User's Manual

16

RL78 ファミリ

データ・フラッシュ・ライブラリ Type04
日本リリース版

インストーラ名：RENESAS_RL78_FDL_T04_xVxx

16 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.1.10　2023.12

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

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，RL78ファミリのデータ・フラッシュ・ライブラリ Type04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。

対応MCU：以下のリストを参照してください。

日本語版：

マイコン対応セルフプログラミングライブラリ(日本リリース版)一覧(R20UT2741)、及び

RL78ファミリ セルフプログラミングライブラリ セルフRAMリスト(R20UT2943)

目　　的　このユーザーズマニュアルは，RL78ファミリのデータ・フラッシュ・メモリの書き換えを行うために使用するデータ・フラッシュ・ライブラリ　Type04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。

・概要説明

・プログラミング環境

・データ・フラッシュ・ライブラリ関数

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3章　データ・フラッシュ・ライブラリ関数**を参照してください。

なお，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	：×××（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　概　説

1.1　概　要

データ・フラッシュ・ライブラリは、RL78マイクロコントローラに搭載された機能を使用し、データ・フラッシュ・メモリへの操作を行うためのソフトウェア・ライブラリです。

データ・フラッシュ・ライブラリはユーザプログラムから呼び出すことにより、データ・フラッシュ・メモリの書き換えや読み出しを実行します。

なお、本ユーザーズマニュアルは、対象のRL78マイクロコントローラのユーザーズマニュアルと合わせてご使用ください。

用　語　このマニュアルで使用する用語について、その意味を次に示します。

・データ・フラッシュ・ライブラリ
RL78マイクロコントローラが提供する機能によるデータ・フラッシュ・メモリ操作のためのライブラリです。
コード・フラッシュ・メモリへの操作はできません。

・フラッシュ・セルフ・プログラミング・ライブラリ
RL78マイクロコントローラが提供する機能によるコード・フラッシュ・メモリ操作のためのライブラリです。
データ・フラッシュ・メモリへの操作はできません。

・EEPROMエミュレーション・ライブラリ
搭載されているフラッシュ・メモリへEEPROMのようにデータを格納させるための機能を提供するライブラリです。

・ブロック番号
フラッシュ・メモリのブロックを示す番号です。データ・フラッシュ・ライブラリ Type04では消去の操作単位です。

・内部ベリファイ
フラッシュ・メモリへの書き込み後、フラッシュ・メモリのセルの信号レベルが適正であるかを確認することです。
内部ベリファイでエラーとなった場合、そのフラッシュ・メモリは正常では無いと判断されます。ただし、内部ベリファイ・エラー後に、再度、データの消去→書き込み→内部ベリファイを実行し、正常終了した場合には、そのフラッシュ・メモリは正常であると判断します。

・FDL
データ・フラッシュ・ライブラリの略称です。

・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シーケンサと呼びます。

・BGO (バック・グラウンド・オペレーション)

シーケンサにフラッシュ・メモリの制御を任せることによって、ユーザプログラムを動作させながら、フラッシュ・メモリの書き換えを実行できる状態のことです。概要や詳細については2. 1　ハードウェア環境、および 3. 4　BGO（バック・グラウンド・オペレーション）の項を参照してください。

・ステータス・チェック

シーケンサを使用する場合、フラッシュ･メモリの制御を行うプログラムでシーケンサの状態(フラッシュ・メモリに対する制御の状態)を確認し、継続実行に必要な設定を行う処理が必要です。この処理を本書ではステータス・チェックと呼びます。

1. 2　データ・フラッシュ・ライブラリ Type04の呼び出し方法

データ・フラッシュ・ライブラリ Type04を使用してデータ・フラッシュ・メモリの書き換えを行うためには、データ・フラッシュ・ライブラリ Type04の初期化処理や、使用する機能に対応する関数をC言語、ならびにアセンブリ言語のどちらかでユーザプログラムから実行する必要があります。

図1－1にデータ・フラッシュ・ライブラリ Type04の状態遷移図を、図1－2にデータ・フラッシュ・ライブラリ Type04を利用したデータ・フラッシュ・メモリ書き換えフロー例を示します。

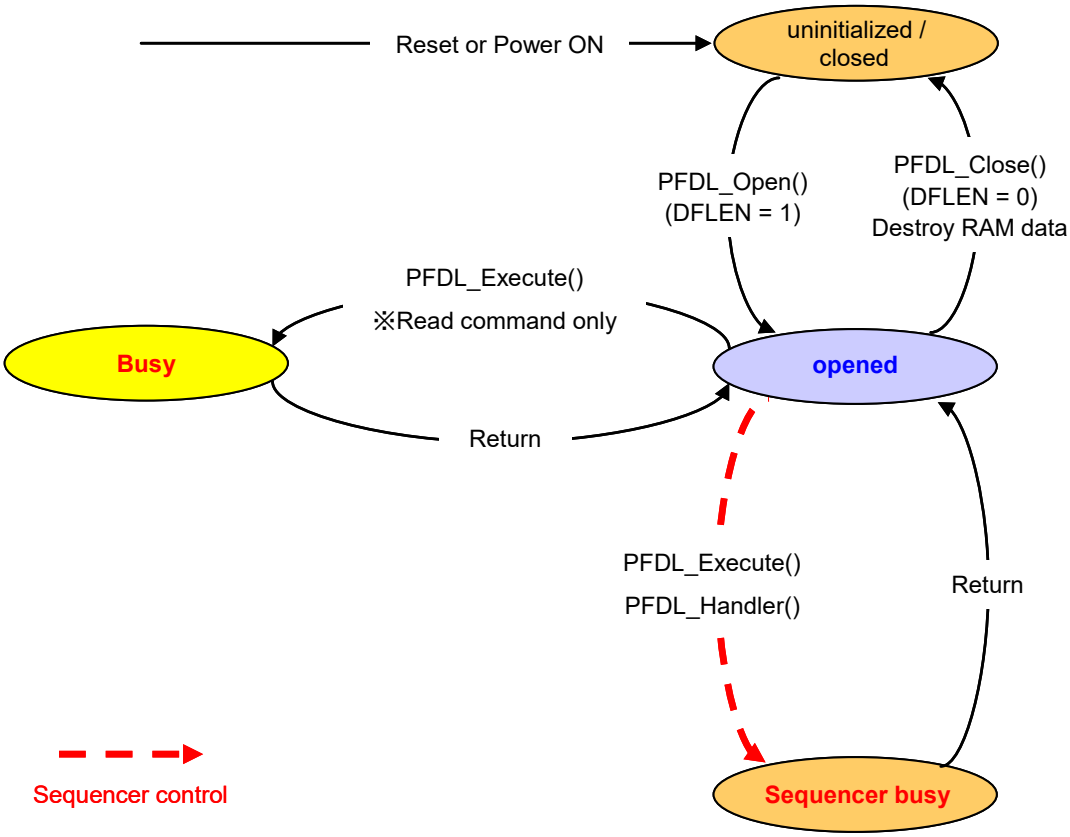


図 1－1 データ・フラッシュ・ライブラリ Type04 の状態遷移図

【状態遷移図の概要】

データ・フラッシュ・ライブラリ Type04を使用してデータ・フラッシュ・メモリを操作するためには、用意されている関数を順に実行し、処理を進める必要があります。関数の詳細については、**第3章 データ・フラッシュ・ライブラリ関数**の項を参照してください。

(1) uninitialized / closed

RL78マイクロコントローラのPower ON、またはReset解除時の状態です。フラッシュ・セルフ・プログラミング・ライブラリやEEPROMエミュレーション・ライブラリ、Type04以外のデータ・フラッシュ・ライブラリ、STOP命令、HALT命令を実行する場合は、opened状態からPFDL_Closeを実行し、この状態に遷移させてください。

(2) opened

uninitialized / closed状態からPFDL_Open関数を実行し、データ・フラッシュ・ライブラリ Type04を初期化することで、データ・フラッシュ・メモリへのアクセスが可能になった状態です。PFDL_Closeを実行し、uninitialized / closed状態に遷移するまでの間は、フラッシュ・セルフ・プログラミング・ライブラリやEEPROMエミュレーション・ライブラリ、Type04以外のデータ・フラッシュ・ライブラリ、STOP命令、HALT命令は実行できません。

また、PFDL_Open関数を実行すると、データ・フラッシュ・コントロール・レジスタ（DFLCTL）をデータ・フラッシュ・メモリへのアクセス許可状態（DFLEN = 1）に設定し、PFDL_Close関数を実行すると、データ・フラッシュ・コントロール・レジスタをデータ・フラッシュ・メモリへのアクセス禁止状態（DFLEN = 0）に設定します。

(3) busy

指定された処理を実行している状態です。処理が終了するまでユーザプログラムには戻りません。

(4) sequencer busy

シーケンサを使用して指定された処理を実行している状態です。PFDL_Execute関数でデータ・フラッシュ・メモリへの制御内容を指定し、PFDL_Handler関数でステータス・チェックを行います。実行した関数はシーケンサの動作終了を待たずにユーザプログラムに戻ります。また、シーケンサの使用中はデータ・フラッシュ・メモリを参照できません。

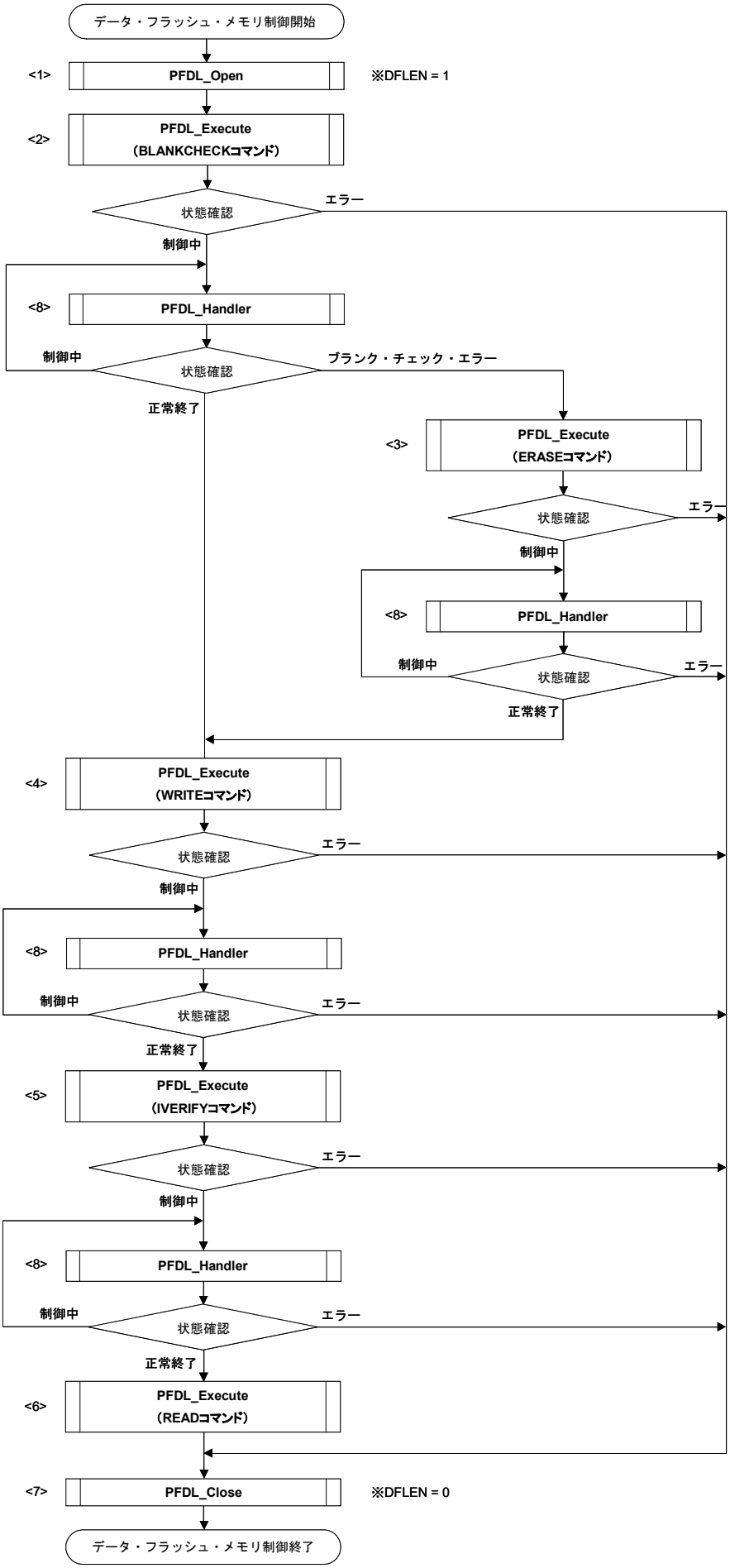


図 1―2 データ・フラッシュ・ライブラリ Type04 の操作フロー例

<1>　PFDL_Open：データ・フラッシュ・ライブラリ Type04で使用するRAMの初期化と開始

PFDL_Open関数を呼び出し、データ・フラッシュ・ライブラリ Type04で使用するRAMを初期化し、データ・フラッシュ・ライブラリ Type04を使用可能にします。

また、データ・フラッシュ・コントロール・レジスタ（DFLCTL）をデータ・フラッシュ・メモリへのアクセス許可状態（DFLEN = 1）に設定します。

<2>　PFDL_Execute：指定アドレスに対する1-1024バイトのブランク・チェック

PFDL_Execute関数(PFDL_CMD_BLANKCHECK_BYTESコマンドを指定)を呼び出し、指定アドレスに対して1-1024バイトのブランク・チェック（書き込み可能な領域であることの確認）を行います。ブロックを跨って処理を実行することはできません。

<3>　PFDL_Execute：指定ブロック（1 Kバイト）の消去

PFDL_Execute関数(PFDL_CMD_ERASE_BLOCKコマンドを指定)を呼び出し、指定ブロック（1 Kバイト）の消去を行います。

<4>　PFDL_Execute：指定アドレスに対する1-1024バイトのデータ書き込み

PFDL_Execute関数(PFDL_CMD_WRITE_BYTESコマンドを指定)を呼び出し、指定アドレスに対する1-1024バイトの書き込みを行います。ブロックを跨って処理を実行することはできません。また、書き込みはブランク状態か、もしくは消去済みの領域へのみ行えます。既に書き込みを行っている領域(0xFFを書いた領域も含みます)へ再度書き込みを行う事(上書き)はできません。

<5>　PFDL_Execute：指定アドレスに対する1-1024バイトの内部ベリファイ[注]

PFDL_Execute関数(PFDL_CMD_IVERIFY_BYTESコマンドを指定)を呼び出し、指定アドレスに対する1-1024バイトの内部ベリファイ[注]を行います。ブロックを跨って処理を実行することはできません。

注. 内部ベリファイとは、フラッシュ・メモリのセルの信号レベルが適正であるかを確認することです。
データの比較による確認は行いません。

<6>　PFDL_Execute：指定アドレスに対する1-1024バイトの読み込み

PFDL_Execute関数(PFDL_CMD_READ_BYTESコマンドを指定)を呼び出し、指定アドレスに対する1-1024バイトの読み込みを行います。読み込み処理はPFDL_Execute関数内で全ての処理を実行します。ブロックを跨って処理を実行することはできません。

<7>　PFDL_Close：データ・フラッシュ・ライブラリ Type04の終了

PFDL_Close関数を呼び出し、データ・フラッシュ・ライブラリ Type04を終了します。

また、データ・フラッシュ・コントロール・レジスタ（DFLCTL）をデータ・フラッシュ・メモリへのアクセスを禁止状態(DFLEN = 0)に設定します。PFDL_Close関数は、データ・フラッシュ・メモリへの制御を終了させる必要がある場合に実行します。

<8>　PFDL_Handler：ステータス・チェック

PFDL_Handler関数を呼び出し、ステータス・チェックを行います。シーケンサによるデータ・フラッシュ・メモリへの制御が終了するまで、ステータス・チェックを行う必要があります。

第2章　プログラミング環境

この章では、ユーザがデータ・フラッシュ・ライブラリ Type04を使用してデータ・フラッシュ・メモリの書き換えを行ううえで、必要なハードウェア環境とソフトウェア環境について説明します。

2.1　ハードウェア環境

RL78マイクロコントローラのデータ・フラッシュ・ライブラリ Type04はシーケンサを使用し、データ・フラッシュ・メモリの書き換え制御を実行します。

データ・フラッシュ・メモリへの制御をシーケンサが行ってくれるので、データ・フラッシュ・メモリ制御中にユーザプログラムを動作させる事が可能です。この事をBGO（バック・グラウンド・オペレーション）と言います。

データ・フラッシュ・メモリの制御中はデータ・フラッシュ・メモリを参照できませんが、コード・フラッシュ・メモリの参照は可能なため、割り込み処理[注]やユーザプログラム、およびデータ・フラッシュ・ライブラリ Type04は通常どおりROM上に配置して使用することが可能です。

図2－1にデータ・フラッシュ・メモリ制御中の状態、図2－2にデータ・フラッシュ・メモリへの制御を行う場合のデータ・フラッシュ・ライブラリ関数の実行例を示します。

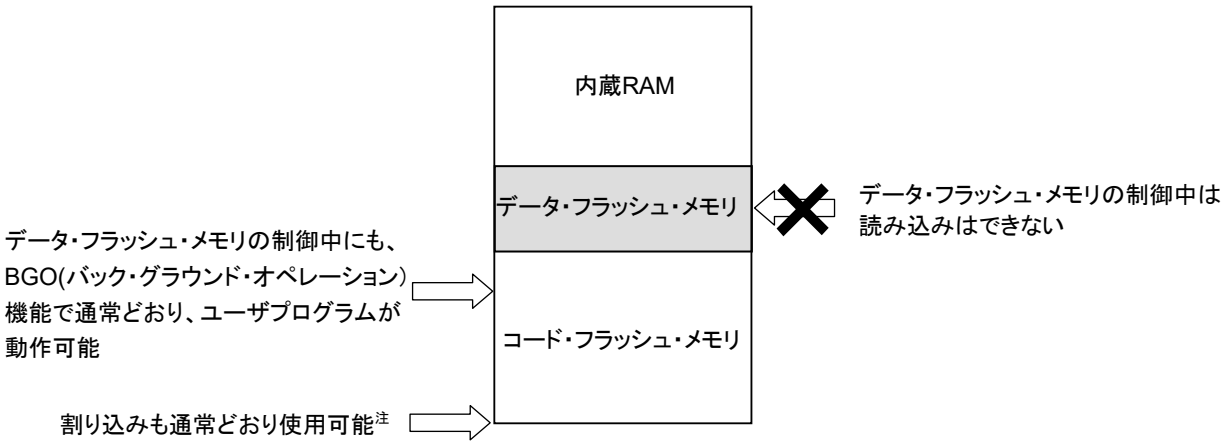


図 2－1 データ・フラッシュ・メモリ制御中の状態

注 R5F10266製品はデータ・フラッシュ・ライブラリ Type04使用中、割り込みは禁止です。

・RL78マイクロコントローラのシーケンサに該当処理の実行要求を行ったのち、制御を直ちにユーザプログラムに戻します。データ・フラッシュ・メモリへの制御の結果については、ユーザプログラムからステータス・チェック関数（PFDL_Handler関数）を呼び出し、データ・フラッシュ・メモリの制御状態を確認する処理が必要です。

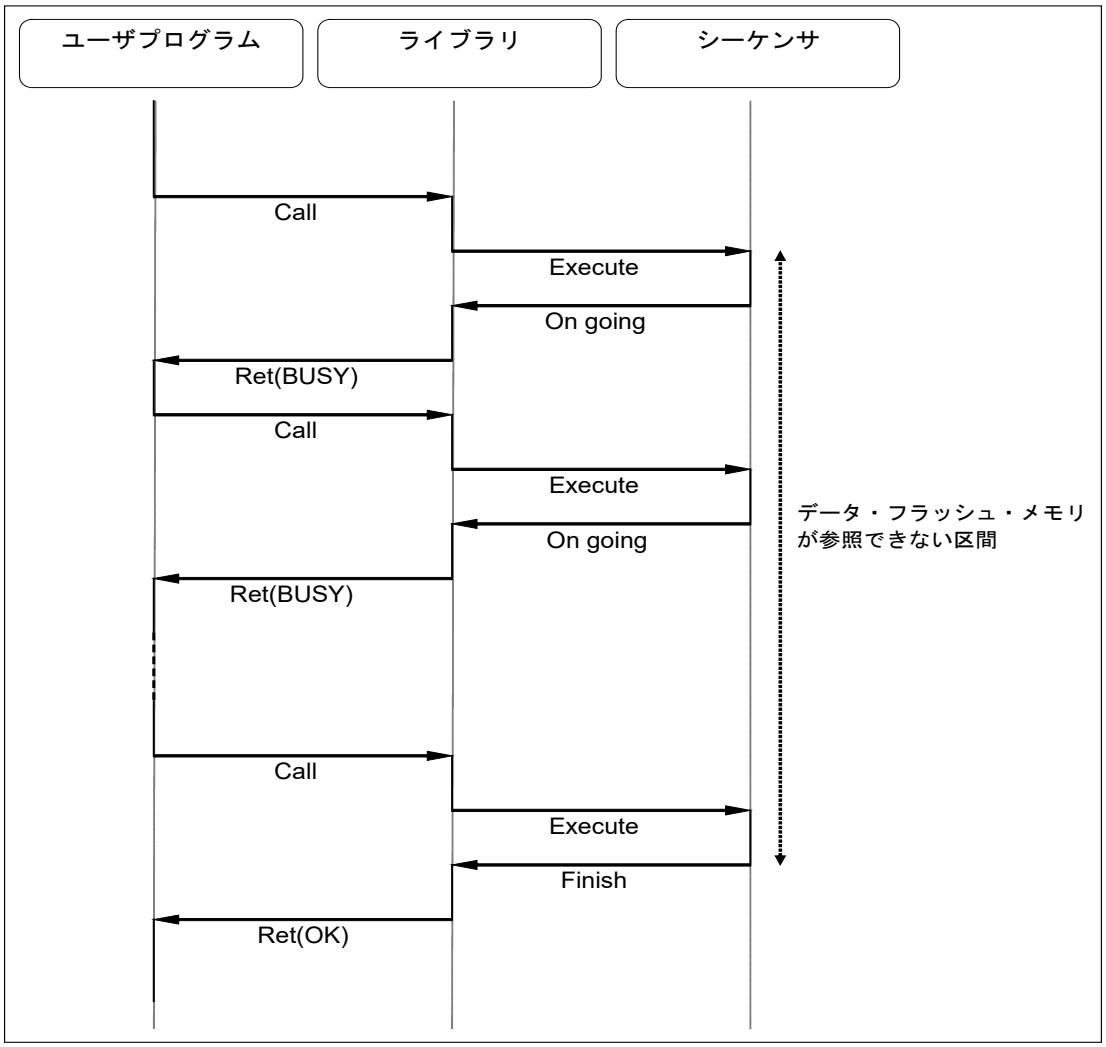


図 2−2 データ・フラッシュ・メモリの制御例

2. 1. 1　初期設定

データ・フラッシュ・ライブラリ Type04を使用してデータ・フラッシュ・メモリの制御を行う場合、以下の設定を行う必要があります。

（1） 高速オンチップ・オシレータの起動

データ・フラッシュ・ライブラリ Type04の使用中は高速オンチップ・オシレータを動作させておく必要があります。高速オンチップ・オシレータを停止させている場合は、使用前に起動させてください。

（2） CPUの動作周波数[注1]の設定

データ・フラッシュ・ライブラリ Type04内部のタイミング計算を行うため、初期化時にCPUの動作周波数を設定する必要があります。CPUの動作周波数の設定方法には関しては、PFDL_Open関数の説明を参照してください。

（3）フラッシュ書き換えモード[注2]の設定

書き込み時のフラッシュ書き換えモードの設定を行うため、データ・フラッシュ・ライブラリ Type04の初期化時に、以下のフラッシュ書き換えモードを設定する必要があります。フラッシュ書き換えモードの設定方法に関しては、PFDL_Open関数の説明を参照してください。

- フルスピード・モード
- ワイド・ボルテージ・モード

注 1.　CPUの動作周波数はデータ・フラッシュ・ライブラリ Type04内部のタイミング計算用のパラメータとして使用されます。本設定によってCPUの動作周波数が変わることはありません。また、高速オンチップ・オシレータの動作周波数ではありません。

2.　フラッシュ書き換えモードの詳細については、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。

2. 1. 2　データ・フラッシュ・コントロール・レジスタ（DFLCTL）

データ・フラッシュ・メモリへのアクセス許可／禁止を設定するレジスタです。PFDL_Open関数を実行すると、データ・フラッシュ・コントロール・レジスタ（DFLCTL）をデータ・フラッシュ・メモリへのアクセス許可状態（DFLEN = 1）に設定し、PFDL_Close関数を実行すると、データ・フラッシュ・コントロール・レジスタをデータ・フラッシュ・メモリへのアクセス禁止状態（DFLEN = 0）に設定します。

アドレス：F0090H　　リセット時：00H　　R/W

略号	7	6	5	4	3	2	1	0
DFLCTL	0	0	0	0	0	0	0	DFLEN

DFLEN	データ・フラッシュ・メモリのアクセス制御
0	データ・フラッシュ・メモリのアクセス禁止
1	データ・フラッシュ・メモリのアクセス許可

図 2−3 データ・フラッシュ・コントロール・レジスタ（DFLCTL）のフォーマット

注意　　データ・フラッシュ・コントロール・レジスタ（DFLCTL）の設定に関する内容については、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。

2.1.3　ブロック

RL78マイクロコントローラは、フラッシュ・メモリが1 Kバイト単位でブロック分割されています。データ・フラッシュ・ライブラリでは、このブロックを単位としてデータ・フラッシュ・メモリに対し、消去処理を行います。読み込みや書き込み、ブランク・チェック、内部ベリファイは開始アドレス[注]と実行サイズを指定して実行します。

図2－4にデータ・フラッシュ・メモリのブロック位置とブロック番号の例を示します。

注　データ・フラッシュ・ライブラリ Type04 を使用し、データ・フラッシュ・メモリに書き込みや読み込みを行う場合に使用するアドレス値は、データ・フラッシュ・メモリのブロック0 を開始アドレス（アドレス0）とした相対アドレスです。絶対アドレスではありませんので、ご注意ください。

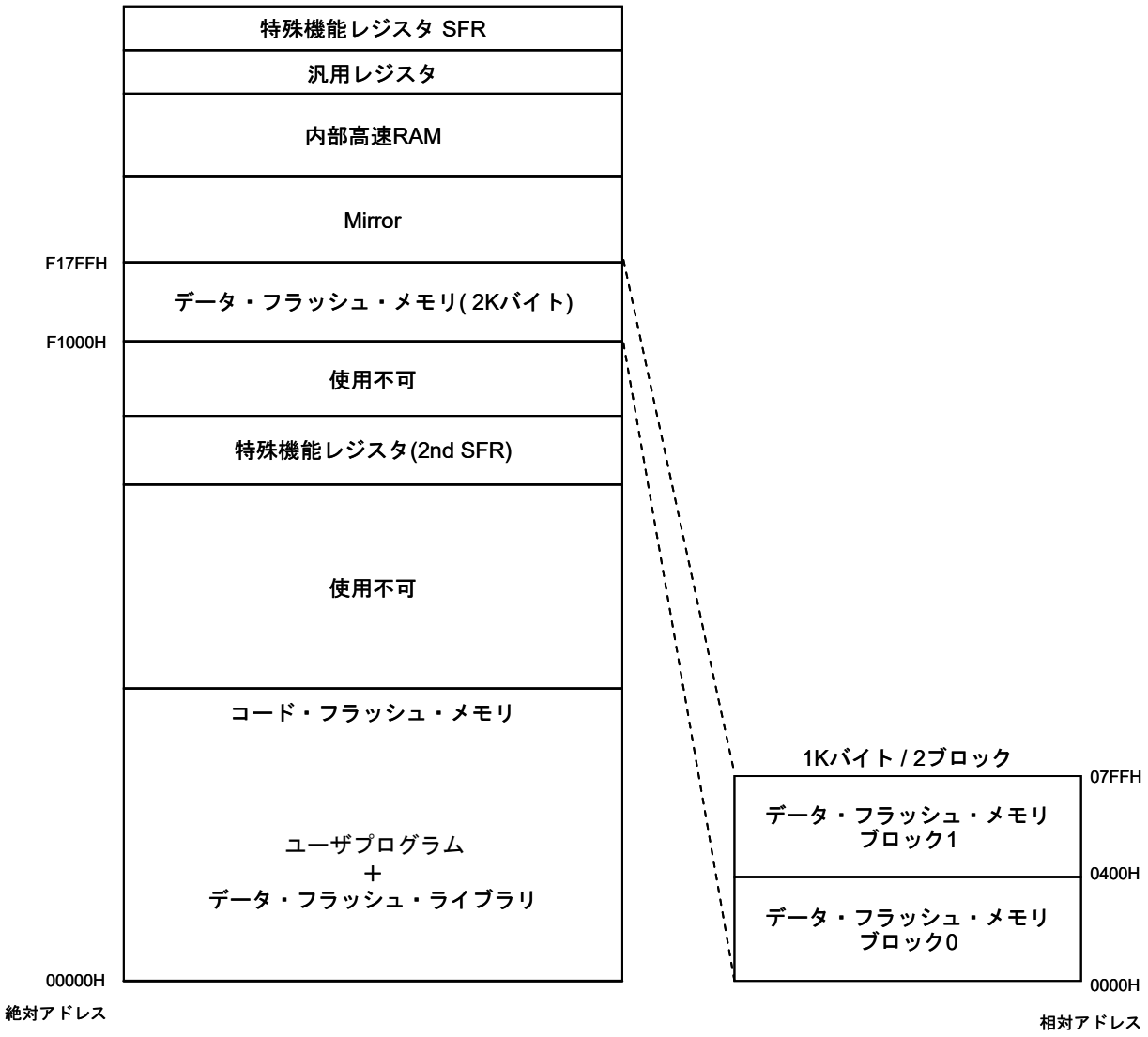


図 2－4 データ・フラッシュ・メモリのブロック（RL78/G12:データ・フラッシュ・メモリが 2 K バイトの場合）

2. 1. 4　処理時間

この節ではデータ・フラッシュ・ライブラリ Type04 の処理時間(Read コマンドを除く)について記載します。データ・フラッシュ・ライブラリ関数は内部 ROM 領域（フラッシュ・メモリ）に配置した場合と内部 RAM 領域に配置した場合では、実行クロックが異なります。処理時間は ROM から実行した場合に対し、RAM から実行した場合、最大 2 倍程度になる場合があります。

ここで記載している関数の処理時間は、データ・フラッシュ・ライブラリ関数を ROM で実行した場合の処理時間です。データ・フラッシュ・ライブラリ関数のセグメントについては、3.2 章 データ・フラッシュ・ライブラリ関数のセグメントをご参照ください。

（1）関数処理時間

データ・フラッシュ・ライブラリ Type04 の関数がユーザプログラムから実行された後、処理を終了してユーザプログラムに戻るまでの時間です。関数処理時間の概念を図 2−5、 処理時間を表 2−1〜2−2 に示します。

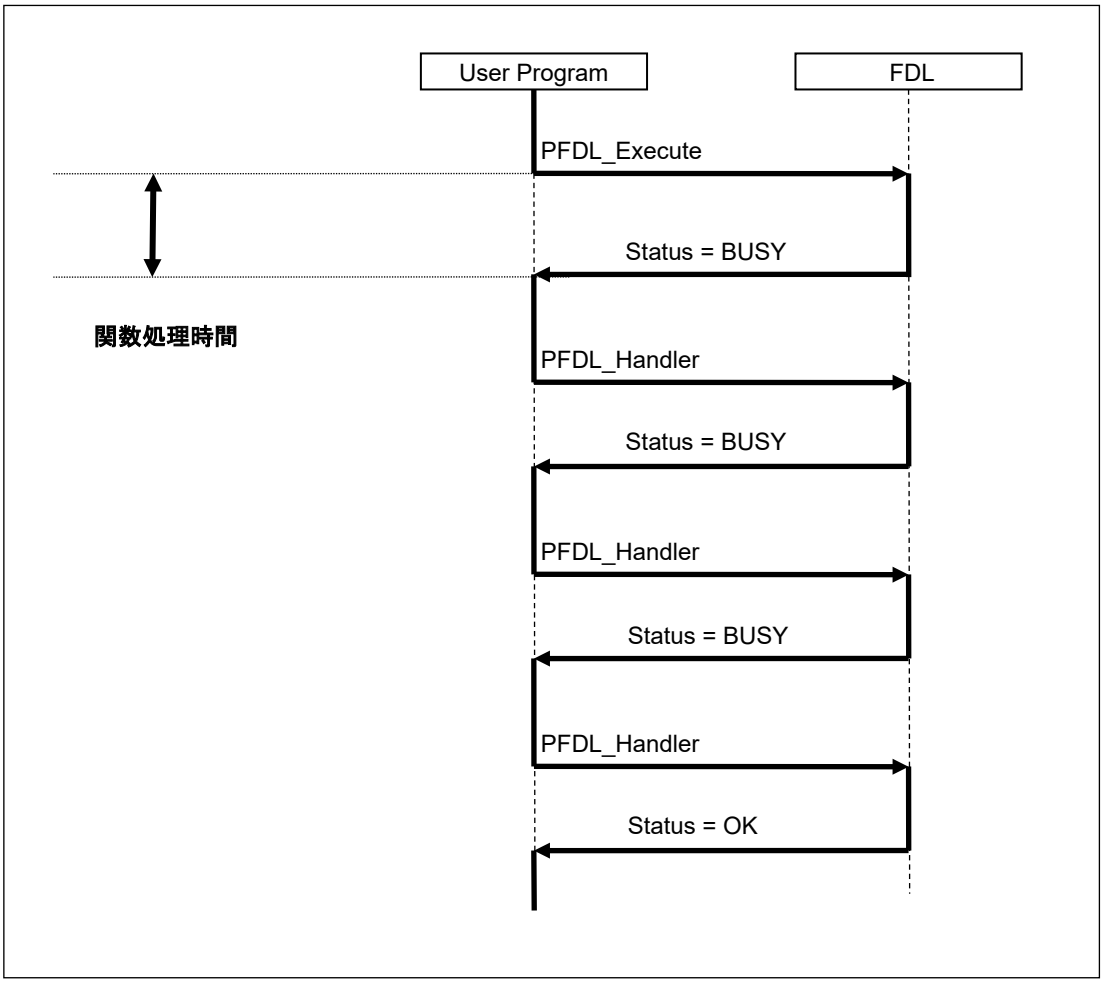


図 2−5 データ・フラッシュ・ライブラリ関数処理時間の概念図

表 2−1 フルスピード・モードのデータ・フラッシュ・ライブラリ関数処理時間

PFDL_Functions	Max (μs)
PFDL_Open	862 / fCLK
PFDL_Execute(Erase)	536 / fCLK
PFDL_Execute(BlankCheck)	484 / fCLK
PFDL_Execute(Write)	549 / fCLK
PFDL_Execute(IVerify)	502 / fCLK
PFDL_Execute(Read)	53 / fCLK + 17 / fCLK × Bytes
PFDL_Handler	251 / fCLK + 14
PFDL_Close	823 / fCLK + 443
PFDL_GetVersionString	10 / fCLK

備考1.　fCLK：CPUの動作周波数(例：20MHz時のfCLK = 20)

2.　Bytes：1バイト単位の書き込みデータ長（ 例：8バイト指定時Bytes = 8 ）

表 2−2 ワイド・ボルテージ・モードのデータ・フラッシュ・ライブラリ関数処理時間

PFDL_Functions	Max (μs)
PFDL_Open	862 / fCLK
PFDL_Execute(Erase)	536 / fCLK
PFDL_Execute(BlankCheck)	484 / fCLK
PFDL_Execute(Write)	549 / fCLK
PFDL_Execute(IVerify)	502 / fCLK
PFDL_Execute(Read)	53 / fCLK + 17 / fCLK × Bytes
PFDL_Handler	251 / fCLK + 14
PFDL_Close	779 / fCLK + 968
PFDL_GetVersionString	10 / fCLK

備考1.　fCLK：CPUの動作周波数(例：20MHz時のfCLK = 20)

2.　Bytes：1バイト単位の書き込みデータ長（ 例：8バイト指定時Bytes = 8 ）

（2）PFDL_Handler(ステータス・チェック)推奨実行間隔

Read 処理を除き、データ・フラッシュ・ライブラリ Type04 では、PFDL_Handler 関数でステータス・チェックを行いますが、シーケンサの制御が終了する前に PFDL_Handler 関数を実行しても処理の終了確認はできないため、PFDL_Execute 関数でシーケンサの制御を伴う処理を実行する場合は、一定の間隔を空けることによってステータス・チェック処理の効率が向上します。また、Write コマンドによる書き込みは、1byte 毎にステータス・チェック処理によるトリガーが必要なため、1byte 単位でのステータス・チェック処理が必要です。推奨実行間隔の概念を図 2−6～2−7、 処理時間を表 2−3～2−4 に示します。

- データ・フラッシュ・ライブラリ Type04 で 3byte の書き込みを行う場合、シーケンサは 1byte 毎に書き込みを行うため、1byte の書き込み終了時に PFDL_Handler 関数によって次の書き込みへのトリガーが必要です。1byte 以降の書き込み処理が残っている状態で PFDL_Handler 関数を実行しない場合、次の書き込み処理に遷移できないため、書き込み処理が終了しません。

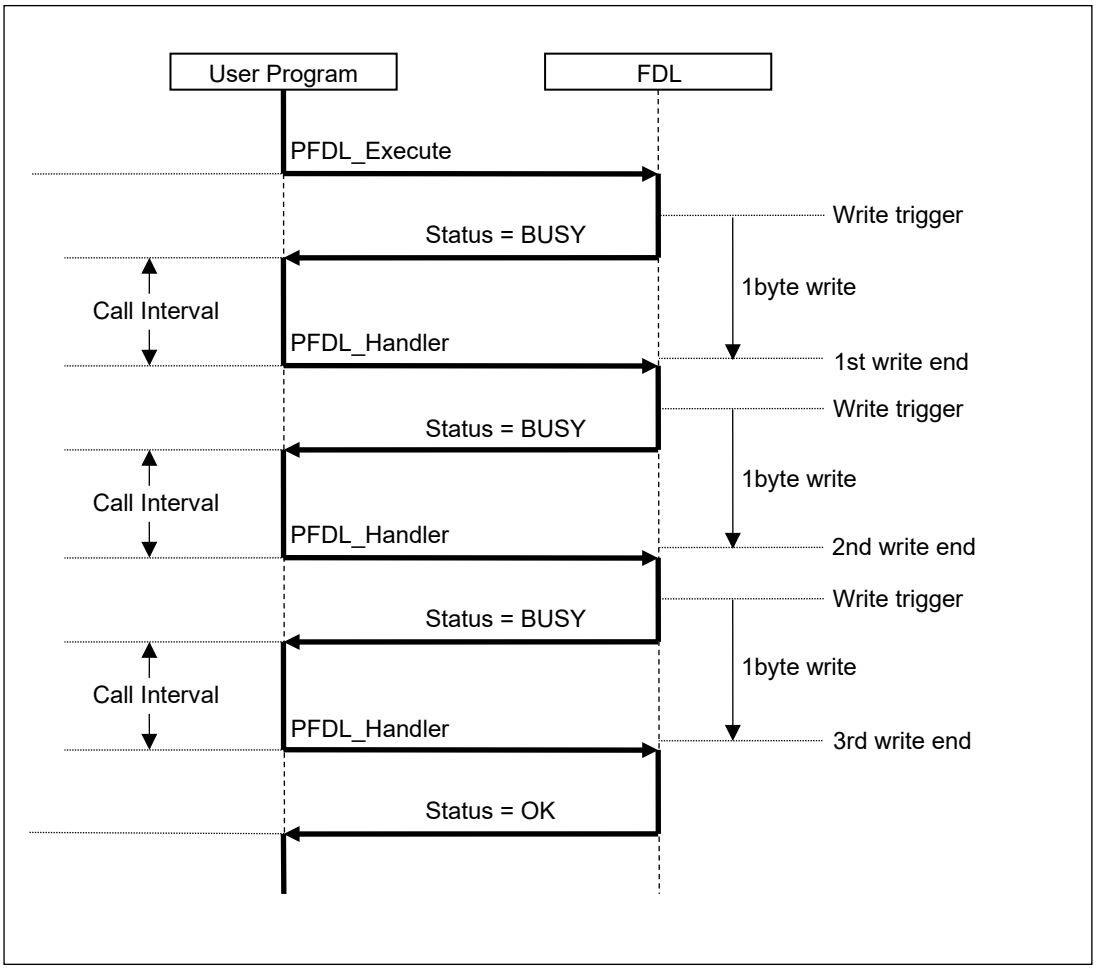


図 2−6 Write コマンドのステータス・チェック推奨実行間隔の概念図(3byte 書き込みの場合)

・　データ・フラッシュ・ライブラリ Type04 で Write 以外の処理を行う場合、シーケンサは全ての処理を終了するまで Busy 状態となり、PFDL_Handler 関数によるトリガーは必要ありません。

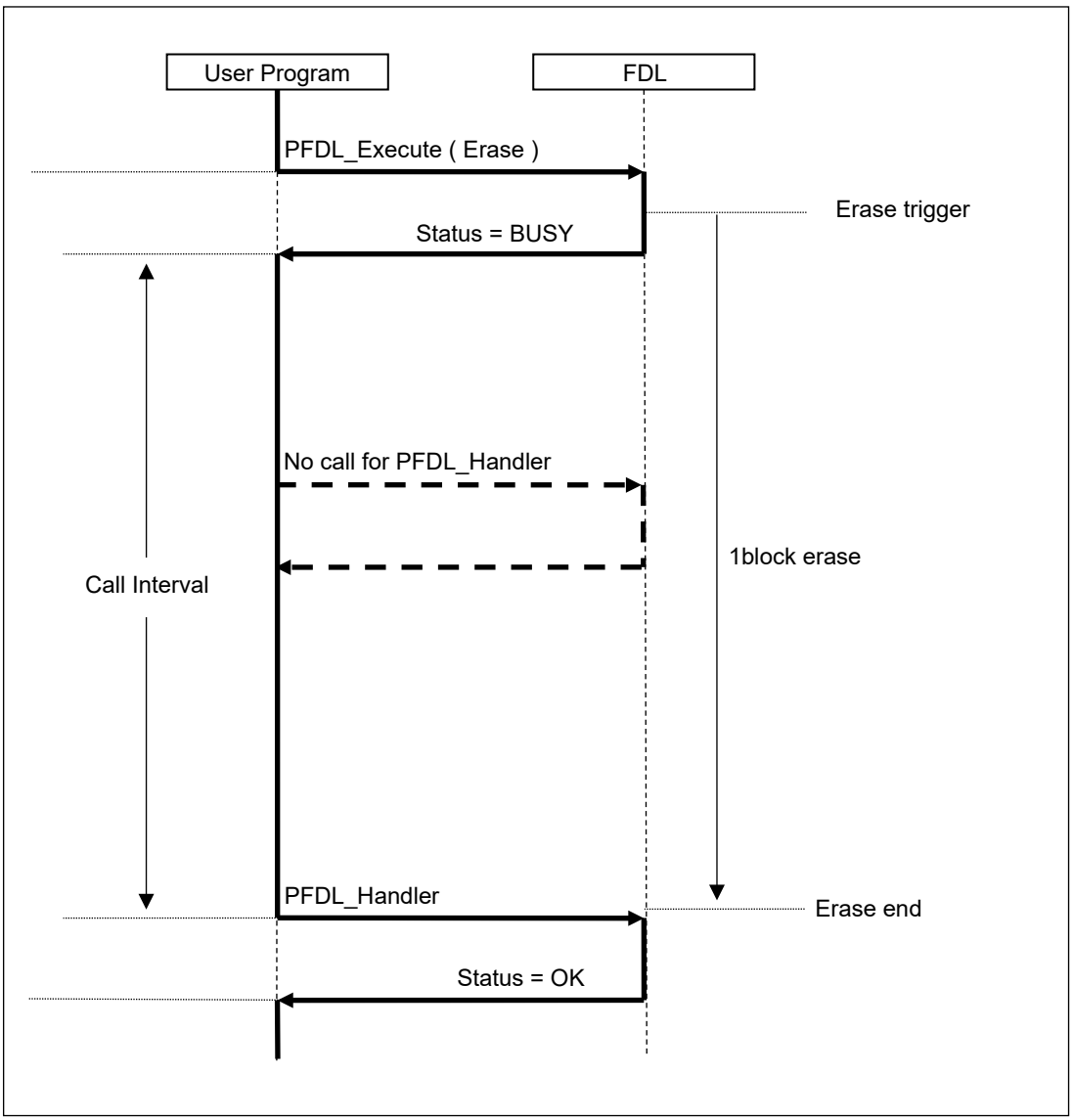


図 2－7 Write コマンド以外のステータス・チェック推奨実行間隔の概念図(消去の場合)

表 2−3 フルスピード・モードのデータ・フラッシュ・ライブラリ Type04 推奨実行間隔

PFDL_Functions		Max (μs)
PFDL_Execute(Erase)	case:block is Blank	5860 / f_{CLK} + 335
	case:block is not Blank	12734 / f_{CLK} + 6946
PFDL_Execute(BlankCheck)		33 / f_{CLK} + 18 + (6 / f_{CLK} + 0.31) × Bytes
PFDL_Execute(Write)		61 / f_{CLK} + 47
PFDL_Execute(IVerify)		13 / f_{CLK} + 17 + (28 / f_{CLK} + 4) × Bytes

備考1.　f_{CLK}：CPUの動作周波数(例：20MHz時のf_{CLK} = 20)

2.　1バイト単位の書き込みデータ長（ 例：8バイト指定時のBytes = 8 ）

表 2−4 ワイド・ボルテージ・モードのデータ・フラッシュ・ライブラリ Type04 推奨実行間隔

PFDL_Functions		Max (μs)
PFDL_Execute(Erase)	case:block is Blank	5065 / f_{CLK} + 1167
	case:block is not Blank	11144 / f_{CLK} + 8620
PFDL_Execute(BlankCheck)		30 / f_{CLK} + 57 + (5 / f_{CLK} + 1.084) × Bytes
PFDL_Execute(Write)		57 / f_{CLK} + 99
PFDL_Execute(IVerify)		14 / f_{CLK} + 44 + (17 / f_{CLK} + 29) × Bytes

備考1.　f_{CLK}：CPUの動作周波数(例：20MHz時のf_{CLK} = 20)

2.　Bytes：1バイト単位の書き込みデータ長（ 例：8バイト指定時のBytes = 8 ）

（3）総合処理時間

PFDL_Execute 関数での各コマンド実行時の総合処理時間は正常終了する場合の時間です。ユーザが PFDL_Handler 関数を実行するまでの時間(Call Interval)によるオーバーヘッドやエラーなどにより異常終了する場合の処理時間は含んでいません。

PFDL_Execute 関数での各コマンド実行時の総合処理時間の概念を図 2－8、 処理時間を表 2－5～2－6 に示します。

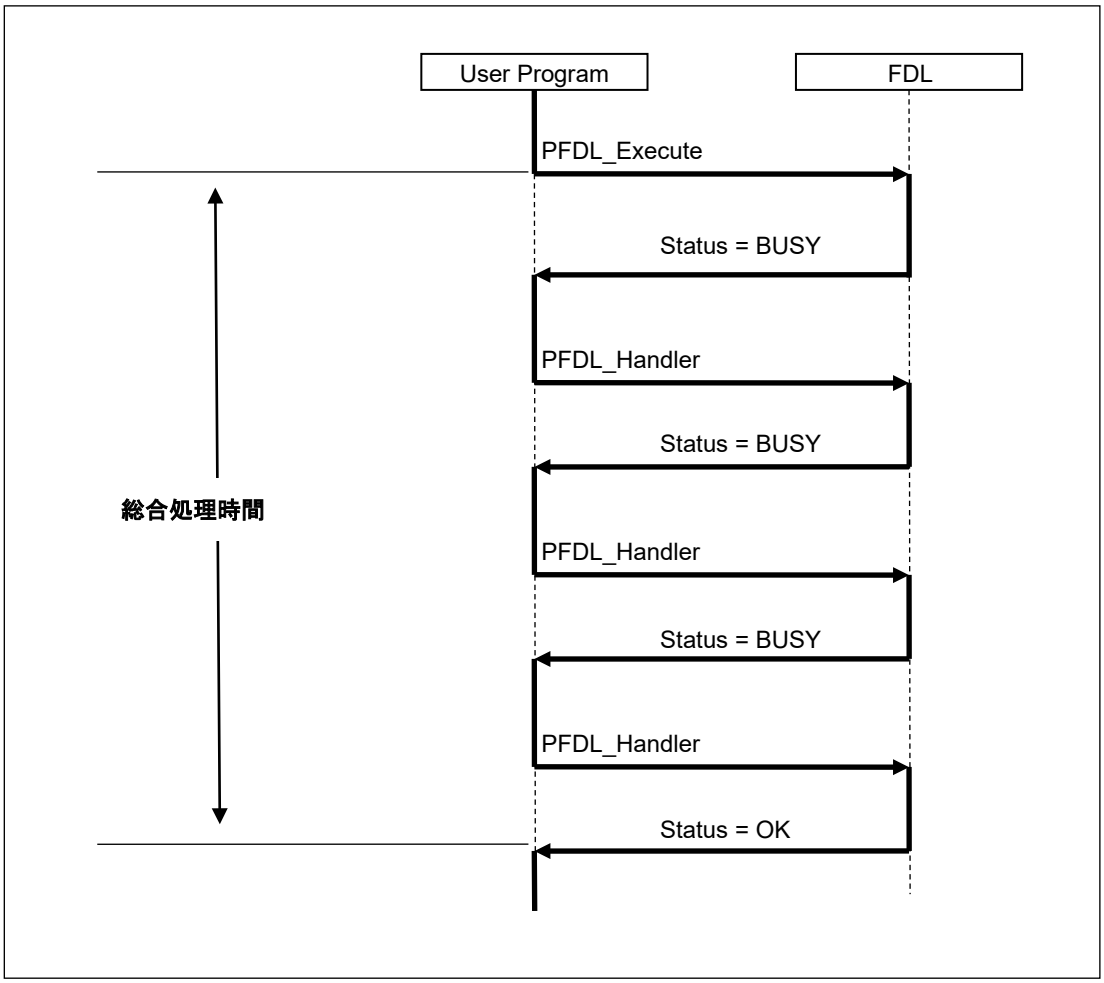


図 2－8 総合処理時間の概念図

表 2−5 フルスピード・モードのデータ・フラッシュ・ライブラリ Type04 総合処理時間

PFDL_Functions	Typical (μs)	Max (μs)
PFDL_Execute(Erase)	11250 / fCLK + 5800	281561 / fCLK + 264790
PFDL_Execute(BlankCheck)	906 / fCLK + 30 +(5 / fCLK +0.26) × Bytes	1088 / fCLK + 36 + (6 / fCLK + 0.31) × Bytes
PFDL_Execute(Write)	487 / fCLK + 11.67 + (212 / fCLK + 39.17) × Bytes	585 / fCLK + 14 + (714 / fCLK + 430) × Bytes
PFDL_Execute(IVerify)	621 / fCLK + 25 + (23 / fCLK + 3.33) × Bytes	746 / fCLK + 30 + (28 / fCLK + 4) × Bytes

備考1.　fCLK：CPUの動作周波数(例：20MHz時のfCLK = 20)

2.　Bytes：1バイト単位の書き込みデータ長（ 例：8バイト指定時のBytes = 8 ）

表 2−6 ワイド・ボルテージ・モードのデータ・フラッシュ・ライブラリ Type04 総合処理時間

PFDL_Functions	Typical (μs)	Max (μs)
PFDL_Execute(Erase)	9925 / fCLK + 7194.17	249000 / fCLK + 299307
PFDL_Execute(BlankCheck)	903 / fCLK + 62.5 +(4 / fCLK +0.9) × Bytes	1084 / fCLK + 75 + (5 / fCLK + 1.084) × Bytes
PFDL_Execute(Write)	487 / fCLK + 11.67 + (208 / fCLK + 82.5) × Bytes	585 / fCLK + 14 + (669 / fCLK + 954) × Bytes
PFDL_Execute(IVerify)	622 / fCLK + 48.33 + (14 / fCLK + 24.17) × Bytes	747 / fCLK + 58 + (17 / fCLK + 29) × Bytes

備考1.　fCLK：CPUの動作周波数(例：20MHz時のfCLK = 20)

2.　Bytes：1バイト単位の書き込みデータ長（ 例：8バイト指定時のBytes = 8 ）

2. 2　ソフトウェア環境

データ・フラッシュ・ライブラリ Type04では、該当プログラムをユーザ領域に配置するため、使用するライブラリの容量のプログラム領域を消費し、データ・フラッシュ・ライブラリ Type04自身は、CPU、スタック、データ・バッファを使用します。

★ また、データ・フラッシュ・ライブラリ Type04 は、CA78K0R コンパイラ用、CC-RL コンパイラ用とLLVMコンパイラ用があります。各表中では、CA78K0R コンパイラ用を「CA78」、CC-RL コンパイラ用を「CCRL」、LLVM コンパイラ用を「LLVM」と略す場合があります。

表2－7に、必要となるソフトウェア・リソースの一覧[注1、2]、図2－9、2－10にRAMの配置イメージ例を示します。

★ **表 2－7 データ・フラッシュ・ライブラリ Type04 ソフトウェア・リソース**

項目	容量(バイト)		データ・フラッシュ・ライブラリ Type04の使用領域[注1、2]
	CA78	CCRL LLVM	
セルフRAM[注3]	0 ～ 136[注3]	0 ～ 136[注3]	RL78ファミリ データ・フラッシュ・ライブラリ Type04 Ver.1.05で使用するセルフRAM領域は デバイス毎に異なります。詳細については、『RL78ファミリ セルフプログラミングライブラリ セルフRAMリスト (R20UT2943) 』を参照してください。
スタック	MAX 46[注4]	MAX 40[注4]	セルフRAM、FFE20H-FFEFFH以外のRAM領域に配置可能[注2]
データ・バッファ[注5]	1 ～ 1024	1 ～ 1024	
ライブラリ関数の引数	0 ～ 8	0 ～ 8	
ライブラリ・サイズ	ROM: MAX 177	ROM: MAX 168	セルフRAM、FFE20H-FFEFFH以外のプログラム領域に配置可能

注1. 『RL78 ファミリ セルフプログラミングライブラリ セルフ RAM リスト(R20UT2943)』に掲載の無い製品については、お問い合わせください。

2. R5F10266 製品については、本表の対象外です。別途「2.2.1 R5F10266 製品のソフトウェア・リソース」の項を参照してください。

3. データ・フラッシュ・ライブラリ Type04 でワーク・エリアとして使用する領域を本書、及びリリース・ノートではセルフ RAM と呼びます。セルフ RAM はマッピングされず、データ・フラッシュ・ライブラリ実行時に自動的に使用される（以前のデータは破壊される）領域のため、ユーザ設定等は必要ありません。データ・フラッシュ・ライブラリを使用していない状態の場合は、通常の RAM 空間として使用できます。

★ 4. ライブラリで使用するスタック（CA78K0R コンパイラ用では 42 バイト、CC-RL コンパイラ用では 36 バイト、LLVM コンパイラ用では 36 バイト）に、関数コールに必要なスタック量（最大 4 バイト）を合計した値です。

5. データ・バッファは、読み書きを行うデータを入力するために必要な RAM 領域です。必要なサイズは、読み書きを行う単位によって変わります。1 バイトの読み書きを行う場合、必要なデータ・バッファのサイズは 1 バイトです。

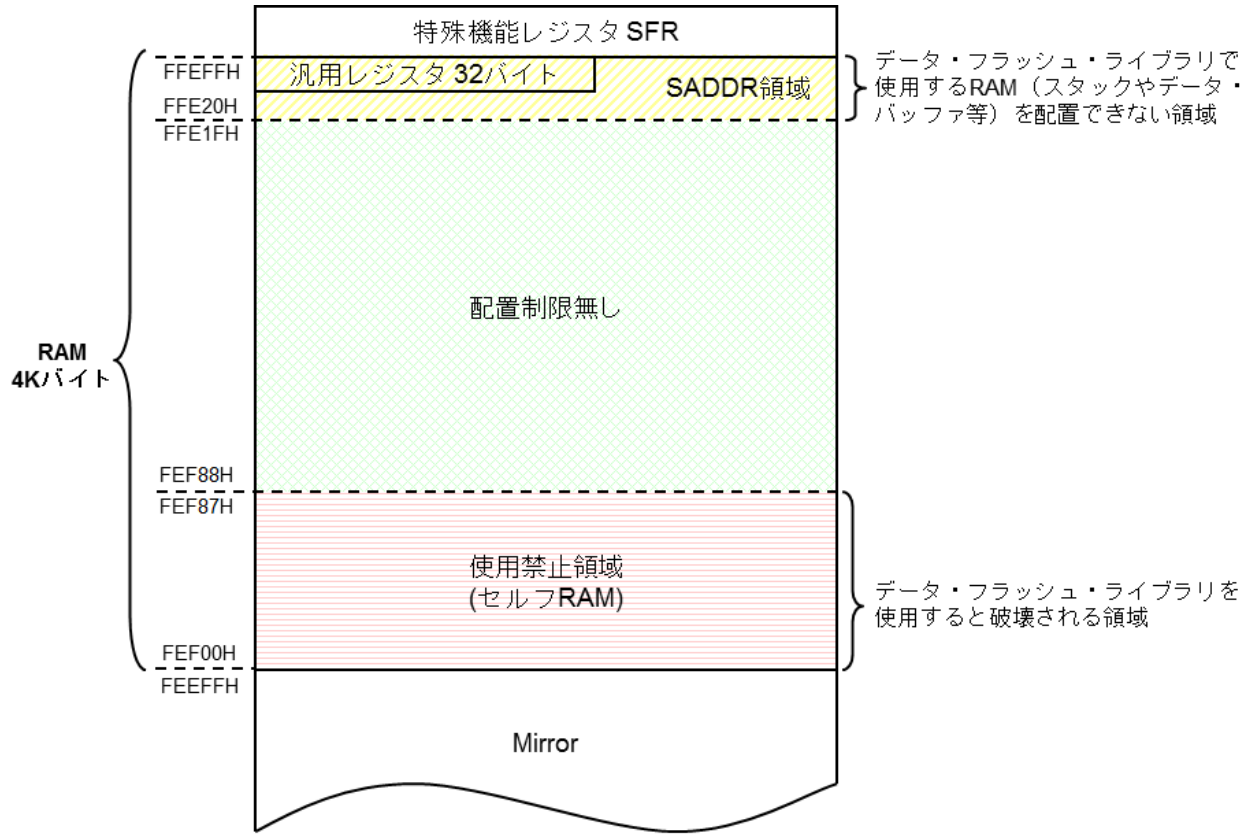


図 2－9 RAM の配置イメージ例 1 /セルフ RAM 有り（RL78/G13：RAM 4KB/ROM 64KB 製品）

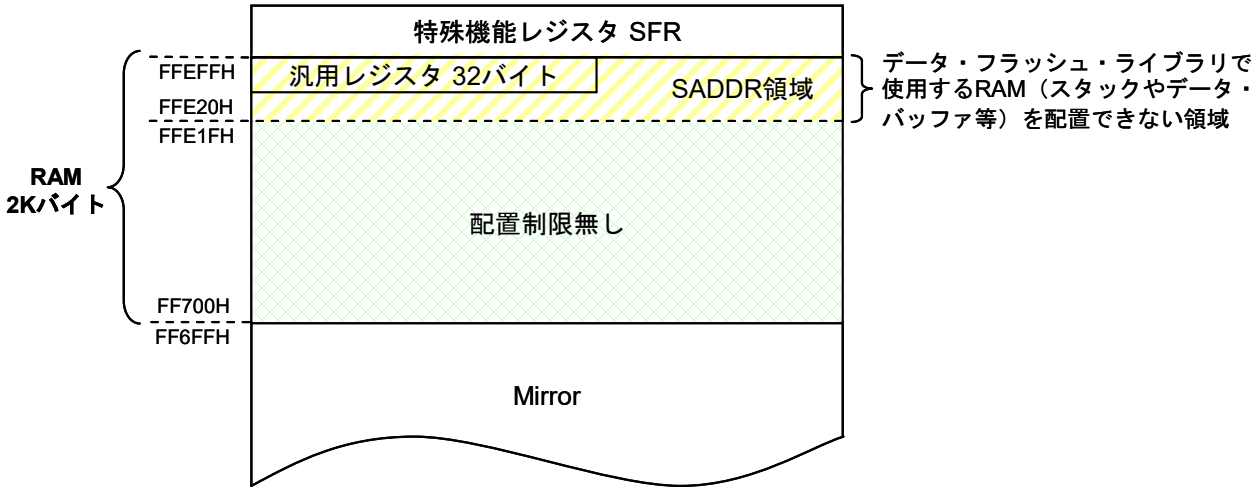


図 2－10 RAM の配置イメージ例 2 /セルフ RAM 無し（RL78/G13：RAM 2KB/ROM 32KB 製品）

2. 2. 1　R5F10266製品のソフトウェア・リソースについて

R5F10266製品については、使用禁止領域や、スタックの配置方法に制限があり、ソフトウェア・リソースの設定が他製品と異なります。また、データ・フラッシュ・ライブラリ Type04使用中、割り込みは禁止です。

表2－8に、R5F10266製品で必要なソフトウェア・リソースの一覧、図2－11にRAMの領域イメージ例、図2－12、2－13に使用言語毎のスタックの配置イメージを示します。

★ **表 2－8 R5F10266 製品のデータ・フラッシュ・ライブラリ Type04 ソフトウェア・リソース**

項目	容量(バイト)		R5F10266製品のデータ・フラッシュ・ライブラリ Type04の使用領域[注2、3]
	CA78	CCRL LLVM	
セルフRAM	0	0	セルフRAMはありません。
スタック	MAX 46[注1]	MAX 40[注1]	FFEA2H-FFEDFHのRAM領域に配置[注2]
データ・バッファ[注3]	1 ～ 24	1 ～ 24	セルフRAM、FFE20H-FFEFFH以外のRAM領域に配置可能
ライブラリ関数の引数	8	8	
ライブラリ・サイズ	ROM: MAX 177	ROM: MAX 168	コード・フラッシュ・メモリ領域に配置してください。

★ 注 1.　ライブラリで使用するスタック（CA78K0R コンパイラ用では 42 バイト、CC-RL コンパイラ用では 36 バイト、LLVM コンパイラ用では 36 バイト）に、関数コールに必要なスタック量（最大 4 バイト）を合計した値です。

2.　ライブラリで使用するスタックが、必ずこの範囲に収まるように指定してください。

3.　データ・バッファは、読み書きを行うデータを入力するために必要な RAM 領域です。必要なサイズは、読み書きを行う単位によって変わります。1 バイトの読み書きを行う場合、必要なデータ・バッファのサイズは 1 バイトです。

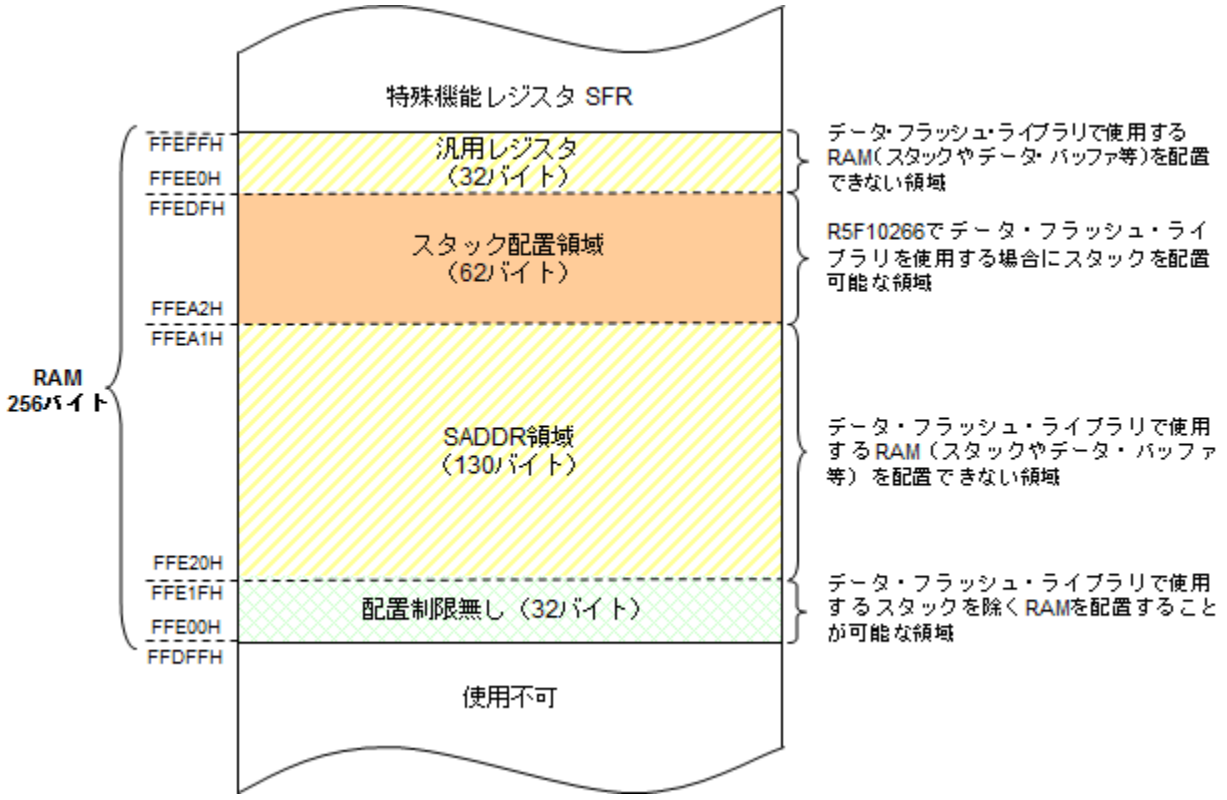


図 2－11 R5F10266 製品でデータ・フラッシュを使用する場合の RAM の領域イメージ

★

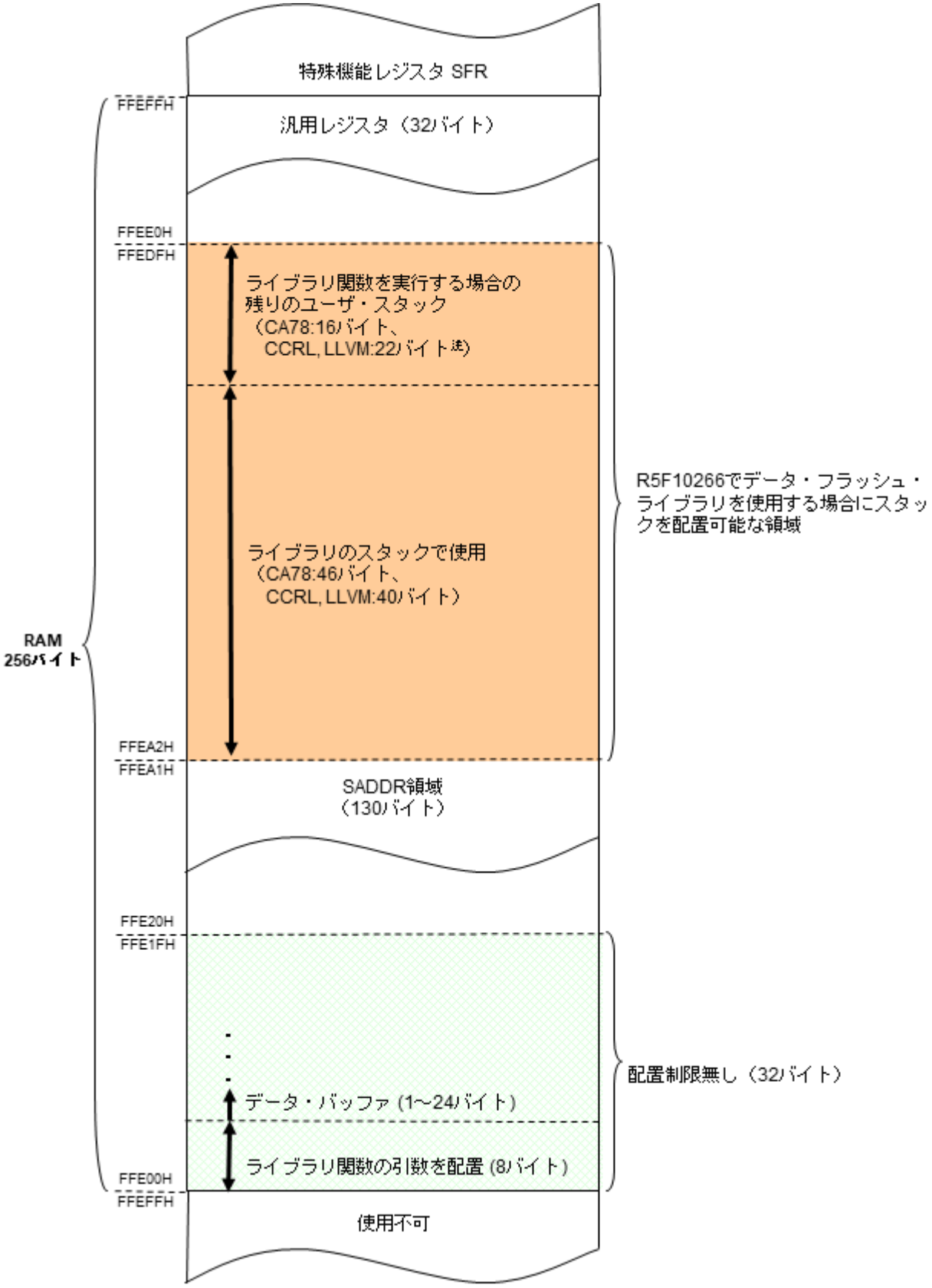


図 2－12 スタックの配置イメージ例 1（R5F10266 製品 / アセンブリ言語の場合）

★　注意　オンチップ・デバッグ機能を使用する場合は、オンチップ・デバッグ機能で使用するスタック容量（4バイト）が必要になり、同スタックを本領域に配置する必要があります。
このスタックを配置する場合、データ・フラッシュ・ライブラリ Type04を使用中にユーザ・スタックとして使用できる容量は、CA78では残り12バイト、CCRL、およびLLVMでは残り18バイトです。

★

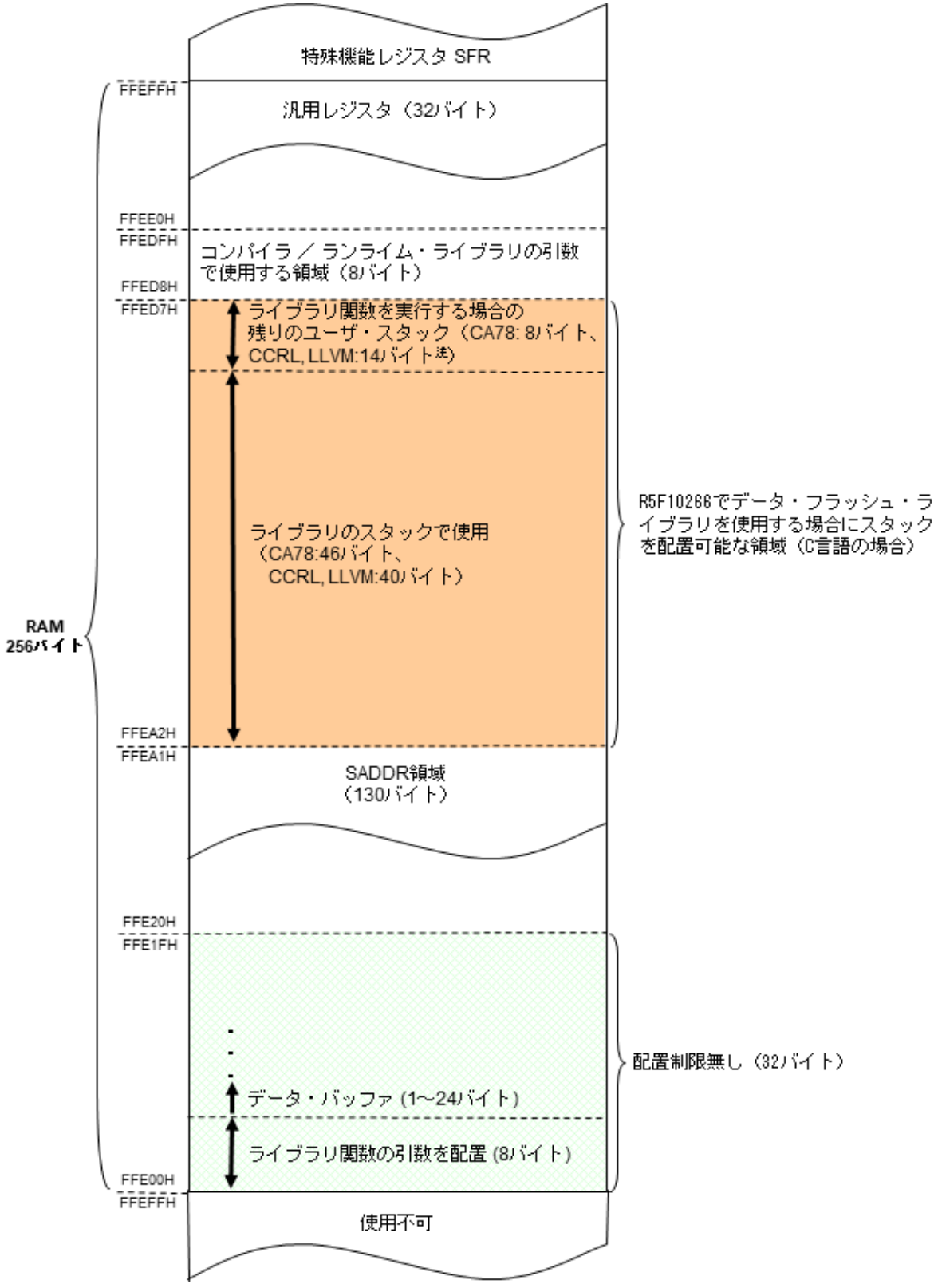


図 2－13 スタックの配置イメージ例 2（R5F10266 製品 / C 言語の場合）

★　注意　C言語で使用する場合は、スタートアップルーチンからmain関数を実行する際に使用するスタック容量（4バイト）、オンチップ・デバッグ機能を使用する場合は、オンチップ・デバッグ機能で使用するスタック容量（4バイト）が必要になり、同スタックを本領域に配置する必要が有ります。
これらのスタックを両方使用する場合、データ・フラッシュ・ライブラリ Type04を使用中にユーザ・スタックとして使用できる容量は、CA78では残り0バイト、CCRL、およびLLVMでは残り6バイトです。

2. 2. 2　セルフRAM

データ・フラッシュ・ライブラリ Type04は、ワーク・エリアとしてRAM領域を使用する場合があります。この領域をセルフRAMと呼びます。このセルフRAMで使用するデータは、ライブラリ内で定義され、ユーザがデータを定義する必要はありません。データ・フラッシュ・ライブラリ関数を呼び出すとセルフRAM領域のデータが書き換わります。また、データ・フラッシュ・プログラミングで使用するセルフRAMの領域はマイコンによって異なり、ユーザRAMを使用する場合があります。

★ この場合、ユーザがユーザRAM上にセルフRAM領域の確保をする必要がありますので、リンク時にセルフRAM領域の確保(CA78K0Rコンパイラではリンクディレクティブファイルで設定可能、CC-RLコンパイラではセクションを配置しないことで設定可能、LLVMコンパイラではリンカスクリプトファイルで設定可能)を行ってください。リンクディレクティブファイルでの設定方法については、リリース・ノートの『内蔵RAM領域の定義』の章をご参照ください。

2. 2. 3　レジスタ・バンク

データ・フラッシュ・ライブラリ Type04はユーザが選択しているレジスタ・バンクの汎用レジスタ、 ES/CS レジスタ、 SP およびPSW を使用します。

2. 2. 4　スタック、データ・バッファ

データ・フラッシュ・ライブラリ Type04は、シーケンサを使用してデータ・フラッシュ・メモリへの書き込みを行いますが、事前の設定や制御を行うためにCPUを使用します。このため、データ・フラッシュ・ライブラリ Type04を使用するためには、ユーザプログラムで指定されているスタックも必要です。

★ 備考　ユーザが指定するアドレスに、スタック、データ・バッファを配置するためには、CA78K0Rコンパイラではリンク・ディレクティブを使用します。CC-RLコンパイラでは、セクションの配置をリンカオプションで設定します。LLVMコンパイラではリンカスクリプトファイルを使用します。

・スタック

ユーザプログラムで使用するスタックに加え、データ・フラッシュ・ライブラリ関数で必要なスタックの容量を事前に確保し、データ・フラッシュ・ライブラリ Type04を実行する場合におけるスタック処理で、ユーザ使用RAMが破壊されないように配置する必要があります。スタックの指定可能範囲は、セルフRAM領域とFFE20H-FFEFFHの領域を除く、内蔵RAM領域です。

・データ・バッファ

データ・バッファの用途は以下のとおりです。

- 書き込み実行時、書き込むデータを配置する領域
- 読み込み実行時、取得する読み込みデータを配置する領域

なお、データ・バッファの先頭アドレスの指定可能範囲は、スタックと同様、セルフRAM領域とFFE20H-FFEFFHの領域を除く、内蔵RAM領域です。

2. 2. 5　データ・フラッシュ・ライブラリ

すべてのデータ・フラッシュ・ライブラリ関数がリンクされるのではなく、使用するデータ・フラッシュ・ライブラリ関数に限定してリンク[注]します。

・データ・フラッシュ・ライブラリ Type04のメモリ配置について

データ・フラッシュ・ライブラリ Type04では使用する関数や変数にセグメントが割り当てられており、データ・フラッシュ・ライブラリ Type04で使用される領域を特定の場所に指定することができます。

注　アセンブリ言語の場合は、インクルード・ファイルから使用しない関数を削除することで、使用するデータ・フラッシュ・ライブラリ関数に限定してリンクできます。

2. 2. 6　プログラム領域

データ・フラッシュ・ライブラリ Type04、およびデータ・フラッシュ・ライブラリ Type04を使用するユーザプログラムを配置する領域です。

RL78マイクロコントローラのデータ・フラッシュ・ライブラリ Type04では、シーケンサを使用してデータ・フラッシュ・メモリの制御を行うため、データ・フラッシュ・メモリの制御中にユーザプログラムを動作させる事が可能です。（バック・グラウンド・オペレーション）

2. 3　プログラミング環境に関する注意事項

（1）　データ・フラッシュ・ライブラリ Type04実行中にフラッシュ・セルフ・プログラミング・ライブラリ、EEPROMエミュレーション・ライブラリ、またはType04以外のデータ・フラッシュ・ライブラリを実行しないで下さい。フラッシュ・セルフ・プログラミング・ライブラリ、EEPROMエミュレーション・ライブラリ、またはType04以外のデータ・フラッシュ・ライブラリを使用する場合は、必ずPFDL_Close関数まで実行し、データ・フラッシュ・ライブラリを終了状態にする必要があります。

（2）　データ・フラッシュ・ライブラリ Type04実行中にSTOP命令、およびHALT命令は実行しないで下さい。STOP命令、およびHALT命令を実行する必要がある場合は、PFDL_Close関数まで実行し、データ・フラッシュ・ライブラリを終了させてください。

（3）　ウォッチドック・タイマはデータ・フラッシュ・ライブラリ Type04実行中も停止しません。

（4）　データ・フラッシュ・ライブラリ Type04によるデータ・フラッシュ・メモリ操作中はデータ・フラッシュ・メモリを読み出せません。

（5）　データ・フラッシュ・ライブラリ関数で使用する引数（データ・バッファなど）やスタックを0xFFE20（0xFE20）以上のアドレスに配置しないでください。

（6）　データ・フラッシュ・ライブラリ Type04実行中に データ・トランスファ・コントローラ（DTC）を使用する場合は、DTCで使用するRAM領域をセルフRAM、および 0xFFE20(0xFE20)以上のアドレスに配置しないでください。

（7）　データ・フラッシュ・ライブラリ Type04を終了させるまで、データ・フラッシュ・ライブラリが使用する RAM 領域(セルフRAMを含む)を使用しないでください。

（8）　データ・フラッシュ・ライブラリ Type04はデータ・フラッシュ・ライブラリ関数の多重実行に対応していないため、データ・フラッシュ・ライブラリ関数を割り込み処理内で実行しないで下さい。

（9）　データ・フラッシュ・ライブラリ Type04はデータ・フラッシュ・ライブラリ関数の多重実行に対応していないため、OS上でデータ・フラッシュ・ライブラリ Type04を実行する場合は、複数のタスクからデータ・フラッシュ・ライブラリ関数を実行しないで下さい。

（10）　データ・フラッシュ・ライブラリ Type04を開始する前に高速オンチップ・オシレータを起動しておく必要があります（RL78 マイクロコントローラ内のハードウェアがフラッシュ書き換えの際に使用します）。

（11） CPUの動作周波数と初期化関数(PFDL_Open)で設定するCPUの動作周波数設定値について、以下の点に注意してください。

- CPUの動作周波数を 4MHz未満[注1]で使用する場合は、1MHz、 2MHz、 3MHz を使用することができます。（1.5MHz のように整数値でない周波数は使用できません）初期化関数で設定するCPUの動作周波数も 1、2、3 の整数値を設定してください。
- CPUの動作周波数を 4MHz以上[注1]で使用する場合は、小数点以下の数値を持つ周波数を使用することができます。ただし、初期化関数(PFDL_Open)で設定するCPUの動作周波数は、小数点以下を切り上げた整数値を設定してください。
  （例：4.5MHz の場合は、初期化関数で 5 を設定してください）
- 初期化関数(PFDL_Open)で設定するCPUの動作周波数設定値は、ユーザプログラムの実行に使用する動作周波数です。高速オンチップ・オシレータの動作周波数ではありません。（ユーザプログラムの実行に使用するクロックが高速オンチップ・オシレータの場合は、実行する高速オンチップ・オシレータの動作周波数を入力してください。）

注1　CPUの動作周波数の範囲については対象の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。

（12） データ・フラッシュ・ライブラリ Type04の実行中にDFLCTL（データ・フラッシュ・コントロール・レジスタ）を操作しないでください。また、データ・フラッシュ・ライブラリ Type04を終了する場合、PFDL_Close関数で、DFLCTLをデータ・フラッシュへのアクセス禁止状態に設定します。
データ・フラッシュ・ライブラリ Type04終了後もデータ・フラッシュ・メモリへのアクセスが必要な場合は、PFDL_Close関数実行後に、DFLCTLをデータ・フラッシュ・メモリのアクセス許可に設定し、セットアップ処理[注2]を行ってください。

注2　セットアップ方法については対象の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。

（13） データ・フラッシュ・ライブラリ関数で使用する引数(RAM)は一度初期化してください。初期化をしない場合、RAMパリティ・エラーが検出され、RL78マイクロコントローラにリセットが発生する可能性があります。
RAMパリティ・エラーについては、対象の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。

（14） データ・フラッシュ・メモリへの書き込みはブランク状態か、もしくは消去済みの領域へのみ行えます。既に書き込みを行った領域に対し、消去を行わずに再度書き込み（上書き）を行う事はできません。消去を行わずに再度書き込みを行った場合、データ・フラッシュ・メモリへダメージを与える可能性があります。

（15） データ・フラッシュ・ライブラリ Type04ではデータ・フラッシュ・ライブラリ関数の引数として設定したパラメータのエラーチェックを行っていません。従いまして、設定するパラメータは、対象のRL78マイクロコントローラの仕様を確認した上で、必ず正しい値を設定してください。正しい値を設定するためにパラメータ・チェックを行う必要がある場合は、ユーザプログラムにてパラメータ・チェックを行ってください。

（16） R5F10266製品はデータ・フラッシュ・ライブラリ Type04使用中、割り込みは禁止です。

（17） CC-RLコンパイラ用データ・フラッシュ・ライブラリのセグメント(PFDL_COD)は、64KB境界を跨いで配置することができません。
必ず、64KB 境界を跨がないように配置してください。

（18） ルネサス製コンパイラCC-RL のアセンブラを使用する場合、16 進数のPrefix 表現(0x..)とSuffix 表現(..H)は混在できません。ユーザの環境に合わせてpfdl.inc 内のシンボル定義を編集することで表現方法を指定してください。

pfdl.inc

```
; __PFDL_INC_BASE_NUMBER_SUFFIX .SET 1
```

シンボル"__PFDL_INC_BASE_NUMBER_SUFFIX"を定義しない場合(初期状態)、Prefix 表現が選択されます。

pfdl.inc

```
__PFDL_INC_BASE_NUMBER_SUFFIX .SET 1
```

シンボル"__PFDL_INC_BASE_NUMBER_SUFFIX"を定義する場合、Suffix 表現が選択されます。

第3章　データ・フラッシュ・ライブラリ関数

この章では、データ・フラッシュ・ライブラリ関数の詳細について説明します。

3.1　データ・フラッシュ・ライブラリ関数の種類

データ・フラッシュ・ライブラリ Type04は、次に示す関数で構成されています。

表 3−1 データ・フラッシュ・ライブラリ関数一覧

関数名	説　　明
PFDL_Open	データ・フラッシュ・ライブラリ Type04で使用するRAMの初期化と開始
PFDL_Close	データ・フラッシュ・ライブラリ Type04の終了
PFDL_Execute	データ・フラッシュ・メモリへの制御実行
PFDL_Handler	データ・フラッシュ・メモリへの制御状態の確認と継続実行の設定 （ステータス・チェック処理）
PFDL_GetVersionString	データ・フラッシュ・ライブラリ Type04のバージョン情報を取得

★　3.2　データ・フラッシュ・ライブラリ関数のセグメント（セクション）

データ・フラッシュ・ライブラリ関数の実体は、指定の領域に配置する必要があります。

この実体は、CA78K0Rコンパイラでメモリ配置する際にセグメントとして使用され、CC-RLコンパイラ、LLVMコンパイラでは、セクションとして使用されます。

セグメント（セクション）は以下のように構成されています。

・PFDL_COD　　：データ・フラッシュ・ライブラリ関数のセグメント（セクション）です。
ROM, RAMどちらにも配置可能です。

以降、CC-RLコンパイラ用、LLVMコンパイラ用データ・フラッシュ・ライブラリを使用する場合は、"セグメント"を "セクション"と読み替えてください。

3. 3　コマンド

データ・フラッシュ・ライブラリ Type04のデータ・フラッシュ・メモリへの操作内容はPFDL_Execute関数の引数に設定するコマンドで指定します。以下にPFDL_Execute関数で指定するコマンド一覧を示します。実行方法の詳細については、PFDL_Execute関数の項を参照してください。

表 3−2 PFDL_Execute 関数で指定するコマンド一覧

定　義	値	コマンド名
PFDL_CMD_READ_BYTES	0x00	読み込みコマンド
PFDL_CMD_BLANKCHECK_BYTES	0x08	ブランク・チェック・コマンド
PFDL_CMD_ERASE_BLOCK	0x03	消去コマンド
PFDL_CMD_WRITE_BYTES	0x04	書き込みコマンド
PFDL_CMD_IVERIFY_BYTES	0x06	内部ベリファイ・コマンド

3. 4　BGO（バック・グラウンド・オペレーション）

データ・フラッシュ・ライブラリ関数はシーケンサを使用しない関数と、シーケンサを使用する関数[注]に分かれます。また、シーケンサを使用する関数[注]はBGOが可能です。

以下にBGO時のデータ・フラッシュ・ライブラリ Type04の動作例と、関数のシーケンサ制御の有無の一覧を示します。

注　PFDL_CMD_READ_BYTESコマンド実行時は除く。

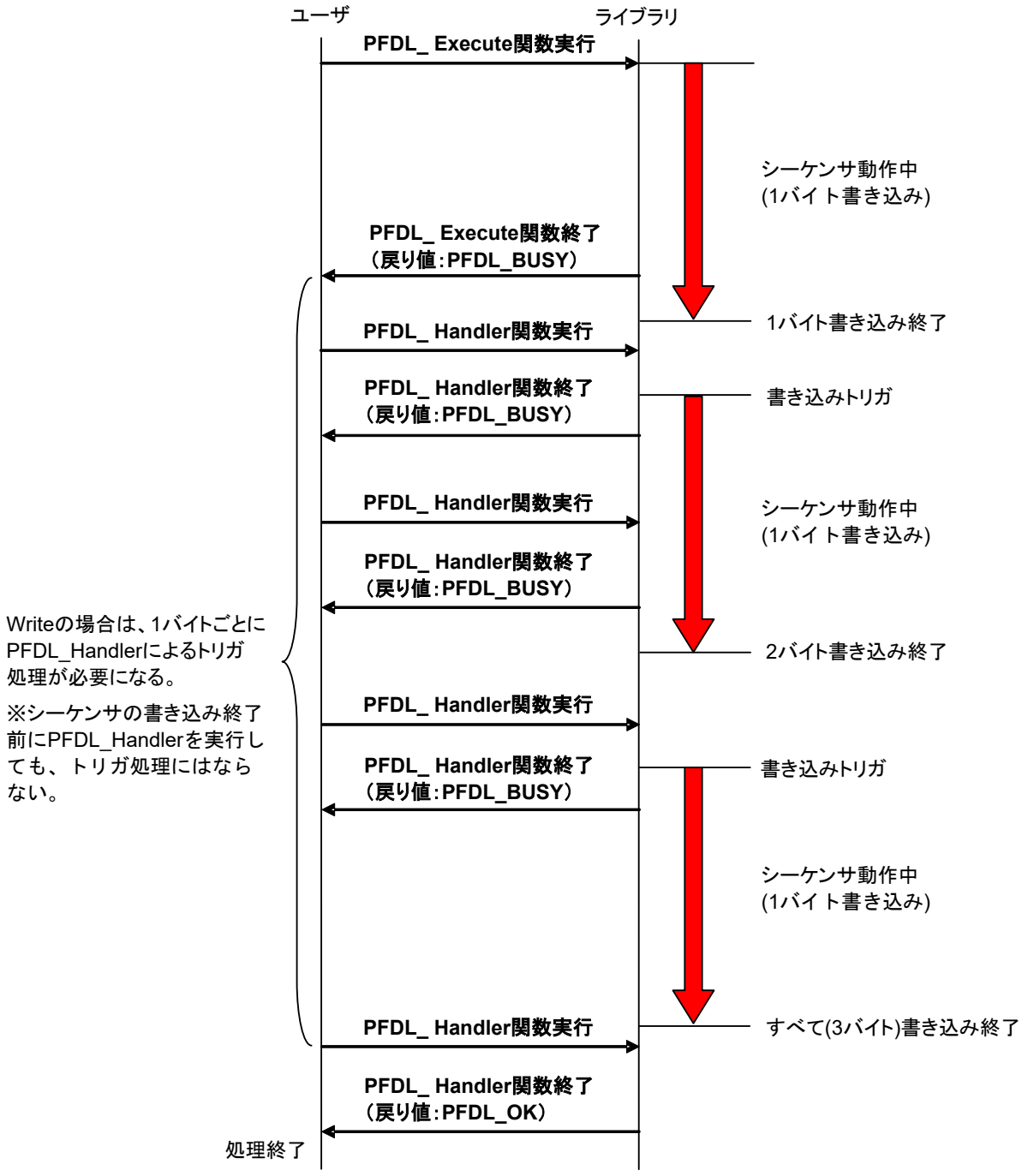


図 3−1 BGO 時の動作例 1（Write:3 バイト書き込み時の例）

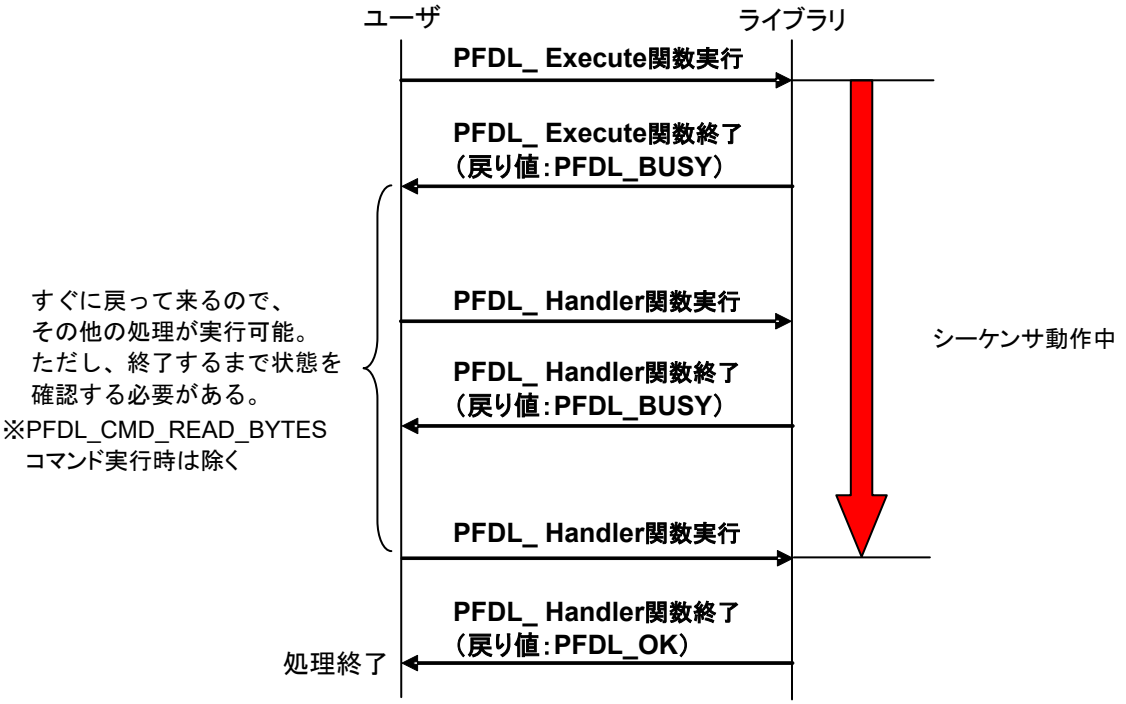


図 3−2 BGO 時の動作例 2（Erase，Iverify，BlankCheck）

表 3−3 データ・フラッシュ・ライブラリ関数の割り込み受け付けと BGO の一覧

関数名	シーケンサ制御／BGO機能	割り込みの受付
PFDL_Open	なし	可
PFDL_Close		
PFDL_Execute	あり注	
PFDL_Handler		
PFDL_GetVersionString	なし	

注　PFDL_CMD_READ_BYTESコマンド実行時は除く。

3. 5　データ型、戻り値の定義、戻り値の型一覧

データ型は次のとおりです。

表 3−4 データ型一覧

定　　義	データ・タイプ	説　　明
pfdl_u08	unsigned char	1バイト(8ビット)符号なし整数
pfdl_u16	unsigned int	2バイト(16ビット)符号なし整数
pfdl_u32	unsigned long int	4バイト(32ビット)符号なし整数

各戻り値の意味は次のとおりです。

表 3−5 戻り値の定義(pfdl_status_t)一覧

定　　義	戻り値	説　　明
PFDL_OK	0x00	正常終了
PFDL_ERR_ERASE	0x1A	消去エラー - 対象領域の消去に失敗した。
PFDL_ERR_MARGIN	0x1B	ブランク・チェック・エラー or 内部ベリファイ・エラー - 対象領域はブランク状態（書き込み可能な領域）ではない。 - 対象領域の内部ベリファイ処理中にエラーが発生した。
PFDL_ERR_WRITE	0x1C	書き込みエラー - 対象領域の書き込みに失敗した。
PFDL_IDLE	0x30	アイドル状態 - PFDL_Execute関数でコマンドは実行されていない。
PFDL_BUSY	0xFF	PFDL_Execute関数のコマンド実行開始、もしくは実行中 - PFDL_Execute関数で指定されたコマンドを実行中
上記以外	上記以外	その他のエラー - 異常な戻り値です。指定したコマンドやリソースの配置を再確認してください。

★　戻り値に使用する汎用レジスタは、RENESAS製 CA78K0R コンパイラとRENESAS製 CC-RL コンパイラ、およびLLVM コンパイラで異なります。

コンパイラ毎の戻り値と使用する汎用レジスタは、次のとおりです。

表 3−6 コンパイラ毎の戻り値一覧

開発ツール	戻り値の型	
	C言語	アセンブリ言語
RENESAS製 CA78K0R コンパイラ	pfdl_status_t	C (汎用レジスタ)
RENESAS製 CC-RL コンパイラ	pfdl_status_t	A (汎用レジスタ)
★ LLVM コンパイラ	pfdl_status_t	A (汎用レジスタ)

3. 6　データ・フラッシュ・ライブラリ関数の説明

フラッシュ関数について、次の形式で説明します。

データ・フラッシュ・ライブラリ関数名

【概　要】

この関数の機能概要を示します。

【書　式】

＜C言語＞

この関数をC言語で記述されたユーザプログラムから呼び出す際の書式を示します。

＜アセンブラ＞

この関数をアセンブリ言語で記述されたユーザプログラムから呼び出す際の書式を示します。

【事前設定】

この関数の事前設定を示します。

【機　能】

この関数の機能詳細と注意事項を示します。

【呼び出し後のレジスタ状態】

この関数を呼び出した後のレジスタの状態を示します。

【引　数】

この関数の引数を示します。

【戻り値】

この関数からの戻り値を示します。

PFDL_Open

【概　要】

データ・フラッシュ・ライブラリで使用するRAMの初期化と開始

【書　式】

＜C言語＞

RENESAS製 CA78K0R コンパイラ:

```
pfdl_status_t   __far PFDL_Open( __near pfdl_descriptor_t* descriptor_pstr )
```

RENESAS製 CC-RL コンパイラ:

```
pfdl_status_t   __far PFDL_Open( __near pfdl_descriptor_t* descriptor_pstr )
```

★ LLVM コンパイラ:

```
pfdl_status_t   __far PFDL_Open( __near pfdl_descriptor_t* descriptor_pstr )
                                                    __attribute__ ((section ("PFDL_COD")))
```

＜アセンブラ＞

```
CALL !PFDL_Open または CALL !!PFDL_Open
```

備考　データ・フラッシュ・ライブラリを00000H-0FFFFHに配置する場合は“！”、それ以外の場合は“！！”で呼び出してください。

【事前設定】

・フラッシュ・セルフ・プログラミング・ライブラリ、もしくはデータ・フラッシュ・メモリを操作するためのプログラムやデータ・フラッシュ・ライブラリ、およびEEPROMエミュレーション・ライブラリが未実行、あるいは終了していること。

・高速オンチップ・オシレータを起動しておくこと。

【機　能】

・データ・フラッシュ・コントロール・レジスタ（DFLCTL）をデータ・フラッシュ・メモリへのアクセス許可状態（DFLEN = 1）に設定します。

・データ・フラッシュ・ライブラリ Type04で使用するセルフRAMの確保、初期化、および開始処理を行います。セルフRAM[注1]が存在する場合は、データ・フラッシュ・ライブラリ Type04が終了するまでセルフRAMを使用しないで下さい。

・引数pfdl_descriptor_tの構造体メンバ、wide_voltage_mode_u08でデータ・フラッシュ・ライブラリ Type04のフラッシュ書き換えモード[注2]を定義します。

　　0x00　　：フルスピード・モード

　　0x01　　：ワイド・ボルテージ・モード

・引数pfdl_descriptor_tの構造体メンバ、fx_MHz_u08にCPUの動作周波数を設定します。設定値はデータ・フラッシュ・ライブラリ Type04内部のタイミング・データの計算に使用します。[注3]

設定するCPUの動作周波数の値（fx_MHz_u08）については、以下の点に注意してください。

- CPUの動作周波数を 4MHz未満[注4]で使用する場合は、1MHz, 2MHz, 3MHz を使用することができます。(1.5MHzのように整数値でない周波数は使用できません)初期化関数で設定するCPUの動作周波数も 1, 2, 3 の整数値を設定してください。

- CPUの動作周波数を 4MHz以上[注4]で使用する場合は、小数点以下の数値を持つ周波数を使用することができます。ただし、初期化関数（PFDL_Open）で設定するCPUの動作周波数は、小数点以下を切り上げた整数値を設定してください。（例：4.5MHz の場合は、初期化関数で 5 を設定してください）
- 高速オンチップ・オシレータの動作周波数ではありません。

注　1.　セルフRAMに関しては**2．　2　ソフトウェア環境**の項を参照してください。

2.　フラッシュ書き換えモードの詳細については、対象の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。

3.　データ・フラッシュ・ライブラリ Type04内部のタイミング計算に使用されるために必要なパラメータとなります。この設定によってCPUの動作周波数が変わる事はありません。

4.　CPUの動作周波数の範囲については対象の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。

注意　本関数を実行後、PFDL_Close関数を実行し、データ・フラッシュ・ライブラリ Type04を終了するまで、データ・フラッシュ・コントロール・レジスタ（DFLCTL）をデータ・フラッシュ・メモリへのアクセス禁止状態（DFLEN = 0）に設定しないでください。

【呼び出し後のレジスタ状態】

開発ツール	戻り値	破壊レジスタ
RENESAS製 CA78K0R コンパイラ	C (汎用レジスタ)	AX
RENESAS製 CC-RL コンパイラ	A (汎用レジスタ)	X, HL, C
★ LLVM コンパイラ	A (汎用レジスタ)	X, HL, C

【引　数】

引数の定義

引　　数	説　　明
__near pfdl_descriptor_t* descriptor_pstr	データ・フラッシュ・ライブラリ Type04の初期設定値 （フラッシュ書き換えモード、CPUの動作周波数）

__near pfdl_descriptor_t の定義

開発ツール	C言語（構造体の定義）	アセンブリ言語（変数の定義例）
★ RENESAS製 CA78K0R コンパイラ	typedef struct { pfdl_u08　fx_MHz_u08; pfdl_u08　wide_voltage_mode_u08; } pfdl_descriptor_t;	_descriptor_pstr: _fx_MHz_u08 :　DS 1 _wide_voltage_mode_u08 :　DS 1
RENESAS製 CC-RL コンパイラ	typedef struct { pfdl_u08　fx_MHz_u08; pfdl_u08　wide_voltage_mode_u08; } pfdl_descriptor_t;	_descriptor_pstr: _fx_MHz_u08 :　.DS 1 _wide_voltage_mode_u08 :　.DS 1
LLVM コンパイラ	typedef struct { pfdl_u08　fx_MHz_u08; pfdl_u08　wide_voltage_mode_u08; } pfdl_descriptor_t;	_descriptor_pstr: _fx_MHz_u08 :　.skip 1 _wide_voltage_mode_u08 :　.skip 1

__near pfdl_descriptor_t のパラメータ内容

構造体メンバ	説　　明
fx_MHz_u08	データ・フラッシュ・ライブラリ Type04実行中のCPUの動作周波数
wide_voltage_mode_u08	フラッシュ書き換えモードの設定

引数の設定内容

開発ツール	引数型／レジスタ	
	C言語	アセンブリ言語
RENESAS製 CA78K0R コンパイラ	__near pfdl_descriptor_t*　descriptor_pstr	AX(0-15)：変数の先頭アドレス（16bit）
RENESAS製 CC-RL コンパイラ	__near pfdl_descriptor_t*　descriptor_pstr	AX(0-15)：変数の先頭アドレス（16bit）
★ LLVM コンパイラ	__near pfdl_descriptor_t*　descriptor_pstr	AX(0-15)：変数の先頭アドレス（16bit）

【戻り値】

状　　態	説　　明
0x00(PFDL_OK)	正常終了 - 初期設定が完了した状態 (正常終了以外のパラメータはありません)

PFDL_Close

【概　要】

データ・フラッシュ・ライブラリの終了

【書　式】

＜C言語＞

RENESAS製 CA78K0R コンパイラ

```
void __far PFDL_Close( void )
```

RENESAS製 CC-RL コンパイラ:

```
void __far PFDL_Close( void )
```

★ LLVM コンパイラ:

```
void __far PFDL_Close( void ) __attribute__ ((section ("PFDL_COD")))
```

＜アセンブラ＞

```
CALL !PFDL_Close または CALL !!PFDL_Close
```

備考　データ・フラッシュ・ライブラリを00000H-0FFFFHに配置する場合は“！”、それ以外の場合は“！！”で呼び出してください。

【事前設定】

この関数の実行前にPFDL_Open関数を正常終了させてください。

【機　能】

・データ・フラッシュ・コントロール・レジスタ（DFLCTL）をデータ・フラッシュ・メモリへのアクセス禁止状態（DFLEN = 0）に設定します。
データ・フラッシュ・ライブラリ Type04終了後もデータ・フラッシュ・メモリへのアクセスが必要な場合は、PFDL_Close関数実行後に、DFLCTLをデータ・フラッシュ・メモリのアクセス許可に設定し、セットアップ処理[注]を行ってください。
・データ・フラッシュ・ライブラリ Type04を終了状態にします。

注.　セットアップ方法については対象の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。

【呼び出し後のレジスタ状態】

開発ツール	戻り値	破壊レジスタ
RENESAS製 CA78K0R コンパイラ	—	—
RENESAS製 CC-RL コンパイラ	—	C
★ LLVM コンパイラ	—	C

【引　数】

なし

【戻り値】

なし

PFDL_Execute

【概　要】

データ・フラッシュ・メモリへの制御実行

【書　式】

＜C言語＞

RENESAS製 CA78K0R コンパイラ

```
pfdl_status_t   __far PFDL_Execute(__near pfdl_request_t* request_pstr)
```

RENESAS製 CC-RL コンパイラ:

```
pfdl_status_t   __far PFDL_Execute(__near pfdl_request_t* request_pstr)
```

★ LLVM コンパイラ:

```
pfdl_status_t   __far PFDL_Execute(__near pfdl_request_t* request_pstr)
                                                    __attribute__ ((section ("PFDL_COD")))
```

＜アセンブラ＞

```
CALL !PFDL_Execute または CALL !!PFDL_Execute
```

備考　データ・フラッシュ・ライブラリを00000H-0FFFFHに配置する場合は“！”、それ以外の場合は“！！”で呼び出してください。

【事前設定】

この関数の実行前にPFDL_Open関数を正常終了させてください。

【機　能】

指定されたコマンドに従い、データ・フラッシュ・メモリへの制御を実行します。

【呼び出し後のレジスタ状態】

開発ツール	戻り値	破壊レジスタ
RENESAS製 CA78K0R コンパイラ	C (汎用レジスタ)	AX
RENESAS製 CC-RL コンパイラ	A (汎用レジスタ)	X, BC, DE, HL
★ LLVM コンパイラ	A (汎用レジスタ)	X, BC, DE, HL

【引　数】

引数の定義

引　　数	説　　明
__near pfdl_request_t* request_pstr[注]	リクエスタ：データ・フラッシュ・メモリへの制御内容を指定（コマンドと設定値）

注　本関数の引数として使用する前に必要の無いパラメータに関しても一度「0」などの値を入力し、変数領域を初期化してください。一度も初期化されていない領域がある変数を使用した場合、RAMパリティ・エラーが検出され、RL78マイクロコントローラにリセットが発生する可能性があります。
RAMパリティ・エラーに関しては、対象の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。

__near pfdl_request_t の定義

★

開発ツール	C言語（構造体の定義）	アセンブリ言語（変数の定義例）
RENESAS製 CA78K0R コンパイラ	typedef struct { pfdl_u16 index_u16; __near pfdl_u08* data_pu08; pfdl_u16 bytecount_u16; pfdl_command_t command_enu; }pfdl_request_t	_request_pstr: _index_u16 : DS2 _data_pu08 : DS2 _bytecount_u16 : DS2 _command_enu : DS1
RENESAS製 CC-RL コンパイラ	typedef struct { pfdl_u16 index_u16; __near pfdl_u08* data_pu08; pfdl_u16 bytecount_u16; pfdl_command_t command_enu; }pfdl_request_t	_request_pstr: _index_u16 : .DS2 _data_pu08 : .DS2 _bytecount_u16 : .DS2 _command_enu : .DS1
LLVM コンパイラ	typedef struct { pfdl_u16 index_u16; __near pfdl_u08* data_pu08; pfdl_u16 bytecount_u16; pfdl_command_t command_enu; }pfdl_request_t	_request_pstr: _index_u16 : .skip 2 _data_pu08 : .skip 2 _bytecount_u16 : .skip 2 _command_enu : .skip 1

__near pfdl_request_t のパラメータ内容

構造体メンバ	説　明
pfdl_u16 index_u16	対象領域の先頭アドレス、またはブロック番号設定[注1] - 消去　：ブロック番号[注1] - 消去以外：対象領域の先頭アドレス[注1,3]
__near pfdl_u08* data_pu08	書き込み、読み込みデータ取得用データ・バッファへのポインタ[注1,2] 書き込み、読み込み以外の処理では使用しません
pfdl_u16 bytecount_u16	コマンドの実行範囲 (バイト指定) [注1,2] - 消去　：設定する必要はありません - 消去以外：指定した先頭アドレス[注1]から対象となる領域の範囲
pfdl_command_t command_enu	実行させるコマンド

注 1. 値を設定する必要が無いパラメータについても、本関数の引数として使用する前に一度「0」等を入力して変数領域を初期化してください。一度も初期化されていない領域のある変数を使用した場合、RAMパリティ・エラーが検出され、RL78マイクロコントローラにリセットが発生する可能性があります。
RAMパリティ・エラーについては、対象の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。

2. 対象パラメータが必要なコマンドの場合に設定します。また、データ・バッファのサイズについては、書き込み、読み込みデータのバイト・サイズ分をご用意ください。

3. 指定するアドレス値は、データ・フラッシュ・メモリのブロック0を開始アドレス（アドレス 0）とした相対アドレスです。

引数の設定内容

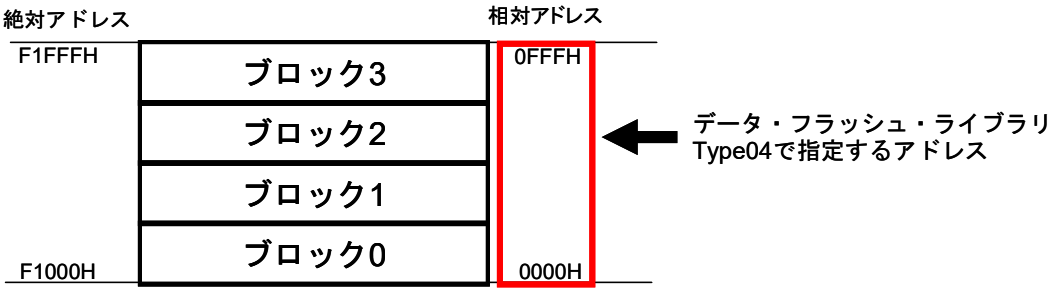
開発ツール	引数型／レジスタ	
	C言語	アセンブリ言語
RENESAS製 CA78K0R コンパイラ	__near pfdl_request_t* request_pstr	AX(0-15) : 変数の先頭アドレス（16bit）
RENESAS製 CC-RL コンパイラ	__near pfdl_request_t* request_pstr	AX(0-15) : 変数の先頭アドレス（16bit）
★ LLVM コンパイラ	__near pfdl_request_t* request_pstr	AX(0-15) : 変数の先頭アドレス（16bit）

実行コマンド(pfdl_command_t)一覧

コマンド	値	説　明
PFDL_CMD_READ_BYTES	0x00	データ・フラッシュ・メモリの指定された先頭アドレス[注2]から、読み込みサイズ分のデータを読み込みデータ入力バッファに読み出します。 読み出し処理はバック・グラウンド・オペレーション(BGO)では無いため、PFDL_Execute関数内で全ての処理を実行しますので、PFDL_Handler関数を実行する必要はありません。 ※ 実行には以下の引数の設定が必要です ・pfdl_request_t.index_u16　：読み込み開始アドレス[注2] ・pfdl_request_t.bytecount_u16：読み込みサイズ[注1] ・pfdl_request_t.data_pu08　：読み込みデータ入力バッファのアドレス
PFDL_CMD_BLANKCHECK_BYTES	0x08	データ・フラッシュ・メモリの指定された先頭アドレス[注2]から、実行範囲分の領域が消去レベル（書き込み可能な領域）であることを確認します。（データ・リードによる確認では、消去レベルの確認は行えません） エラーの場合、対象領域に対する書き込みは実行できません。エラーが発生した領域に書き込みを行う場合は、消去コマンド（PFDL_CMD_ERASE_BLOCK）を実行する必要があります。 ※ 実行には以下の引数の設定が必要です ・pfdl_request_t.index_u16　：開始アドレス[注2] ・pfdl_request_t.bytecount_u16：開始アドレスからの実行範囲[注1]
PFDL_CMD_ERASE_BLOCK	0x03	指定した番号のデータ・フラッシュ・メモリのブロックを消去します。 エラーの場合、対象ブロックに対する書き込みは実行できません。エラーが発生したブロックに書き込みを行う場合は、再度本コマンドを実行し、正常終了させる必要があります。 ※ 実行には以下の引数の設定が必要です ・pfdl_request_t.index_u16　：ブロック番号
PFDL_CMD_WRITE_BYTES	0x04	書き込みデータ入力バッファに入力されているデータを、指定した先頭アドレス[注2]から、書き込みサイズ分、データ・フラッシュ・メモリへ書き込みを行います。データ・フラッシュ・メモリへの書き込みはブランク状態か、もしくは消去済みの領域へのみ行えます。既に書き込みを行った領域に対し、消去を行わずに再度書き込み(上書き)を行う事はできません。 書き込みを実施した領域に対しては、内部ベリファイ・コマンド（PFDL_CMD_IVERIFY_BYTES）を実行し、データ・フラッシュ・メモリの状態を確認してください。 ※ 実行には以下の引数の設定が必要です ・pfdl_request_t.index_u16　：書き込み開始アドレス[注2] ・pfdl_request_t.bytecount_u16：書き込みサイズ[注1] ・pfdl_request_t.data_pu08　：書き込みデータ入力バッファのアドレス
PFDL_CMD_IVERIFY_BYTES	0x06	データ・フラッシュ・メモリの指定された先頭アドレス[注2]から、実行範囲分の領域に内部ベリファイを行います。このベリファイは、指定された範囲のフラッシュ・メモリのセルの信号レベルが、適正であるかを確認するための処理です。 エラーが発生した場合、書き込まれたデータは保証されません。対象領域に再度書き込みを行う場合は、消去コマンド （PFDL_CMD_ERASE_BLOCK）を実行してください。 ※ 実行には以下の引数の設定が必要です ・pfdl_request_t.index_u16　：開始アドレス[注2] ・pfdl_request_t.bytecount_u16：開始アドレスからの実行範囲[注1]

注 1.　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。

2.　データ・フラッシュ・ライブラリ Type04を使用してデータ・フラッシュ・メモリに書き込みや読み込みを行う場合に指定するアドレス値はデータ・フラッシュ・メモリのブロック0を開始アドレス(アドレス0)とした相対アドレスです。絶対アドレスではありませんので、ご注意ください。



【戻り値】

状　態	説　明
0x00(PFDL_OK)	正常終了
0x1A(PFDL_ERR_ERASE)	消去エラー - 消去処理中にエラーが発生した
0x1B(PFDL_ERR_MARGIN)	ブランク・チェック・エラー or 内部ベリファイ・エラー - 対象領域はブランク状態（書き込み可能な領域）ではない - 対象領域の内部ベリファイ処理中にエラーが発生した
0x1C(PFDL_ERR_WRITE)	書き込みエラー - 書き込み処理中にエラーが発生した
0xFF(PFDL_BUSY)	指定したコマンドの実行開始 - 指定したコマンドが実行開始された （実行状態をPFDL_Handler関数により確認してください）

PFDL_Handler

【概　要】

データ・フラッシュ・メモリへの制御状態の確認と継続実行の設定（ステータス・チェック処理）

【書　式】

＜C言語＞

RENESAS製 CA78K0R コンパイラ

```
pfdl_status_t　__far PFDL_Handler( void )
```

RENESAS製 CC-RL コンパイラ:

```
pfdl_status_t　__far PFDL_Handler( void )
```

LLVM コンパイラ:

★

```
pfdl_status_t　__far PFDL_Handler( void ) __attribute__ ((section ("PFDL_COD")))
```

＜アセンブラ＞

```
CALL !PFDL_Handler または CALL !!PFDL_Handler
```

備考　データ・フラッシュ・ライブラリを00000H-0FFFFHに配置する場合は“！”、それ以外の場合は“！！”で呼び出してください。

【事前設定】

この関数の実行前にPFDL_Open関数を正常終了させてください。

【機　能】

直前に実行したPFDL_Execute関数（PFDL_CMD_READ_BYTESコマンド以外）で指定したコマンドの制御状態を確認し、継続実行に必要な設定を行います。

【呼び出し後のレジスタ状態】

開発ツール	戻り値	破壊レジスタ
RENESAS製 CA78K0R コンパイラ	C (汎用レジスタ)	—
RENESAS製 CC-RL コンパイラ	A (汎用レジスタ)	C
★ LLVM コンパイラ	A (汎用レジスタ)	C

【引　数】

なし

【戻り値】

状　態	説　明
0x00(PFDL_OK)	正常終了
0x1A(PFDL_ERR_ERASE)	消去エラー - 消去処理中にエラーが発生した
0x1B(PFDL_ERR_MARGIN)	ブランク・チェック・エラー or 内部ベリファイ・エラー - 対象領域はブランク状態（書き込み可能な領域）ではない - 対象領域の内部ベリファイ処理中にエラーが発生した
0x1C(PFDL_ERR_WRITE)	書き込みエラー - 書き込み処理中にエラーが発生した
0x30(PFDL_IDLE)	アイドル状態 - PFDL_Execute関数でコマンドは実行されていない
0xFF(PFDL_BUSY)	コマンド実行中 - PFDL_Execute関数で指定されたコマンドを実行中

PFDL_GetVersionString

【概　要】

データ・フラッシュ・ライブラリ Type04のバージョン情報を取得

【書　式】

＜C言語＞

RENESAS製 CA78K0R コンパイラ

```
__far pfdl_u08* __far PFDL_GetVersionString( void )
```

RENESAS製 CC-RL コンパイラ:

```
__far pfdl_u08* __far PFDL_GetVersionString( void )
```

LLVM コンパイラ:

★

```
__far pfdl_u08* __far PFDL_GetVersionString( void ) __attribute__ ((section ("PFDL_COD")))
```

＜アセンブラ＞

```
CALL !PFDL_GetVersionString または CALL !!PFDL_GetVersionString
```

備考　データ・フラッシュ・ライブラリを00000H-0FFFFHに配置する場合は“！”、それ以外の場合は“！！”で呼び出してください。

【事前設定】

なし

【機　能】

データ・フラッシュ・ライブラリ Type04 のバージョン情報を取得。

【呼び出し後のレジスタ状態】

開発ツール	戻り値	破壊レジスタ
RENESAS製 CA78K0R コンパイラ	BC(0-15), DE(16-31)	—
RENESAS製 CC-RL コンパイラ	DE(0-15), A(16-23)	—
★ LLVM コンパイラ	DE(0-15), A(16-23)	—

【引　数】

なし

【戻り値】

データ型	説　明
★ __far pfdl_u08*	・データ・フラッシュ・ライブラリ Type04のバージョン情報が格納されている領域の先頭アドレス(far領域) ・ライブラリのバージョン情報の各文字はASCIIコードです。 例：データ・フラッシュ・ライブラリ Type04の場合 “DRL78T04LyyyzGVxxx” バージョン情報　：例：V105 → Ver.1.05 コンパイラ情報（５~６文字）　：CA78K0R [例:RyyyG] CC-RL, LLVM [例:LyyyzG] Type 番号（３文字）　：Type04 対応デバイスファミリ（４文字）：RL78 対象ライブラリ（１文字）　：'D'は FDL

付録A　改版履歴

A. 1　本版で改訂された主な箇所

箇　所	内　容	分類
全般		
－	LLVMコンパイラ用データ・フラッシュ・ライブラリの情報を新規追加	(b)
－	図番号とタイトルを図の上部から下部へ移動	(c)
第2章 プログラミング環境		
p.18	LLVMコンパイラについての説明を追記	(c)
p.18, p20	表2-7、表2-8にLLVMコンパイラ用のソフトウェアリソースを追記	(c)
p.18, P20-p22	LLVMコンパイラ使用時に必要となるスタックの情報を追記	(c)
p.23	LLVMコンパイラ使用時にセルフRAM領域を確保する場合の説明を追記	(c)
p.23	LLVMコンパイラ使用時にスタックを設定する場合の説明を追記	(c)
第3章 データ・フラッシュ・ライブラリ関数		
p.28	LLVMコンパイラについての説明を追記	(c)
p.32 - p.45	各ライブラリ関数にLLVMコンパイラ用の戻り値、C言語書式、引数の定義と設定内容を追記	(c)

備考　表中の「分類」により、改訂内容を次のように区分しています。

（a）：誤記訂正、（b）：仕様（スペック含む）の追加／変更、（c）：説明，注意事項の追加／変更、

（d）：パッケージ、オーダ名称、管理区分の追加／変更、（e）：関連資料の追加／変更

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。なお、適用箇所は各版での章を示します。

（1/4）

版　数	内　容	適用箇所
Rev.1.06	CC-RLコンパイラ用データ・フラッシュ・ライブラリのユーザーズマニュアルと統合	全般
	対象MCUについてのセルフRAMリスト参照説明を変更(「このマニュアルの使い方」を参照)	
	電圧モードを全てフラッシュ書き換えモードに表記統一	
	ZIPファイル名をインストーラ名に変更	表紙
	既に書き込みを行っている領域についての注意事項の補足説明を追記	第1章 概説
	対応コンパイラについての説明を追記	第2章 プログラミング環境
	表2-7、表2-8にCC-RLコンパイラ用のソフトウェアリソースを追記	
	CC-RLコンパイラ使用時に必要となるスタックの情報を追記	
	CC-RLコンパイラ使用時にセルフRAM領域を確保する場合の説明を追記	
	CC-RLコンパイラ使用時にスタックを設定する場合の説明を追記	
	高速オンチップ・オシレータに関する注意事項を追記	
	64KB境界に跨いで各セグメントが配置できないことを追記	
	CC-RLのアセンブラでの16進数表現の設定方法について追記	
	3.2の題名を変更(セクションを追加)、指定領域への配置について、説明を見直し、修正	第3章 データ・フラッシュ・ライブラリ関数
	各ライブラリ関数の戻り値、C言語書式、引数の定義と設定内容を追記および変更	

（2/4）

版　数	内　容	適用箇所
Rev.1.05	対応デバイスを削除	全般
	対象MCUについてのリスト参照説明を追加	
	電圧モードの表記を削除。フラッシュ書き換えモードに表記統一	
	動作周波数の記載について、説明毎に表記方法が異なっていた部分をCPUの動作周波数に表記統一	
	フラッシュ関数をRAMで実行した場合の説明を追加	第2章 プログラミング環境
	(3)総合処理時間 についての説明を見直し、修正	
	表2-5,表2-6のReferenceの表記と計算式をTypicalの表記と計算式に変更	
	表2-7 セルフRAMサイズを変更	
	表2-7 セルフRAMの使用領域に記述を変更	
	表2-7 注1で問合せに関する注意事項を変更	
	表2-7 注4の記載を削除	
	セルフRAMに関する説明を見直し、修正	
	実行コマンド(pfdl_command_t)一覧のPFDL_CMD_READ_BYTESでPFDL_Handler実行不要を追加	第3章 データ・フラッシュ・ライブラリ関数
	PFDL_Handler関数実行が必要なのは、PFDL_CMD_READ_BYTES以外であることを追加	

（3/4）

版　数	内　容	適用箇所
Rev.1.04	表紙に対応ZIPファイル名、リリース版を明記	全般
	RL78/L13を対応デバイスに追加	
	リソースにRL78/L13の項目を追加。 ライブラリ・サイズを変更。	第2章 プログラミング環境
	データ・バッファに関する注意事項を追加	
	PFDL_Execute関数のコマンド説明を追加	第3章 データ・フラッシュ・ライブラリ関数
	索引を削除	

（4/4）

版　数	内　容	適用箇所
Rev.1.03	フラッシュ・ライブラリのドキュメントの扱いをアプリケーション・ノート（旧版R01AN0608）からユーザーズマニュアルへ変更	全般
	フラッシュ・データ・ライブラリからデータ・フラッシュ・ライブラリへ名称変更	
	処理時間、並びにソフトウェアリソースの内容を使用上の留意点から本書に移動。また、それに伴って前述の内容に対する本書の参照先を変更	
	高速OCOの表記を削除。高速オンチップ・オシレータに表記統一	
	動作周波数の記載について、説明毎に表記方法が異なっていた部分をCPUの動作周波数に表記統一	
	関数実行時のアセンブリ言語の書式誤りを修正	
	ブランク状態、ブランク・チェックの用語に補足を追加	
	データ・フラッシュ・コントロール・レジスタ（DFLCTL）への制御内容を追加	第1章 概 説
	書き込み時の注意事項を追加	
	R5F10266製品使用時に割り込みが禁止となる注意事項を追加	第2章 プログラミング環境
	初期設定に関する説明を追加	
	データ・フラッシュ・コントロール・レジスタ（DFLCTL）の説明を追加	
	データ・フラッシュ・ライブラリで指定するアドレスに関しての注意事項を追加	
	処理時間に関する項目を追加（使用上の留意点から処理時間に関する記述を本書に移動）	
	Erase，BlankCheckのReference(旧Min)時間を削除	
	リソースに関する項目を追加（使用上の留意点からリソースに関する記述を本書に移動）	
	R5F10266製品使用時のリソース内容を追加	
	高速オンチップ・オシレータの周波数に関する注意事項を追加	
	DFLCTL（データ・フラッシュ・コントロール・レジスタ）に関する注意事項を追加	
	RAMパリティ・エラーに関する注意事項を追加	
	書き込み時の注意事項を追加	
	DFLCTL（データ・フラッシュ・コントロール・レジスタ）に関する注意事項を追加	
	R5F10266製品使用時に割り込みが禁止となる注意事項を追加	
	DFLCTL（データ・フラッシュ・コントロール・レジスタ）に関する記述を追加	第3章 データ・フラッシュ・ライブラリ関数
	ワイド・ボルテージ・モードの設定値を「00H以外」から，「01H」に変更 (02H～FFHの値でも正常に設定されるが、仕様上の定義値は01Hのため、記載内容を変更)	
	高速オンチップ・オシレータの周波数に関する注意事項を追加	
	DFLCTL（データ・フラッシュ・コントロール・レジスタ）に関する注意事項を追加	
	構造体の定義、引数の設定方法の表を追加・修正	
	DFLCTL（データ・フラッシュ・コントロール・レジスタ）に関する記述を追加	
	DFLCTL（データ・フラッシュ・コントロール・レジスタ）に関する注意事項を追加	
	RAMパリティ・エラーに関する注意事項を追加	
	構造体の定義の表を追加	
	引数/レジスタ型の表を追加	
	データ・フラッシュ・ライブラリで指定するアドレスに関しての注意事項を追加	
	書き込み時の注意事項を追記	
	内部ベリファイに関する注意事項（説明）を追加	
	索引を修正	

RL78ファミリ ユーザーズマニュアル
データ・フラッシュ・ライブラリ

発行年月日　2012年 4月 6日　Rev.1.01
2018年10月31日　Rev.1.06
2023年12月26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）

RL78 ファミリ
データ・フラッシュ・ライブラリ Type04

ルネサス エレクトロニクス株式会社

R01US0049JJ0110