオプション設定時のアクセス領域」を追加。

RH850 ファミリ　ユーザーズマニュアル
コードフラッシュライブラリ Type01

発行年月日　2015年 3月31日　Rev.1.00
2019年 5月31日　Rev.1.02

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）

RH850 ファミリ

ルネサス エレクトロニクス株式会社

R01US0150JJ0102