RL78/L13

R01AN2014JJ0100
Rev.1.00
2014.6.25

データ・フラッシュ・メモリを用いた外付け EEPROM IC 機能の取り込み（EEPROM エミュレーション・ライブラリ編）

要旨

セルフ・プログラミングは、マイコン搭載のフラッシュ・メモリをマイコン自身で書き換える事が出来る機能です。RL78/L13 にはデータ保存に適したデータ・フラッシュ・メモリを搭載しており、データ・フラッシュ・メモリの書き換えは、ルネサスが提供するフラッシュ・データ・ライブラリ(以降 FDL)及び EEPROM エミュレーション・ライブラリ(以降 EEL)で実現する事が出来ます。

本アプリケーションノートでは、不揮発データの保持について、外付け EEPROM IC を使わずに、データ・フラッシュ・メモリと EEL で簡単に実現する方法を説明します。また、電源電圧の低下を検知し、データを素早くデータ・フラッシュ・メモリに退避させ、電源断に備える方法についても説明します。

このアプリケーションノートを適用すると、ユーザは、外付け EEPROM IC の機能をマイコン内に取り込む事が可能です

コンパイラと対応する EEL について

本アプリケーションノートには、サンプルコード（EEL を除く）が付属されています。サンプルコードを動作させるためには、別途、EEL をダウンロードしプロジェクトに登録/リンクさせてください。プロジェクトへの登録/リンク方法は「6.9　EEL の取り込み方」を参照してください。

EEL には CubeSuite+、IAR 版があります。ただし、販売会社毎（地域毎）にサポートしている EEL が異なります。当社ウェブページ(http://www.renesas.com)で地域を選択し、サポートされている EEL を確認してください。また、EEL を使用する前に、EEL のマニュアル、リリースノート（またはダウンロード元の README.txt）を確認してください。

コンパイラと EEL の対応関係

コンパイラの種類	対応する EEL	ダウンロード元
CubeSuite+ 版	RL78 ファミリ用 EEPROM エミュレーション・ライブラリ Pack02 Ver.1.01	http://japan.renesas.com/products/tools/flash_programming/flash_libraries/data_flash_lib/downloads.jsp
	RENESAS_EEL_RL78_T02E_V1.10 （別途、同ダウンロード先でお求めできる下記 FDL とリンクしてご使用ください。FDLRENESAS_FDL_RL78_T02E_V1.10 ）	http://www.renesas.eu/updates?oc=EEPROM_EMULATION_RL78
IAR 版	RENESAS_EEL_RL78_T02E_V1.10 （別途、同ダウンロード先でお求めできる下記 FDL とリンクしてご使用ください。FDLRENESAS_FDL_RL78_T02E_V1.10）	http://www.renesas.eu/updates?oc=EEPROM_EMULATION_RL78

このアプリケーションノートの対象デバイス

本アプリケーションノートは RL78/L13 を対象にして開発したものです。

また、本アプリケーションノートで使用する EEL は RL78 の他のデバイスにも対応しています。

RL78/D1A、RL78/F12、RL78/F13、RL78/F14、RL78/G13、RL78/G14、RL78/G1A、RL78/G1E、RL78/I1A、RL78/L1C

EEL の対応デバイスについては、最新の EEL のユーザーズマニュアルでご確認ください。

本アプリケーションノートを他の RL78 マイコンに転用する場合は、十分に評価をして頂いた上でご使用ください。

目次

1. はじめに 5
1.1 EEL 概要 5
1.2 FDL 概要 5
1.3 FDL と EEL の使い分け 6
1.4 EEPROM IC からの置き換えのメリットと注意事項 9
1.4.1 EEPROM IC に対するメリット 9
1.4.2 EEPROM IC との違い 9

2. 仕様 11
2.1 EEL 書き込み時間の短縮 14
2.2 EEL アーキテクチャ 16
2.2.1 EEL プール 16
2.2.2 EEL ブロック 19
2.3 EEL ユーザ設定初期値 21
2.4 格納ユーザ・データ数とユーザ・データの合計サイズ 24
2.5 EEL 使用時の注意事項 26

3. 動作確認条件 27

4. 関連アプリケーションノート 28

5. ハードウェア説明 29
5.1 ハードウェア構成例 29
5.2 使用端子一覧 29

6. ソフトウェア説明 30
6.1 動作概要 30
6.2 ファイル構成 33
6.3 オプション・バイトの設定 34
6.4 定数一覧 35
6.5 変数一覧 36
6.6 関数一覧 37
6.7 関数仕様 38
6.8 フローチャート 46
6.8.1 全体フローチャート 46
6.8.2 周辺機能初期設定 46
6.8.3 ポート初期設定 47
6.8.4 CPU クロック初期設定 48
6.8.5 TAU0 初期設定 49
6.8.6 INTP 初期設定 51
6.8.7 LVD 初期設定 51
6.8.8 メイン処理 52
6.8.9 メイン初期化処理 54
6.8.10 EEL 初期化処理 55
6.8.11 EEL 読み出し 56
6.8.12 LED 点滅データ有効範囲チェック 56
6.8.13 EEL 関数ステータスチェック 57
6.8.14 TAU01 動作許可設定 58
6.8.15 TAU01 割り込みハンドラ 59
6.8.16 TAU01 動作禁止設定 60
6.8.17 INTP0 動作許可設定 60
6.8.18 INTP0 割り込みハンドラ 61
6.8.19 TAU00 動作許可設定 61
6.8.20 TAU00 割り込みハンドラ 62

6.8.21 EEL 書き込み 63
6.8.22 TAU00 動作禁止設定 63
6.8.23 INTP0 動作禁止設定 63
6.8.24 LVD 割り込み許可設定 64
6.8.25 LVD 割り込みハンドラ 64
6.9 EEL の取り込み方 65
6.9.1 CubeSuite+版 65
6.9.2 IAR 版 65
6.10 サンプルコードの修正について 66

7. サンプルコード 69

8. 参考ドキュメント 69

1. はじめに

セルフ・プログラミング・ライブラリには、フラッシュ・セルフ・プログラミング・ライブラリ（以降 FSL）、FDL、EEL の 3 種類があります。各ライブラリの一覧を表 1.1に記載します。

その内、データ・フラッシュ・メモリを扱うライブラリとして、EEL の概要を1.1に、FDL の概要を1.2 に記載します。尚、本アプリケーションノートでは表 1.1の太文字で記載している EEL について説明します。

表1.1 セルフ・プログラミング・ライブラリ一覧

ライブラリ名	対象フラッシュ・メモリ	説明
FSL	コード・フラッシュ・メモリ	コード・フラッシュ・メモリのデータを書き換えるためのライブラリ
FDL	**データ・フラッシュ・メモリ**	データ・フラッシュ・メモリのデータの書き換えや読み出しを行うためのライブラリ
EEL		**データ・フラッシュ・ライブラリを EEPROM のように使用し、データの書き換えや読み出しを行うためのライブラリ**

1.1　EEL 概要

EEL は、RL78 マイクロコントローラに搭載されたデータ・フラッシュ・メモリを EEPROM のようにデータを格納するためのソフトウェア・ライブラリです。EEL 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、EEL の初期化処理や使用する機能に対応する関数をユーザ・プログラムから呼び出します。

EEL では、データごとに 1 バイトの識別子（データ ID：1～64）をユーザが割り振り、割り振った識別子ごとに 1～255 バイトの任意の長さで読み出し／書き込みを行うことができます（識別子は最大 64 個まで扱うことができます）。

1.2　FDL 概要

FDL 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。FDL 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、FDL の初期化処理や使用する機能に対応する関数をユーザ・プログラムから呼び出します。

FDL の基本的な使い方として、書き込まれていない（ブランク状態の）データ・フラッシュ・メモリのアドレスに対して 1 バイト単位で書き込みを行います。ただし、同じアドレスに対して上書きすることができません。同じアドレスに対して上書きをする場合は、事前に上書き対象のブロックに対してブロック単位でデータを消去する必要があります。

1.3　FDL と EEL の使い分け

FDL と EEL は書き換え、使用リソース、実行時間、データ管理などが異なります。両者の主な特徴を表1.2に示します。

FDL はデータ・フラッシュ・メモリへの基本的なアクセス関数のみであるため、ユーザ・プログラム次第で柔軟にデータ管理の仕様を作ることが可能です。EEL はデータ管理の仕様が予め決まっている為、開発負荷が低いという特徴を持ちます。

アプリケーションの要件に応じて、FDL と EEL を選択してください。

表1. 2 FDL と EEL の特徴

	FDL	EEL
書き換え方式	ユーザ・プログラムに依存	アドレスを変更して書き込み
使用リソース	少ない	多い
データ・サイズ	最大 1024 バイト	最大 255 バイト
実行時間	短い	長い
データ管理方式	なし （ユーザがアドレスで管理）	あり （データ番号で管理）

注意　FDL の特徴は上位のアプリケーション(データ管理の仕様)に依存します。

(1)　書き換え方式

書き込み対象アドレスのデータ・フラッシュ・メモリが未使用状態の場合のみ書き込むことができます。同じアドレスに対して上書きする場合は、事前に上書き対象のブロックに対してブロック単位でデータを消去する必要があります。

FDL 単体では、データを管理する機構を持っていません。データ管理方式は、アプリケーション層（ユーザ）で考える必要があります。一方、EEL はデータ管理する機構を持っており、書き込み時には未使用状態のデータ・フラッシュ・メモリを示すアドレスにアドレスを変えながら書き込みを行います。書き込み対象のブロックがデータで満たされるまでデータを書き込めるため、比較的データ数や書き込み回数が多い用途に向いています。

(2) 使用リソース

FDL、EEL が必要とする各ソフトウェアリソースを**表 1.3**に示します。セルフ RAM、スタック、データ・バッファの 3 つは RAM を使用します。EEL は FDL を利用しているため、EEL の ROM リソースは FDL よりも多くなります。

表1.3 FDL/EEL のソフトウェアリソース（RL78/L13 の例）

項目	容量（バイト）	
	FDL	EEL
セルフ RAM 注1	0 ~ 1024	0 ~ 1024
スタック	MAX 46	MAX 80
データ・バッファ 注2	1 ~ 1024	1 ~ 255
ライブラリ・サイズ	ROM : MAX 177	ROM :MAX 3400 (FDL : 600、EEL : 2800)

注１．ワークエリアとして使用する領域をセルフ RAM と呼びます。セルフ RAM はマッピングされず、FDL/EEL 実行時に自動的に使用される領域のため、ユーザ設定は必要ありません。

注２．読み書きを行うデータを入力するために必要な RAM 領域をデータ・バッファと呼びます。必要となるサイズは、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、必要となるデータ・バッファは 1 バイトです。

注３．本表に記載のリソースは FDL RL78 Type04 Ver1.05、EEL RL78 Pack02 Ver1.01 におけるリソースです。今後、ライブラリのバージョンアップ等によって変動する可能性があります。最新のリソース情報は各ライブラリのマニュアルをご確認ください。

(3) データ・サイズ

FDL は最大 1024 バイト(データ・フラッシュ・メモリの 1 ブロック分)の読み書きが可能です。EEL は最大 255 バイトの読み書きが可能です。大きなデータを保存する場合は FDL が有利です。

なお、**表 1.3**のデータ・バッファは 1 度に読み書きできるデータのサイズを表しています。

(4) 実行時間

FDL と EEL のライブラリ関数実行時間を表 1.4に記載します。データ管理機構がない FDL の方が高速にデータの読み書きをすることが可能です。

表1. 4 FDL/EEL のライブラリ関数実行時間（動作周波数 24MHz、フルスピード・モードの例）

処理	FDL(255 バイト)	EEL(255 バイト)
書き込み FDL : PFDL_Execute(Write) EEL : EEL_Execute(Write)	519.7[μs]	11399.7[μs]
読み出し FDL : PFDL_Execute(Read) EEL : EEL_Execute(Read)	167.7[μs]	179.7[μs]
ベリファイ FDL : PFDL_Execute(IVerify) EEL : EEL_Execute(Verify)	959.7[μs]	3919.7[μs]

備考．本アプリケーションノート記載の実行時間は、統合開発環境 CubeSuite+上で FDL RL78 Type04 Ver1.05、EEL RL78 Pack02 Ver1.01 を動作させたときの実測値です。デバイスの個体差や実行条件によって値は変動します。

(5)　データ管理方式

FDL は、データ・フラッシュ・メモリへのアクセス方法としてアドレスを利用します。最新データが格納されているアドレスは変更されるため、アドレスを管理する必要があります。一方、EEL はデータ ID でデータを管理します。そのため、EEL では最新データが格納されているアドレスを管理する必要はありません。

1.4 EEPROM IC からの置き換えのメリットと注意事項

本節では、EEPROM IC の機能を、EEL を用いてデータ・フラッシュ・メモリで置き換えるときのメリットおよび EEPROM IC との違いを説明します。

1.4.1 EEPROM IC に対するメリット

EEPROM IC から置き換えるときのメリットを以下に示します。

- 外付け EEPROM IC が必要なくなるため、部品コスト削減、実装面積の削減ができます。
- デバイス内部で完結する動作であるため、シリアル通信を行う必要がありません。マイコンの通信ピンを他の機能で使用することができます。また、ソフト開発時もデバッガで書き込みされた値を直接確認することが可能です。
- 通信が必要ないため、処理時間を短くできます。（ただし、データ構造に依存します。）
  ※EEPROM IC では、シリアルでの通信時間＋書き込み完了時間（数ミリ秒）掛かります。
- データ ID を用いたデータ管理をしているため、アドレスを気にする必要がありません。

1.4.2 EEPROM IC との違い

EEPROM IC から置き換えるときの違いを以下に示します。

- エミュレーションであるため、データ・フラッシュ・メモリのサイズに対してユーザが使用できる領域が少なくなります。
  ※ユーザが使用できる領域の計算方法は「2.4

格納ユーザ・データ数とユーザ・データの合計サイズ」を参照してください。

- EEPROM IC との通信プログラムに代わり、FDL と EEL が必要になります。
- データ数が最大 64 個、1 個のデータの最大サイズは 255 バイトになります。
  ※データ数の詳細については EEL のユーザーズマニュアルを参照してください。

2.　仕様

本アプリケーションでは、スイッチ押下により LED0 または LED1 が 10 回点滅します。電源電圧低下時には、LED を点滅させるための情報をデータ・フラッシュ・メモリに保存します。再起動時に保存した情報を読み出し、中断した点滅処理の続きを行います。

まず、リセットが解除されると EEL を用いてデータ・フラッシュ・メモリから退避済みの LED 点滅状態のデータ（点滅対象 LED と LED 点滅回数）を読み出します。

次に、読み出したデータに応じて LED を 500ms 間隔で点滅させ、点滅が 10 回終了するとスイッチ入力待ちとなります。

LED が点滅していない状態でスイッチを押下すると、直前に点滅をしていなかった方の LED が点滅を開始します。また、LED が点滅している最中にはスイッチ入力は無効となります。

電源電圧の低下は LVD 機能で検出します。電源電圧の低下を検知すると、EEL を用いて LED 点滅状態のデータ（点滅対象 LED と点滅回数）をデータ・フラッシュ・メモリに退避し、データの退避完了を示す LED3 を点灯させて STOP モードに入ります。

また、EEL 関数でデータ・フラッシュ・メモリにアクセスする際にエラーが発生すると、LED0、LED1 を点灯させて STOP モードに入ります。

退避するデータの構造を図 2.1に示します。1 バイトのユーザ・データ上位 4 ビットは点滅対象 LED を示すデータになっており、下位 4 ビットは LED の点滅回数を示すデータになっています。図 2.1の例では LED1 が 5 回点滅を残しているデータであることを示しています。

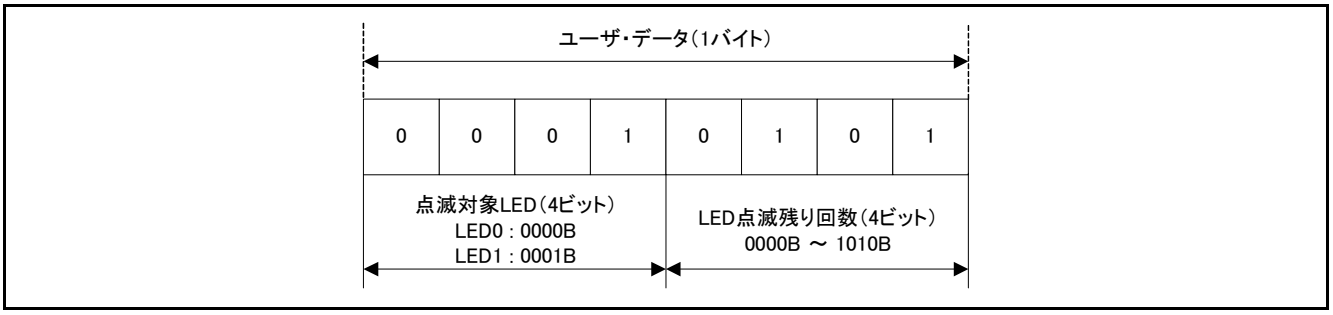


図2.1　格納データ

表 2.1に使用する周辺機能と用途を、図 2.2にアプリケーション全体像を、図 2.3に動作概要を示します。

表2.1　使用する周辺機能と用途

周辺機能	用途
LVD	電源電圧(V_{DD})を監視
外部割り込み（INTP0）	動作切り替えスイッチ入力
P05	LED 点灯制御(LED0)
P45	LED 点灯制御(LED1)
P41	LED 点灯制御(LED3)
タイマ・アレイ・ユニット(以降 TAU)0 チャンネル 0	スイッチのチャタリング回避のウェイト時間の生成（10ms）
TAU0 チャンネル 1	LED 点滅時間間隔の生成（500ms）

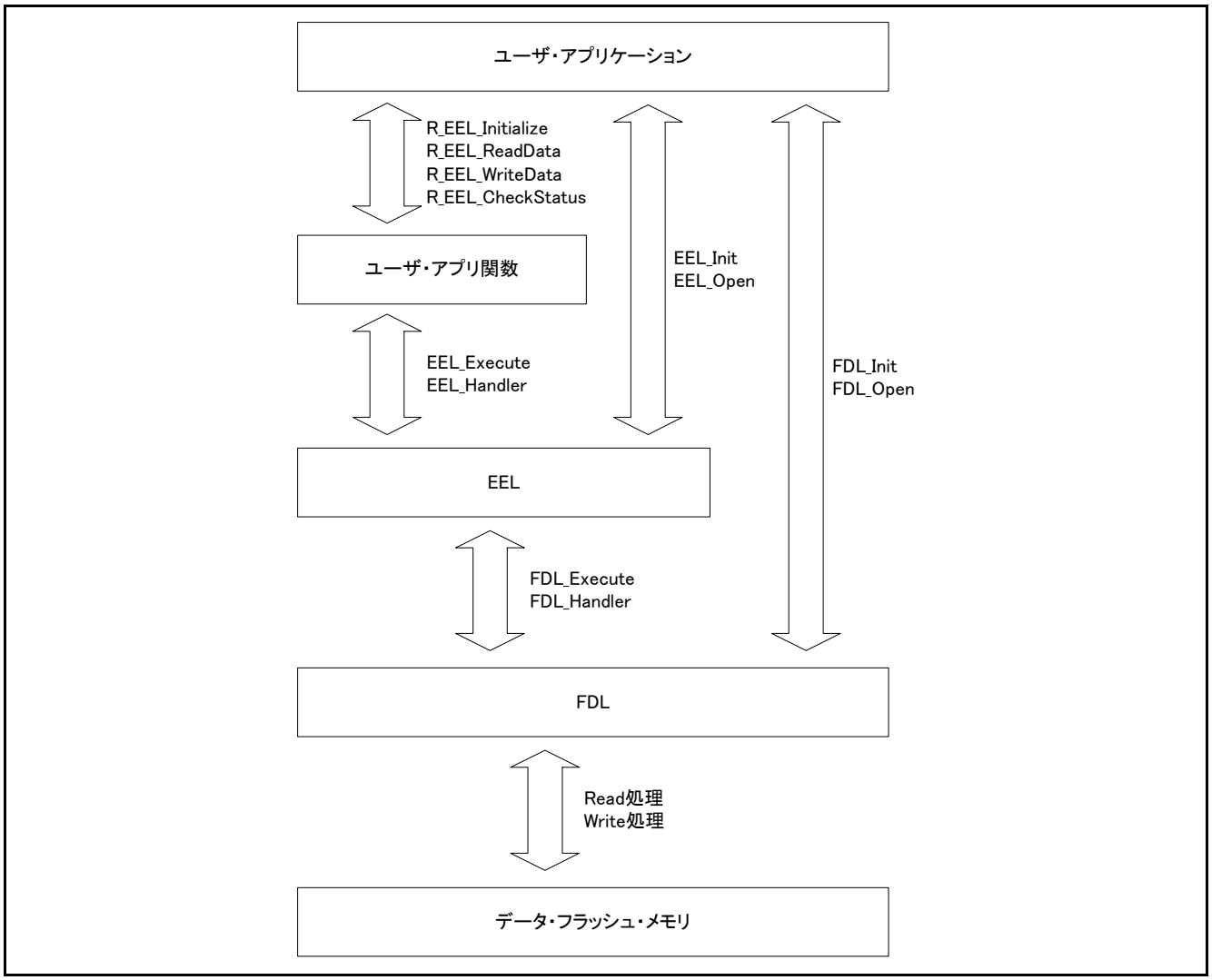


図2.2　アプリケーション全体像

ユーザ・アプリケーションからデータ・フラッシュ・メモリにアクセスするためには FDL/EEL の Init/Open を行うことで、データ・フラッシュ・メモリへのアクセスを許可状態にしたり、FDL/EEL で使用するリソースの確保をしたりする必要があります。Init/Open 処理の後、EEL の Startup を実行することでデータ・フラッシュ・メモリへの読み書きが可能になります。

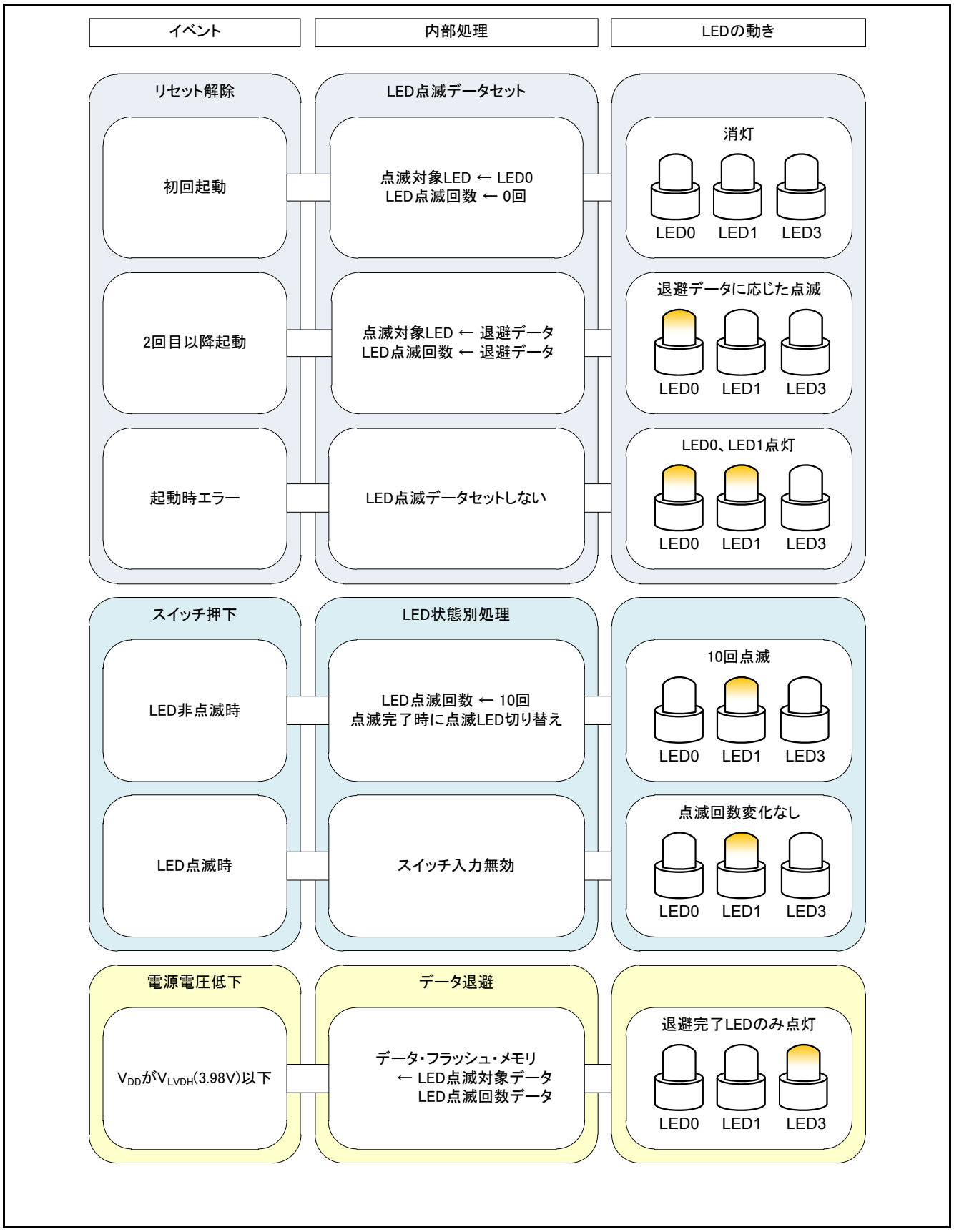

イベント
内部処理
LEDの動き
リセット解除
初回起動
2回目以降起動
起動時エラー
LED点滅データセット
点滅対象LED ← LED0
LED点滅回数 ← 0回
点滅対象LED ← 退避データ
LED点滅回数 ← 退避データ
LED点滅データセットしない
消灯
LED0 LED1 LED3
退避データに応じた点滅
LED0 LED1 LED3
LED0、LED1点灯
LED0 LED1 LED3
スイッチ押下
LED非点滅時
LED点滅時
LED状態別処理
LED点滅回数 ← 10回
点滅完了時に点滅LED切り替え
スイッチ入力無効
10回点滅
LED0 LED1 LED3
点滅回数変化なし
LED0 LED1 LED3
電源電圧低下
VDDがVLVDH(3.98V)以下
データ退避
データ・フラッシュ・メモリ
← LED点滅対象データ
LED点滅回数データ
退避完了LEDのみ点灯
LED0 LED1 LED3

図2.3　動作概要

2.1　EEL 書き込み時間の短縮

ユーザ・アプリケーションから EEL を使用してデータ・フラッシュ・メモリにアクセスするためには、データ・フラッシュ・メモリへのアクセスを許可状態にしたり、FDL/EEL で使用するリソースの確保をしたりする必要があります。そのため、EEL ではいくつかのライブラリ関数を呼ぶことで前述処理を実現しています。必要な処理は以下となります。

- FDL_Init 関数：FDL で使用する RAM の初期化
- FDL_Open 関数：データ・フラッシュ・メモリへのアクセス許可
- EEL_Init 関数：EEL で使用する RAM の初期化
- EEL_Open 関数：データ・フラッシュ・メモリを制御可能な状態に変更
- EEL_Execute 関数（STARTUP コマンド）：EEPROM エミュレーション実行可能な状態に変更

しかし、電源電圧低下時に上記の準備処理を行うと、データ退避中に電源断になる可能性があります。そのため、本アプリケーションノートではデータ退避にかかる時間を短縮するため、EEL の処理を準備処理と退避処理の 2 つに分割して行います。

処理を一括で行った場合のデータ退避処理を図 2.4に、処理を分割して行った場合のデータ退避処理を図 2.5に示します。データ退避時間は、一括処理で約 991[μs]、分割処理で約 683[μs]です。

備考．本アプリケーションノート記載の計測値は、統合開発環境 CubeSuite+上で EEL RL78 Pack02 Ver.1.01 を動作させたときの実測値です。

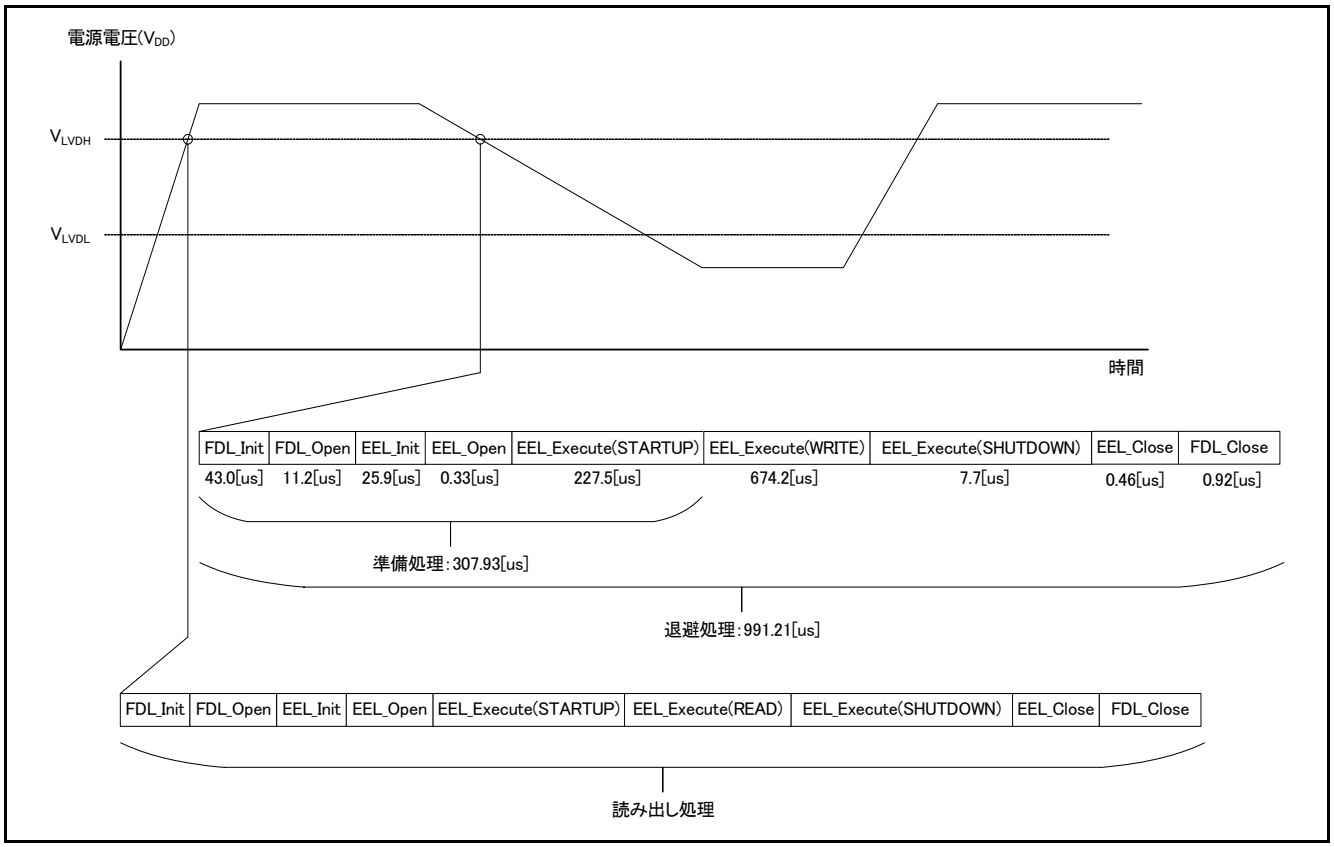


図2.4　データ退避処理（一括）

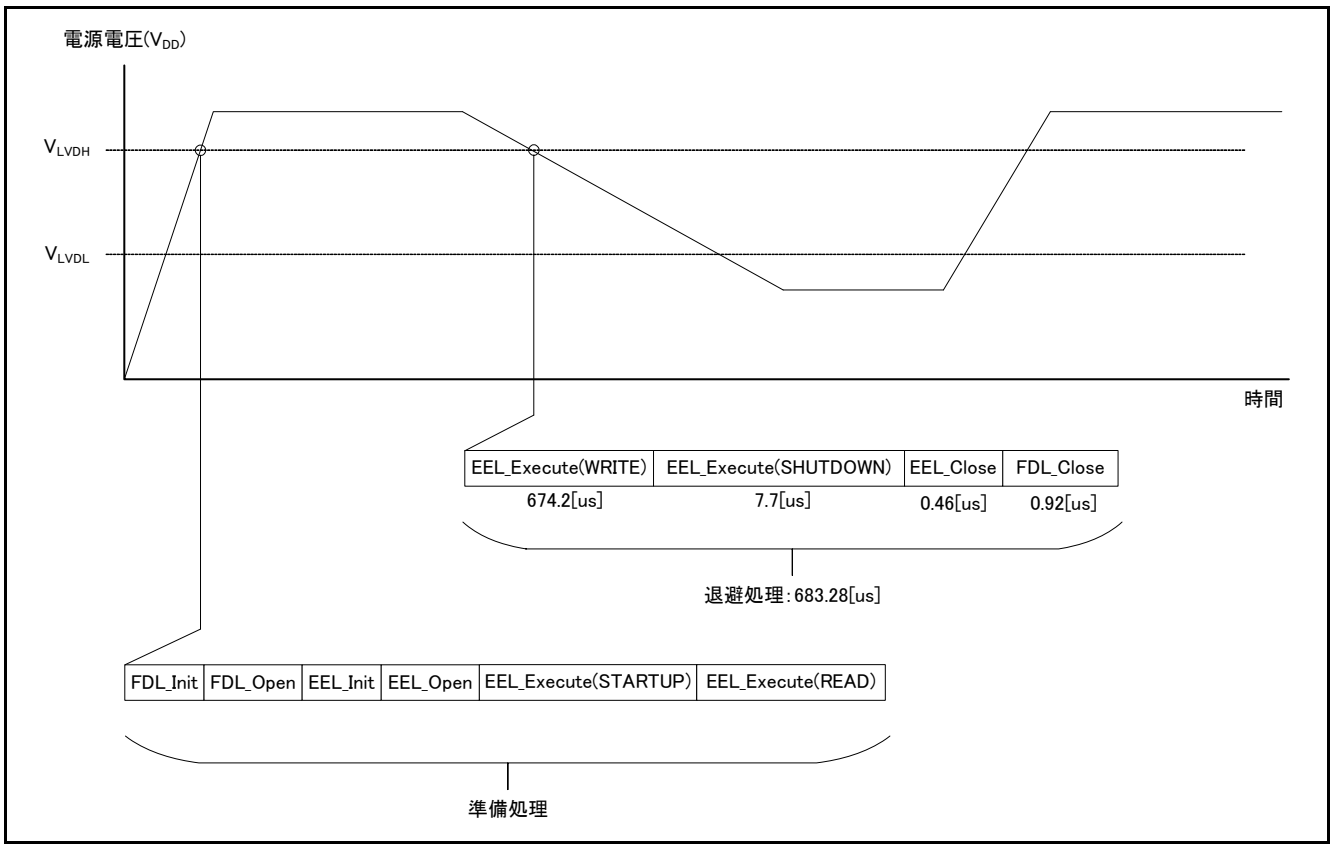


図2.5　データ退避処理（分割）

2.2 EEL アーキテクチャ

EEL の動作原理について説明します。EEL はデータをデータ領域で、データ ID を参照領域で管理しています。これら領域は同一ブロック内に構成され、ブロック毎に管理されています。使用しているブロックの未使用領域がなくなると、次のブロックを使用します。EEL でのデータ・フラッシュ・メモリの使い方について、本章で説明します。

2.2.1 EEL プール

EEL プールはユーザによって定義される EEL がアクセス可能なデータ・フラッシュ領域です。ユーザ・プログラムからのデータ・フラッシュのアクセスは、EEL 経由での EEL プールへのアクセスのみ許可されます。

対象デバイスに搭載されているデータ・フラッシュ・メモリのブロック数を EEL プールのブロック数に必ず設定してください。なお、設定方法につきましては、2.3

EEL ユーザ設定初期値をご参照ください。

EEL では EEL プールを 1024 バイトのブロックに分割します。各ブロックには状態があり、これがブロックの現在の使用状態を示しています。

表 2.2 EEL プール内の各ブロックの状態

状態	内容
有効	1 つの EEL ブロックが有効となり、定義済みのデータを格納します。有効ブロックは EEL プールに割り当てられたデータ・フラッシュ・ブロック群を循環します。
無効	無効ブロックにはデータは格納されません。EEL ブロックは EEL によって無効とされるか、消去ブロックの場合は無効となります。

図 2.6に 4KB のデータ・フラッシュ・メモリを有するデバイスのEEL プール構成を示します。

有効ブロック（例ではブロック 1）に書き込み可能領域がなくなり追加データの格納ができなくなったとき（write コマンドの失敗）には、新規の有効ブロックが循環的に選定され、その時点で有効なデータ群が新規の有効ブロックにコピーされます。このプロセスは「リフレッシュ」と呼ばれます。
EEL_CMD_REFRESH コマンド実行後に元の有効ブロックは無効となり、1 つの有効ブロックのみ存在します。

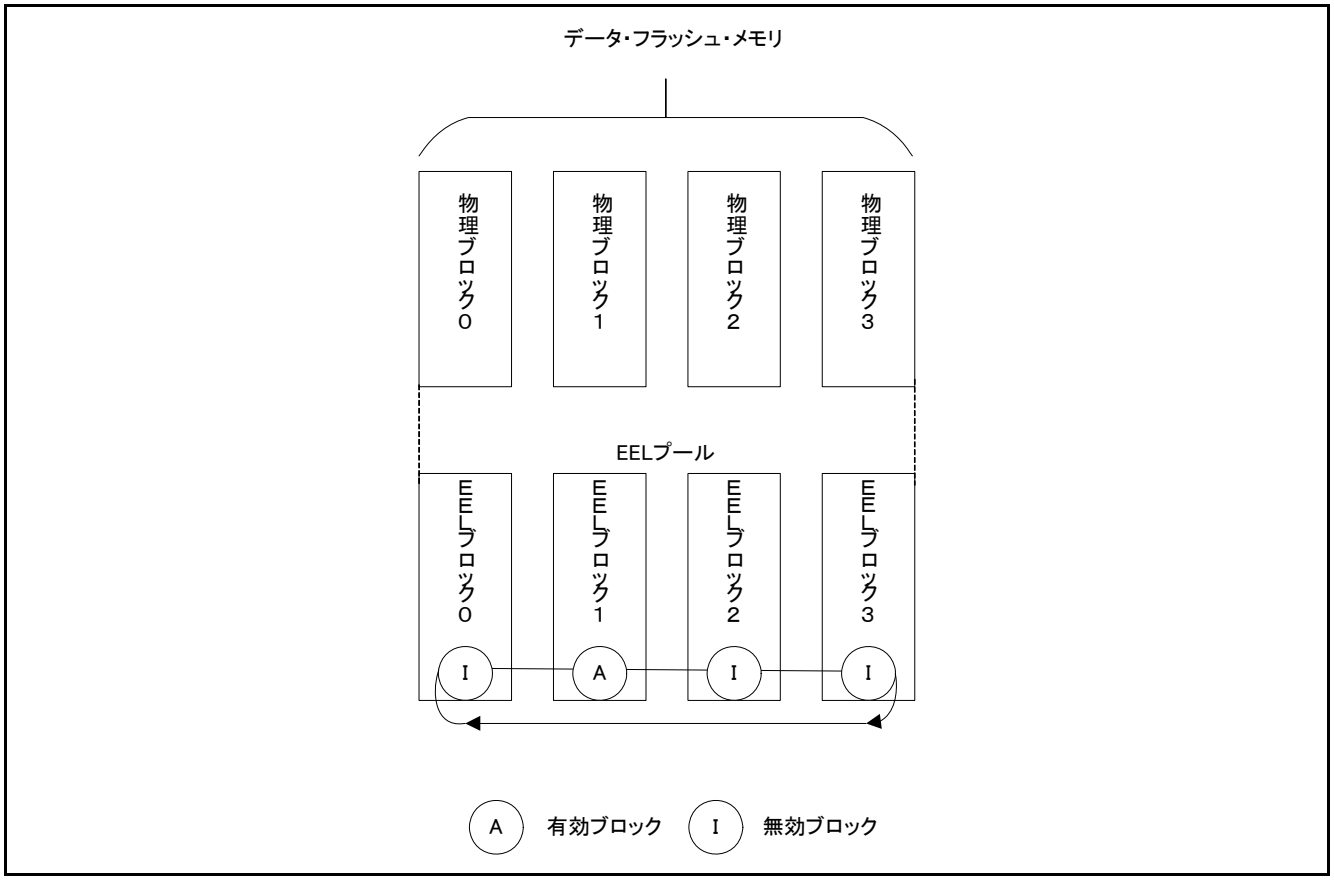


図2.6 EEL プール構成

図 2.7にEEL ブロックのライフサイクルを示します。EEL ブロックは有効状態と無効状態の間を行き来します。

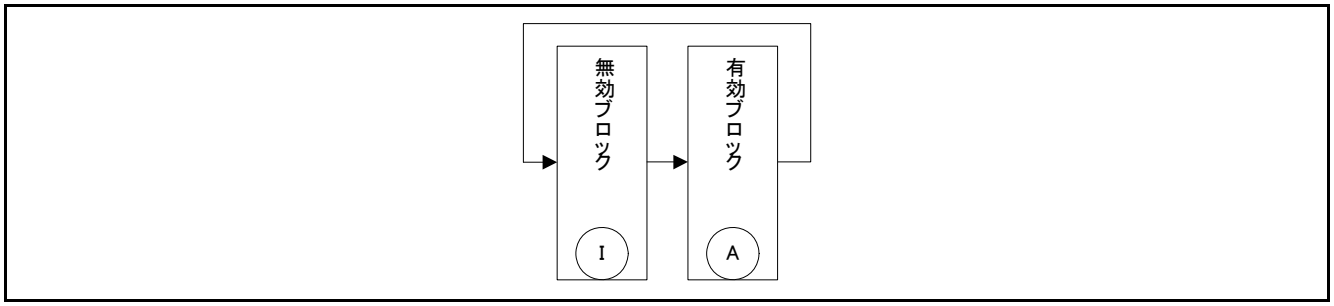


図2.7 EEL ブロックのライフサイクル

2.2.2　EEL ブロック

EEL が使用する EEL ブロック構造を図 2.8に示します。EEL ブロックは、ブロック・ヘッダ、参照領域、データ領域の 3 つの利用領域からなります。

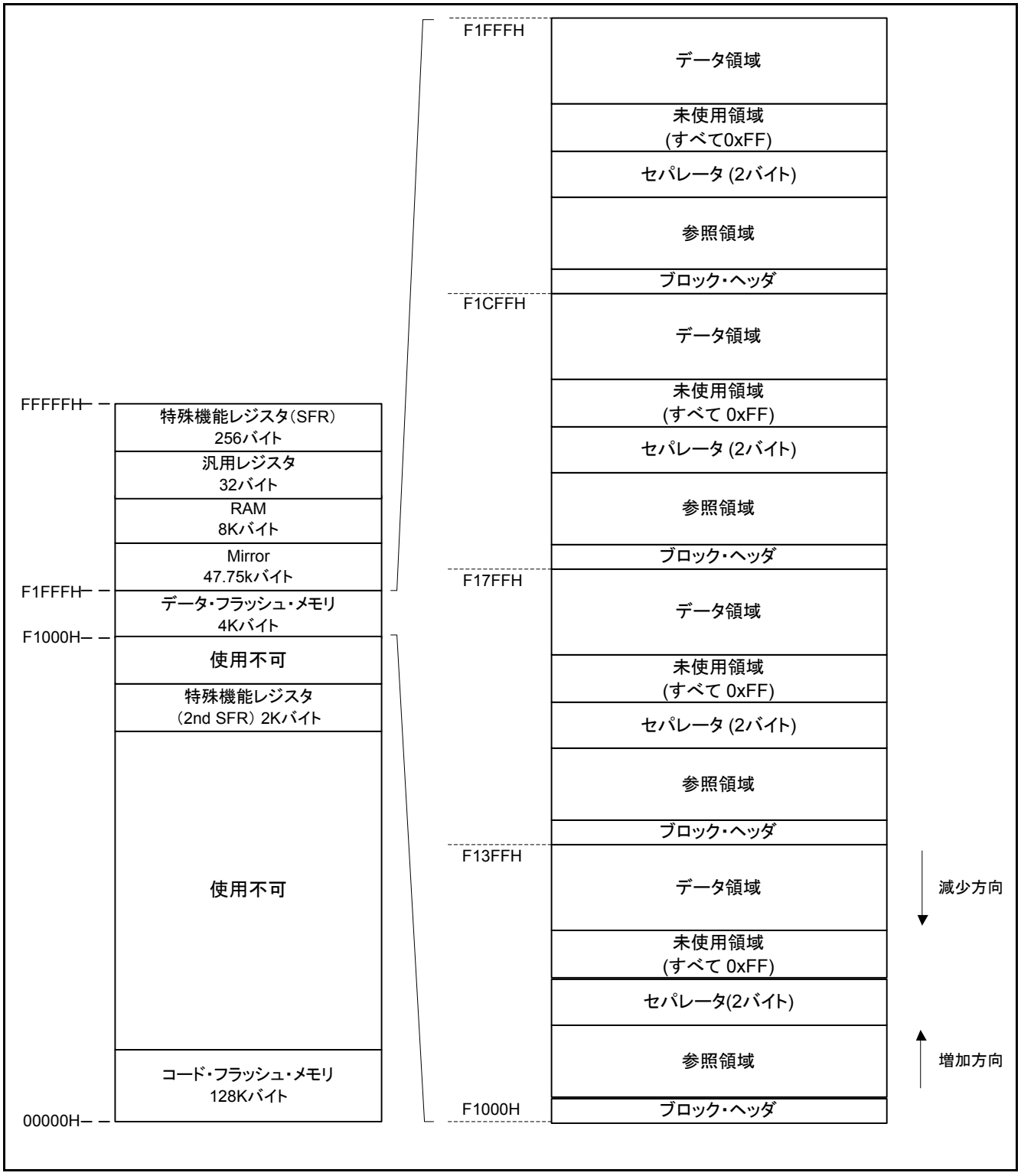


図2.8 EEL ブロックの構成(RL78/L13(R5F10WMG)の例)

表 2.3 EEL ブロックの構成一覧

名称	説明
ブロック・ヘッダ	EEL ブロック内のブロック管理に必要なブロック状態の情報が格納されています。8 バイトの固定サイズです。
参照領域	データの管理に必要な参照データが格納されています。データが書き込まれると、アドレスの増加方向に拡大します。
データ領域	ユーザ・データが格納されています。データが書き込まれると、アドレスの減少方向に拡大します。

参照領域とデータ領域の間には、未使用領域があります。データが更新される（データの書き込みが行われる）たびに、未使用領域は減少します。しかし、参照領域とデータ領域の間には、領域の分離とブロック管理のために最低でも 2 バイトの未使用領域が必要となります。これは、図 2.8ではセパレータとして示されています。

2.3　EEL ユーザ設定初期値

EEL の設定初期値として、次に示す項目を必ず設定する必要があります。また、EEL を実行する前に、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。外部クロックを使用時も、高速オンチップ・オシレータは起動しておく必要があります。各設定の右側に記載している括弧書きは次ページの番号と関連しています。

各項目の設定は、本アプリケーションに合わせた設定を記載しております。

＜FDL ユーザ・インクルード・ファイル（fdl_descriptor.h）＞注1、2

```
#define    FDL_SYSTEM_FREQUENCY    24000000    : (1) 動作周波数
#define    FDL_WIDE_VOLTAGE_MODE               : (2) 電圧モード
#define    FDL_POOL_BLOCKS         0           : (3) FDL プール・サイズ
#define    EEL_POOL_BLOCKS         4           : (4) EEL プール・サイズ
```

＜EEL ユーザ・インクルード・ファイル（eel_descriptor.h）＞注1、2

```
#define    EEL_VER_NO              1           : (5) 格納データ数
```

＜EEL ユーザタイプ・インクルード・ファイル（eel_user_types.h）＞注1、2

```
typedef    eel_u08                 type_A;     : (6) データ・サイズ
```

＜EEL ユーザ・プログラム・ファイル（eel_descriptor.c）＞注1、2

```
__far const eel_u08 eel_descriptor[EEL_VAR_NO+2] =        : (7) データ ID のデータ・サイズ
{
    (eel_u08)(EEL_VAR_NO),   /* variable count */     ¥
    (eel_u08)(sizeof(type_A)),    /* id = 1 */         ¥
    (eel_u08) (0x00),            /* zero terminator */ ¥
};
```

注 1. 使用しているマクロは EEL 共通です。数値以外は変更しないでください。

注 2. EEL ブロックの初期化後（EEL_CMD_FORMAT コマンド実行後）は各値を変更しないでください。変更する場合は EEL ブロックの再初期化（EEL_CMD_FORMAT コマンド実行）を行ってください。

(1) CPU の動作周波数

RL78 マイクロコントローラで使用されている CPU の動作周波数を設定します。注 1
設定値は以下の計算式により FDL_Init 関数の周波数パラメータへ設定されます。
本アプリケーションノートでは、CPU の動作周波数は 24MHz なので、24 に設定します。

注 1. 本設定はデータ・フラッシュ・メモリの制御に必要な値になります。本設定により、RL78 マイクロコントローラの 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) 電圧モード

データ・フラッシュ・メモリの電圧モードを設定します。注 2
FDL_WIDE_VOLTAGE_MODE が定義されていない場合　　：フルスピード・モード
FDL_WIDE_VOLTAGE_MODE が定義されている場合　　：ワイド・ボルテージ・モード

本アプリケーションノートでは、フルスピード・モードで動作させるため、
FDL_WIDE_VOLTAGE_MODE を定義しません。

注 2. 電圧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ください。

(3) FDL プール・サイズ注 3

0 を設定してください。

注 3. ユーザによって定義される FDL がアクセス可能なデータ・フラッシュ領域を FDL プールといいます。

(4) EEL プール・サイズ注 4

必ず対象デバイスに搭載されているデータ・フラッシュ・メモリのブロック数を EEL プールのブロック数に設定してください。

注 4. 3(3 ブロック)以上の値を設定してください。（推奨）

(5) 格納データ数

EEPROM エミュレーションで使用するデータ数を設定します。設定できる値は 1～64 の範囲です。
本アプリケーションノートでは、1 種類のデータを扱うため、格納データ数は 1 となります。

(6) データ・サイズ登録

EEL ディスクリプタ・テーブルにデータ ID ごとのデータ・サイズを登録します。

type_A、type_B、type_C、type_D、type_E、type_F、type_X、type_Z の 8 サイズが標準で定義されており、扱うユーザ・データのサイズによってサイズを変更する必要があります。

本アプリケーションノートでは 1 種類の 1 バイトデータ（LED 点滅状態）を扱うため、データ ID1 に 1 バイトの type_A を使用します。

(7) データ ID のデータのサイズ

各データ ID のデータのサイズを規定するテーブルです。これを EEL ディスクリプタ・テーブルといいます。EEL では、プログラム動作中に識別子を追加のみすることができます。書き込みを行うデータは(6)のように EEL ディスクリプタ・テーブルに事前に登録する必要があります。

EEL ディスクリプタ・テーブル

__far const eel_u08 eel_descriptor[格納データ数(1) + 2]

EEL_VAR_NO
データ ID1 のバイト・サイズ(type_A)
0x00

・EEL_VAR_NO

ユーザが指定する EEL で使用するデータの数です。

・データ IDx のバイト・サイズ

ユーザが指定する各ユーザ・データのバイト・サイズです。

・終端領域(0x00)

終端情報として 0 を設定します。

2.4　格納ユーザ・データ数とユーザ・データの合計サイズ

EEPROM エミュレーションで使用できるユーザ・データの合計サイズには制限があります。リフレッシュ処理を考慮すると、全ユーザ・データと 1 データ以上の未使用領域が 1 ブロックに収まる必要があります。

また、使用できる格納データ数は実際に格納するユーザ・データのサイズによって変わります。

以下に実際にユーザ・データの書き込みで使用できるサイズ、及び 1 ブロックあたりの書き込み可能回数の計算方法を示します。

【ユーザ・データの書き込みに使用できる 1 ブロックの最大使用可能サイズ】

データ・フラッシュ・メモリの 1 ブロックのサイズ	: 1024 バイト
EEPROM エミュレーションでブロックの管理に必要なサイズ	: 8 バイト
終端用の情報として必ず必要な空き容量（セパレータ）	: 2 バイト

1 ブロックの最大使用可能サイズ = 1024 バイト – 8 バイト – 2 バイト = 1014 バイト

【ユーザ・データごとの書き込みサイズの計算方法】

書き込まれる個々のユーザ・データのサイズ = データ・サイズ + 参照エントリ・サイズ（2 バイト）

本アプリケーションノートでは書き込むデータのサイズが 1 バイトなので、ユーザ・データのサイズは 3 バイトとなります。

【1 ブロックあたりの書き込み可能回数】

1 ブロックの最大使用可能サイズが 1014 バイトに対してユーザ・データのサイズが 3 バイトなので

書き込み可能回数 = 1014 / 3 = 338 回

となります。

本アプリケーションノートでは、338 回の書き込み毎にリフレッシュ処理を行う必要があります。なお、リフレッシュは準備処理（2.1

EEL 書き込み時間の短縮を参照）内で行います。リフレッシュ処理時は、準備処理時間が通常時に比べて 6.74[ms]長くなります。

2.5 EEL 使用時の注意事項

EEL を使用する上での注意事項を以下に示します。

- EEL 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。
- ウォッチドッグ・タイマは EEL 実行中も停止しません。
- EEL は多重処理に対応していないため、EEL 関数を割り込み処理内で実行しないでください。
- EEPROM エミュレーション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。また、外部クロックを使用時も、高速オンチップ・オシレータは起動しておく必要があります。
- EEL 関数および FDL 関数で使用するデータ・バッファ（引数）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。
- データ・フラッシュ・メモリを EEPROM エミュレーションで使用するためには初回起動時に EEL_CMD_FORMAT コマンドを実行し、データ・フラッシュ・メモリを EEL ブロックとして使用できるように初期化を行う必要があります。
- EEL を使用するためには、データ・フラッシュ・メモリを 3 ブロック以上使用することを推奨します。
- EEL は多重実行に対応していません。OS 上で EEL 関数を実行する場合は、複数のタスクから EEL 関数を実行しないでください。
- RL78 マイクロコントローラの CPU の動作周波数と初期化関数（FDL_Init 関数）で設定する CPU の動作周波数値について、以下の点に注意してください。
  - RL78 マイクロコントローラの CPU の動作周波数として 4MHz 未満の周波数を使用する場合は、1MHz、2MHz、3MHz のみを使用することができます。
    （1.5MHz のように整数値にならない周波数は使用できません）
  - RL78 マイクロコントローラの CPU の動作周波数として 4MHz 以上[注]の周波数を使用する場合は RL78 マイクロコントローラに任意の周波数を使用することができます。
  - 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。

注 最大周波数については、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。

3.　動作確認条件

本アプリケーションノートのサンプルコードは、下記の条件で動作を確認しています。

表3.1　動作確認条件

項目	内容
使用マイコン	RL78/L13(R5F10WMGA)
動作周波数	・高速内蔵発振クロック(f_{HOCO})：24MHz(標準) ・CPU/周辺ハードウェア・クロック(f_{CLK})：24MHz
動作電圧	5.0V(4.1V～5.5V で動作可能) LVD 動作：割り込み＆リセット・モード V_{LVDH}(立ち上がり 4.06V/立ち下がり 3.98V) V_{LVDL}(立ち下がり 2.75V)
CubeSuite+版開発環境 ・統合開発環境 ・C コンパイラ ・EEL	 ルネサス エレクトロニクス製 CubeSuite+ V2.01.00 ルネサス エレクトロニクス製 CA78K0R V1.70 EELRL78 Pack02 Ver1.01 注
IAR 版開発環境 ・統合開発環境 ・C コンパイラ ・FDL ・EEL	 ルネサス エレクトロニクス製 e2studio V2.2.0.13 IAR システムズ株式会社製 IAR C/C++ Compiler V1.30.5.50715 for RL78 FDLRL78 T02E V1.10 注 EELRL78 T02E V1.00 注
使用ボード	Renesas Starter Kit for RL78/L13　(R0K5010WMC000BR)

注　最新バージョンをご使用／評価の上、ご使用ください。

4. 関連アプリケーションノート

本アプリケーションノートに関連するアプリケーションノートを以下に示します。併せて参照してください。

RL78 ファミリ EEPROM エミュレーション・ライブラリ Pack02（R01US0068J）ユーザーズマニュアル

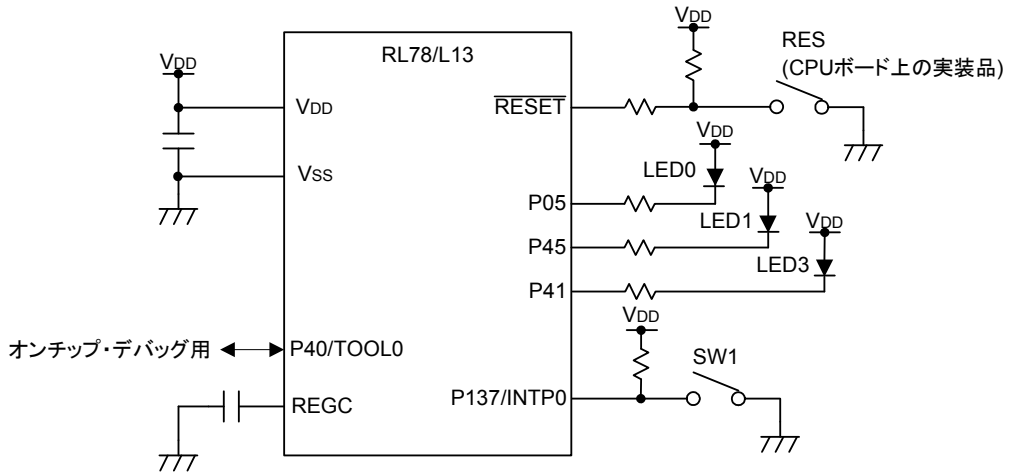
Data Flash Access Library (Type T02 (Tiny), European Release) (R01US0061ED0100) アプリケーションノート

EEPROM Emulation Library (Type T02 (Tiny), European Release) (R01US0070ED0102) アプリケーションノート

5.　ハードウェア説明

5.1　　ハードウェア構成例

図 5.1に接続例を示します。



注1. この回路イメージは接続の概要を示すために簡略化しています。実際に回路を作成される場合は、端子処理などを適切に行い、電気的特性を満たすように設計してください。(入力専用ポートは個別に抵抗を介してVDDまたはVSSに接続してください)
注2. VDDはLVDに設定したリセット解除電圧(VLVDH)以上にしてください。

図5.1　接続例

5.2　　使用端子一覧

表 5.1に使用端子と機能を示します。

表5.1　使用端子と機能

端子名	入出力	内容
P05	出力	LED 点灯（LED0）制御ポート
P45	出力	LED 点灯（LED1）制御ポート
P41	出力	LED 点灯（LED3）制御ポート
P137/INTP0	入力	スイッチ入力（SW1）ポート

6. ソフトウェア説明

6.1 動作概要

本アプリケーションでは、スイッチ押下により LED0 または LED1 が 10 回点滅します。電源電圧低下時には、LED を点滅させるための情報をデータ・フラッシュ・メモリに保存します。再起動時に保存した情報を読み出し、中断した点滅処理の続きを行います。

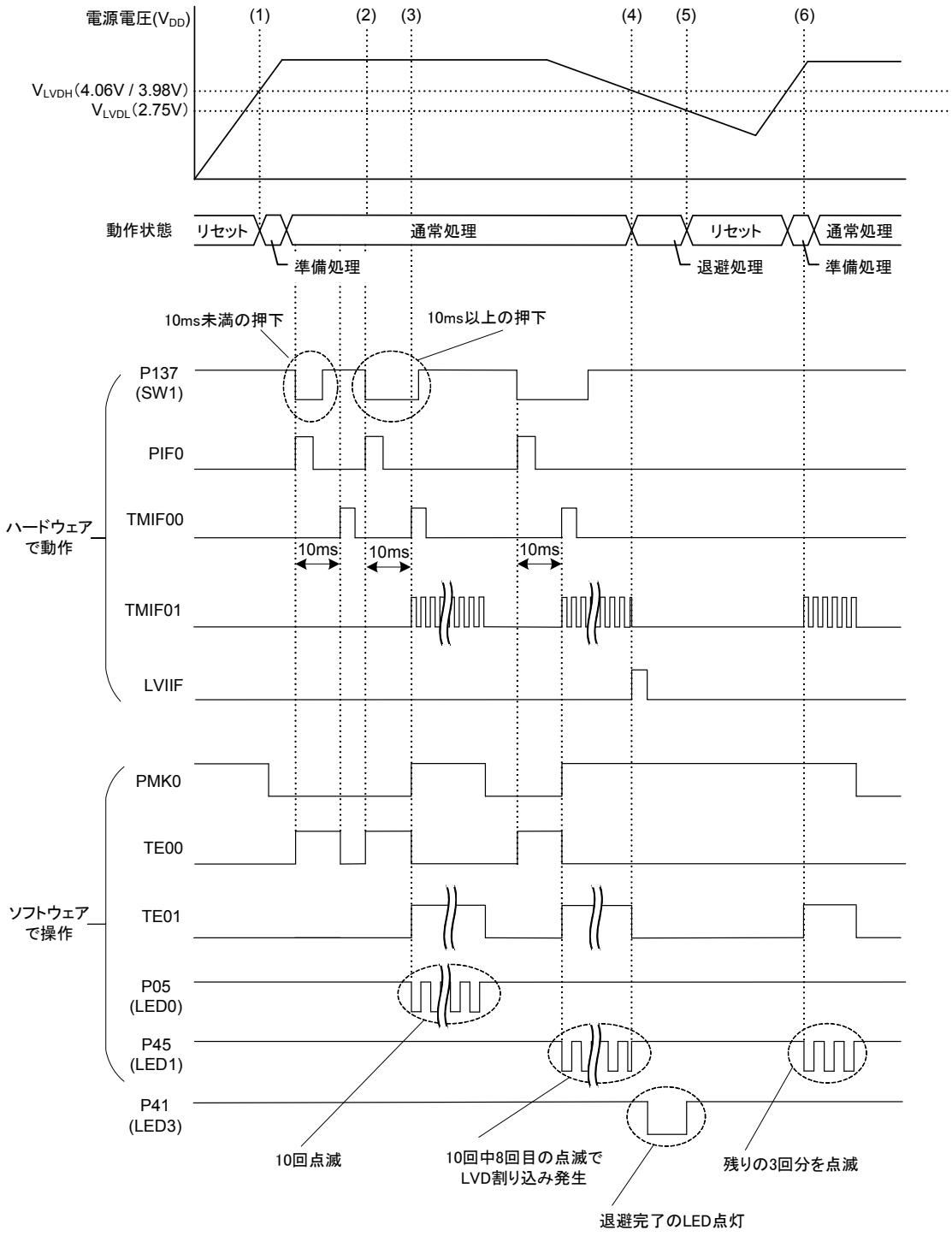
まず、リセットが解除されると EEL を用いてデータ・フラッシュ・メモリから退避済みの LED 点滅状態のデータ（点滅対象 LED と LED 点滅回数）を読み出します。

次に、読み出したデータに応じて LED を 500ms 間隔で点滅させ、点滅が 10 回終了するとスイッチ入力待ちとなります。LED が点滅していない状態でスイッチを押下すると、直前に点滅をしていなかった方の LED が点滅を開始します。また、LED が点滅している最中にはスイッチ入力は無効となります。

電源電圧の低下は LVD 機能で検出します。電源電圧の低下を検知すると、EEL を用いて LED 点滅状態のデータ（点滅対象 LED と点滅回数）をデータ・フラッシュ・メモリに退避し、データの退避完了を示す LED3 を点灯させて STOP モードに入ります。また、EEL 関数でデータ・フラッシュ・メモリにアクセスするときにエラーが発生すると、LED0、LED1 を点灯させて STOP モードに入ります。

1. 入出力ポートを設定します。
   ・LED 点灯制御（LED0、LED1、LED3）：P05、P45、P41 を出力ポートに設定
   （LED0、LED1、LED3 いずれも消灯状態）
   ・スイッチ入力：P137/INTP0 を INTP0 立ち下がりエッジ検出割り込みに設定（割り込み無効設定）
2. FDL/EEL で使用する RAM の初期化と準備処理をして、EEPROM エミュレーションを開始します。
   具体的には、以下の順にライブラリ関数をコールします。
   FDL_Init、FDL_Open、EEL_Init、EEL_Open、EEL_Execute(Startup)
3. LED 点灯状態(データ ID:1)を読み出して､データに応じて対象の LED を 500ms 間隔で点滅させます。
   ・読み出したデータは、上位 4 ビットが点滅対象 LED のデータ（0000B：LED0、0001B：LED1）
   下位 4 ビットが点滅回数のデータ（範囲：0000B～1010B）を示しています。
   ・データが存在しない場合は、点滅対象の LED を LED0 に、点滅回数を 0 に設定します。
   ・読み出したデータに応じた点滅を開始します。
   ・データの読み出しには EEL_Execute(Read)関数を用います。
4. データ・フラッシュ・メモリの空き容量を取得して書き込みができる領域を確保します。
   ・空き容量が 3 バイト未満（ユーザ・データのサイズより小さい）の場合、リフレッシュ処理を行うことで別のブロックに領域を確保し、最新データを移動させます。
   ・空き容量が 3 バイト以上の場合、リフレッシュ処理を実行しません。
5. スイッチが押下されると LED が 10 回点滅します。
   ・P137/INTP0 の立ち下がりエッジを検出して割り込み処理を行います。10ms のチャタリング検出を行い、スイッチ入力と判定した場合は LED の点滅を開始します。
   ・点滅対象となる LED はスイッチ押下の度に変更されます。
   ・一度スイッチを押下されてから点滅が終了するまで次のスイッチ押下を受け付けません。
6. LVD 割り込みが発生すると、LED の点滅残り回数と点滅対象の LED のデータをデータ・フラッシュ・メモリに退避し、退避完了の LED（LED3）を点灯した上で FDL/EEL を停止して STOP モードに入ります。具体的には、以下の順にライブラリ関数をコールしたあとに STOP 命令を実行します。
   EEL_Execute(Write)、EEL_Execute(Shutdown)、EEL_Close、FDL_Close
7. EEL でデータ・フラッシュ・メモリにアクセスする際にエラーが発生すると、FDL/EEL を停止して LED0、LED1 を点灯させて STOP モードに入ります。
   具体的には、以下の順にライブラリ関数をコールしたあとに STOP 命令を実行します。
   EEL_Execute(Shutdown)、EEL_Close、FDL_Close
8. リセットが発生すると、1 の処理に戻ります。

図 6.1にタイミング図を示します。



備考1. 準備処理：データ退避のためのEEL書き込み準備を行います。退避したデータを読み出してLEDを点滅開始させます。
ブロックの空き容量をチェックし、書き込み用領域が無ければリフレッシュを行います。
備考2. 通常処理：スイッチ押下待ちをします。スイッチが押下されると対象のLEDを10回点滅させ、再度スイッチ押下待ちに
なります。点滅するLEDはスイッチ押下の度に変わります。
備考3. 退避処理：LED点滅残り回数と点滅対象LEDのデータをデータ・フラッシュ・メモリに書き込み、LEDを消灯させ、
退避完了のLED3を点灯させます。 INTP0割り込みを無効にしてSTOPモードに入ります。

図6.1　タイミング図

(1) リセット解除

リセットが解除され、起動すると FDL/EEL で使用する RAM の初期化、LED 点滅データの読み出しをします。読み出したデータに応じて LED を点滅開始します。

(2) SW1 押下

チャタリング回避用のインターバル・タイマのカウントを開始します。

(3) SW1 押下検知

SW1 押下 10ms 後に、SW1 が押されていたら SW1 押下とみなし、500ms のインターバル・タイマを動作させて LED の点滅を開始します。

(4) 電源電圧低下検知

LED 点滅データ（LED 点滅残り回数、点滅対象 LED）をデータ・フラッシュ・メモリ（データ ID : 1）に書き込んで LED を消灯させます。また、退避完了の LED（LED3）を点灯した上で INTP0 割り込みを無効（SW1 操作を無効）にし、STOP モードに入ります。図 6.1の例では LED 点滅データは 03H（LED0 の点滅残回数が 3 回）となります。

(5) リセット発生

電源電圧が 2.75V（V_{LVDL} 立ち下がり）以下になると、LVD によるリセットが発生します。

(6) 退避データ処理

リセット解除時に退避に応じた LED が点滅します。図 6.1の例では LED0 が 3 回点滅します。

6.2　ファイル構成

表 6.1にサンプルコードで使用するファイルを示します。なお、統合開発環境で自動生成されるファイルは除きます。

表6.1 ファイル構成

ファイル名	概要	備考
r_eel_function.c	データ退避用関数のソースファイル	追加関数： R_EEL_Initialize R_EEL_CheckStatus R_EEL_ReadData R_EEL_CheckDataRange R_EEL_WriteData
r_eel_function.h	データ退避用関数のヘッダファイル	-
fdl_descriptor.c	FDL ディスクリプタソースファイル	各コンパイラ共通
fdl_descriptor.h	FDL ディスクリプタヘッダファイル	各コンパイラ共通
eel_descriptor.c	EEL ディスクリプタソースファイル	各コンパイラ共通
eel_descriptor.h	EEL ディスクリプタヘッダファイル	各コンパイラ共通
fdl.h 注 1	FDL のヘッダファイル	各コンパイラ共通
eel.h 注 1	EEL のヘッダファイル	各コンパイラ共通
eel_types.h	EEL 型定義ヘッダファイル	各コンパイラ共通
eel_user_types.h	EEL ユーザ型定義ヘッダファイル	各コンパイラ共通
fdl.lib 注 1	FDL	CubeSuite+版
fdl.r87 注 1	FDL	IAR 版
eel.lib 注 1	EEL	CubeSuite+版
eel.r87 注 1	EEL	IAR 版
r_eel.dr 注 2	リンク・ディレクティブ・ファイル	CubeSuite+版
r_eel.xcl 注 2	リンク・ディレクティブ・ファイル	IAR 版

注 1　別途追加する必要があるファイルです。詳細は表紙「コンパイラと対応する EEL について」を参照してください。

注 2　使用するデバイスによって内容に変更が必要な場合があります。

6.3　オプション・バイトの設定

表 6.2にオプション・バイト設定を示します。

表6.2　オプション・バイト設定

アドレス	設定値	内容
000C0H/010C0H	11101111B	ウォッチドッグ・タイマ動作停止 (リセット解除後、カウント停止)
000C1H/010C1H	01110010B	LVD 割り込み＆リセット・モード 検出電圧 V_{LVDH}：立ち上がり 4.06V/立ち下がり 3.98V V_{LVDL}：立ち下がり 2.75V
000C2H/010C2H	11100000B	高速内蔵発振 HS モード 24MHz
000C3H/010C3H	10000100B	オンチップ・デバッグ許可

6.4 定数一覧

表 6.3に定数を示します。

表6.3 定数

定数名	設定値	内容
DATA_ID	0x01	EEL のデータ ID
RET_OK	0x00	正常応答
RET_NG_DEVICE	0x01	デバイス異常
RET_NG_NODATA	0x02	格納データ無し
RET_NG_RANGE	0x03	データが有効範囲外
USER_DATA_SIZE	0x03	ユーザ・データ・サイズ
SHIFT_NUM	0x04	LED 点滅データ用ビットシフト数
STATUS_PENDING	0xFF	保留ステータス
BLINK_LED_MAX	0x01	LED 点滅番号最大値
BLINK_NUM_MAX	0x0A	LED 点滅回数最大値
BLINK_LED0	0x00	点滅対象 LED0
BLINK_LED1	0x01	点滅対象 LED1
LED0	P0.5	LED0 制御ポート（CubeSuite+版）
	P0_bit.no5	LED0 制御ポート（IAR 版）
LED1	P4.5	LED1 制御ポート（CubeSuite+版）
	P4_bit.no5	LED1 制御ポート（IAR 版）
LED3	P4.1	LED3 制御ポート（CubeSuite+版）
	P4_bit.no1	LED3 制御ポート（IAR 版）
LED_ON	0	LED 点灯レベル
LED_OFF	1	LED 消灯レベル
SW1	P13.7	SW1 制御ポート（CubeSuite+版）
	P13_bit.no7	SW1 制御ポート（IAR 版）
SW_ON	0	SW 押下レベル
SW_OFF	1	SW 非押下レベル
BLINK_LED_MASK	0xF0	LED 点滅データの点滅対象マスク
BLINK_NUM_MASK	0x0F	LED 点滅データの点滅回数マスク

6.5 変数一覧

表 6.4にグローバル変数を示します。

表6.4 グローバル変数

型	変数名	内容	使用関数
volatile uint8_t	g_blink_led	点滅対象 LED	main r_tau0_channel0_interrupt r_tau0_channel1_interrupt
volatile uint8_t	g_blink_num	点滅回数	main r_tau0_channel0_interrupt r_tau0_channel1_interrupt
volatile uint8_t	g_lvd_flag	電源電圧低下検出フラグ	main r_lvd_interrupt

6.6 関数一覧

表 6.5に関数を示します。

表6.5 関数

関数名	概要
R_Systeminit	周辺機能初期設定
R_PORT_Create	ポート初期設定
R_CGC_Create	CPU クロック初期設定
R_TAU0_Create	TAU0 初期設定
R_INTC_Create	INTP 初期設定
R_LVD_Create	LVD 初期設定
main	メイン処理
R_MAIN_UserInit	メイン初期化処理
R_EEL_Initialize	EEL 初期化処理
R_EEL_ReadData	EEL 読み出し
R_EEL_CheckDataRange	LED 点滅データ有効範囲チェック
R_EEL_CheckStatus	EEL 関数ステータスチェック
R_TAU0_Channel1_Start	TAU01 動作許可設定
r_tau0_channel1_interrupt	TAU01 割り込みハンドラ
R_TAU0_Channel1_Stop	TAU01 動作禁止設定
R_INTC0_Start	INTP0 動作許可設定
r_intc0_interrupt	INTP0 割り込みハンドラ
R_TAU0_Channel0_Start	TAU00 動作許可設定
r_tau0_channel0_interrupt	TAU00 割り込みハンドラ
R_EEL_WriteData	EEL 書き込み
R_TAU0_Channel0_Stop	TAU00 動作禁止設定
R_INTC0_Stop	INTP0 動作禁止設定
R_LVD_InterruptMode_Start	LVD 割り込み許可設定
r_lvd_interrupt	LVD 割り込みハンドラ
FDL_Init	FDL の初期化
FDL_Open	FDL の準備処理
FDL_Close	FDL の終了処理
EEL_Init	EEL の初期化
EEL_Open	EEL の準備処理
EEL_Close	EEL の終了処理
EEL_Execute	各コマンドによるデータ・フラッシュ操作の実行
EEL_Handler	実行中の EEL を制御
EEL_GetSpace	EEL ブロックの空き容量の確認処理

6.7　関数仕様

サンプルコードの関数仕様を示します。

各関数、共通して r_cg_macrodriver.h ヘッダをインクルードしています。

R_Systeminit

概　要	周辺機能初期設定
ヘッダ	なし
宣　言	void R_Systeminit(void)
説　明	本アプリケーションノートで使用する周辺機能の初期設定を行います。
引　数	なし
リターン値	なし

R_PORT_Create

概　要	ポート初期設定
ヘッダ	r_cg_port.h
宣　言	void R_PORT_Create(void)
説　明	ポート初期設定を行います。
引　数	なし
リターン値	なし

R_CGC_Create

概　要	CPU クロック初期設定
ヘッダ	r_cg_cgc.h
宣　言	void R_CGC_Create(void)
説　明	CPU クロック初期設定を行います。
引　数	なし
リターン値	なし

R_TAU0_Create

概　要	TAU0 初期設定
ヘッダ	r_cg_timer.h
宣　言	void R_TAU0_Create(void)
説　明	TAU00、TAU01 をインターバル・タイマとして使用するための初期設定を行います。
引　数	なし
リターン値	なし

R_INTC_Create

概　要	INTP 初期設定
ヘッダ	r_cg_intc.h
宣　言	void R_INTC_Create(void)
説　明	INTP の初期設定をします。
引　数	なし
リターン値	なし

R_LVD_Create

概　要	LVD 初期設定
ヘッダ	r_cg_lvd.h
宣　言	void R_LVD_Create(void)
説　明	LVD の初期設定をします。
引　数	なし
リターン値	なし

main

概　要	メイン処理
ヘッダ	r_cg_tau.h r_cg_intc.h r_eel_function.h r_cg_userdefine.h
宣　言	void main(void)
説　明	メイン処理を行います。
引　数	なし
リターン値	なし

R_MAIN_UserInit

概　要	メイン初期化処理
ヘッダ	r_lvd.h r_eel_function.h
宣　言	void R_MAIN_UserInit(void)
説　明	メイン関数内の初期化を行います。
引　数	なし
リターン値	なし

R_EEL_Initialize

概　要	EEL 初期化処理
ヘッダ	r_eel_function.h r_cg_userdefine.h
宣　言	uint8_t R_EEL_Initialize(void)
説　明	EEL の初期化処理を行います。
引　数	なし
リターン値	・正常応答　　：RET_OK ・デバイス異常　　：RET_NG_DEVICE

R_EEL_ReadData

概　要	EEL 読み出し	
ヘッダ	r_eel_function.h r_cg_userdefine.h	
宣　言	uint8_t R_EEL_ReadData(uint8_t id, uint8_t* pdata)	
説　明	データ・フラッシュ・メモリからデータを読み出します。	
引　数	uint8_t id	読み出すデータ ID
	uint8_t* pdata	読み出したデータを格納するバッファのポインタ
リターン値	・正常応答	: RET_OK
	・デバイス異常	: RET_NG_DEVICE
	・データなし	: RET_NG_NODATA

R_EEL_CheckDataRange

概　要	LED 点滅データ有効範囲チェック	
ヘッダ	r_eel_function.h	
宣　言	uint8_t R_EEL_CheckDataRange(uint8_t data)	
説　明	LED 点滅データが有効範囲内かどうかをチェックします。	
引　数	uint8_t data	チェック対象のデータ
リターン値	・有効範囲内	: RET_OK
	・有効範囲外	: RET_NG_RANGE

R_EEL_CheckStatus

概　要	EEL 関数ステータスチェック	
ヘッダ	r_eel_function.h r_cg_userdefine.h	
宣　言	uint8_t R_EEL_CheckStatus(eel_request_t* request_pstr)	
説　明	EEL 関数の実行ステータスをチェックします。 EEL_Execute 関数を実行した直後に呼んでください。	
引　数	eel_request_t* request_pstr	EEL_Execute 関数実行時の引数
リターン値	・正常応答	: RET_OK
	・デバイス異常	: RET_NG_DEVICE
	・データなし	: RET_NG_NODATA

R_TAU0_Channel1_Start

概　要	TAU01 動作許可設定
ヘッダ	r_cg_timer.h
宣　言	void R_TAU0_Channel1_Start(void)
説　明	TAU01 のカウントを開始します。
引　数	なし
リターン値	なし

r_tau0_channel1_interrupt

概　要	TAU01 割り込みハンドラ
ヘッダ	r_cg_timer.h r_eel_function.h
宣　言	__interrupt static void r_tau0_channel1_interrupt(void)
説　明	LED の点灯／消灯を切り替え、点滅回数をデクリメントします。 点滅が完了したら点滅対象の LED を切り替えます。
引　数	なし
リターン値	なし

R_TAU0_Channel1_Stop

概　要	TAU01 動作禁止設定
ヘッダ	r_cg_timer.h
宣　言	void R_TAU0_Channel1_Stop(void)
説　明	TAU01 のカウントを停止します。
引　数	なし
リターン値	なし

R_INTC0_Start

概　要	INTP0 動作許可設定
ヘッダ	r_cg_intp.h
宣　言	void R_INTC0_Start(void)
説　明	INTP0 割り込みを有効に設定します。
引　数	なし
リターン値	なし

r_intc0_interrupt

概　要	INTP0 割り込みハンドラ
ヘッダ	r_cg_intp.h r_cg_timer.h
宣　言	__interrupt static void r_intc0_interrupt(void)
説　明	TAU00 の動作を開始します。
引　数	なし
リターン値	なし

R_TAU0_Channel0_Start

概　要	TAU00 動作許可設定
ヘッダ	r_cg_timer.h
宣　言	void R_TAU0_Channel0_Start(void)
説　明	TAU00 のカウントを開始します。
引　数	なし
リターン値	なし

r_tau0_channel0_interrupt

概　要	TAU00 割り込みハンドラ
ヘッダ	r_cg_timer.h r_eel_function.h
宣　言	__interrupt static void r_tau0_channel0_interrupt(void)
説　明	SW1 の状態をチェックして LED 点滅を開始します。
引　数	なし
リターン値	なし

R_EEL_WriteData

概　要	EEL 書き込み	
ヘッダ	r_eel_function.h r_cg_userdefine.h	
宣　言	uint8_t R_EEL_WriteData(uint8_t id, uint8_t* pdata)	
説　明	データ・フラッシュ・メモリにデータを書き込みます。	
引　数	uint8_t id	書き込むデータ ID
	uint8_t* pdata	書き込むデータのバッファのポインタ
リターン値	・正常応答	: RET_OK
	・デバイス異常	: RET_NG_DEVICE

R_TAU0_Channel0_Stop

概　要	TAU00 動作禁止設定
ヘッダ	r_cg_timer.h
宣　言	void R_TAU0_Channel0_Stop(void)
説　明	TAU00 のカウントを停止します。
引　数	なし
リターン値	なし

R_INTC0_Stop

概　要	INTP0 動作禁止設定
ヘッダ	r_cg_intp.h
宣　言	void R_INTC0_Stop(void)
説　明	INTP0 割り込みを無効に設定します。
引　数	なし
リターン値	なし

R_LVD_InterruptMode_Start

概　要	LVD 割り込み許可設定
ヘッダ	r_cg_lvd.h
宣　言	void R_LVD_InterruptMode_Start(void)
説　明	LVD の割り込みを許可します。
引　数	なし
リターン値	なし

r_lvd_interrupt

概　要	LVD 割り込みハンドラ
ヘッダ	r_cg_lvd.h r_eel_function.h
宣　言	__interrupt static void r_lvd_interrupt(void)
説　明	電源電圧低下のフラグをセットします。
引　数	なし
リターン値	なし

FDL_Init

概　要	FDL の初期化	
ヘッダ	fdl.h	
宣　言	fdl_status_t __far FDL_Init(const __far descriptor_t* descriptor_pstr)	
説　明	FDL の初期化処理を行います。 ※FDL のライブラリ関数です。	
引　数	const __far descriptor_t* descriptor_pstr	ディスクリプタ・テーブルへのポインタ
リターン値	正常終了	: FDL_OK
	初期化エラー	: FDL_ERR_CONFIGURATION

FDL_Open

概　要	FDL の準備処理
ヘッダ	fdl.h
宣　言	void __far FDL_Open(void)
説　明	FDL の準備処理を行います。 ※FDL のライブラリ関数です。
引　数	なし
リターン値	なし

FDL_Close

概　要	FDL の終了処理
ヘッダ	fdl.h
宣　言	void __far FDL_Close(void)
説　明	FDL の終了処理を行います。 ※FDL のライブラリ関数です。
引　数	なし
リターン値	なし

EEL_Init

概　要	EEL の初期化	
ヘッダ	eel.h	
宣　言	eel_status_t __far EEL_Init(void)	
説　明	EEPROM エミュレーションで使用する RAM の初期化処理を行います。 ※EEL のライブラリ関数です。	
引　数	なし	
リターン値	正常終了	: EEL_OK
	初期化エラー	: EEL_ERR_CONFIGURATION

EEL_Open

概　要	EEL の準備処理	
ヘッダ	eel.h	
宣　言	void __far EEL_Open(void)	
説　明	EEPROM エミュレーションを実行できる状態にします。 ※EEL のライブラリ関数です。	
引　数	なし	
リターン値	なし	

EEL_Close

概　要	EEL の終了処理	
ヘッダ	eel.h	
宣　言	void __far EEL_Close(void)	
説　明	EEPROM エミュレーションを実行できない状態にします。 ※EEL のライブラリ関数です。	
引　数	なし	
リターン値	なし	

EEL_Execute

概　要	各コマンドによるデータ・フラッシュ操作の実行	
ヘッダ	eel.h	
宣　言	void __far EEL_Execute(__near eel_request_t* request_pstr)	
説　明	EEPROM エミュレーションを操作するための各処理をコマンド形式で本関数の引数に設定させ、処理を実行させます。 ※EEL のライブラリ関数です。	
引　数	__near eel_request_t* request_pstr	リクエスト・ストラクチャーのポインタ
リターン値	なし	

EEL_Handler

概　要	実行中の EEL を制御
ヘッダ	eel.h
宣　言	void __far EEL_Handler(void)
説　明	EEL_Execute 関数で指定された EEPROM エミュレーションの処理を、本関数で継続実行します。 ※EEL のライブラリ関数です。
引　数	なし
リターン値	なし

EEL_GetSpace

概　要	EEL ブロックの空き容量の確認処理	
ヘッダ	eel.h	
宣　言	eel_status_t __far EEL_GetSpace(__near eel_u16* space_pu16)	
説　明	EEL ブロックの空き容量を取得します。 ※EEL のライブラリ関数です。	
引　数	__near eel_u16* space_pu16	現在の有効ブロックの空き容量の情報が入力されるアドレス
リターン値	正常終了	：EEL_OK
	EEL_Init 関数未実行	：EEL_ERR_INITIALIZATION
	EEL_CMD_STARTUP が正常終了していない状態で実行	：EEL_ERR_ACCESS_LOCKED
	コマンド実行中	：EEL_ERR_REJECTED

6.8　フローチャート

6.8.1　全体フローチャート

図 6.2に全体フローチャートを示します。

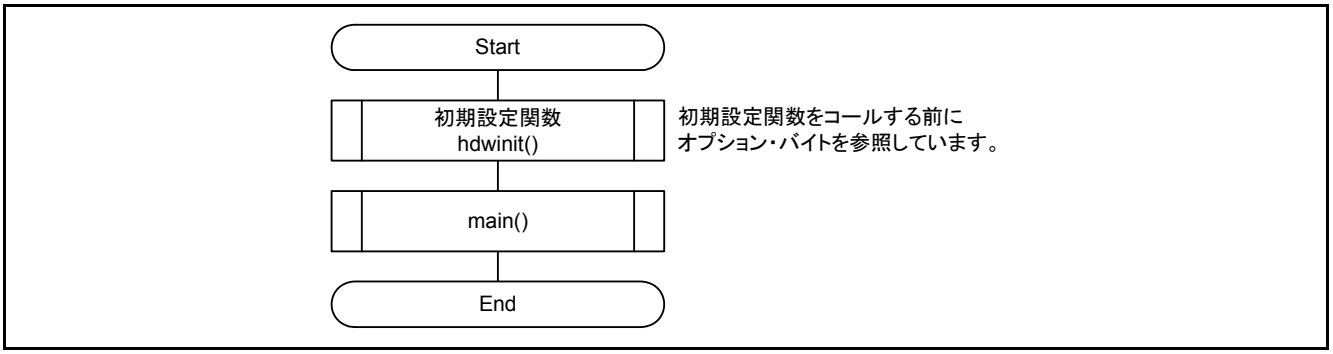


図 6.2 全体フローチャート

6.8.2　周辺機能初期設定

図 6.3に周辺機能初期設定のフローチャートを示します。

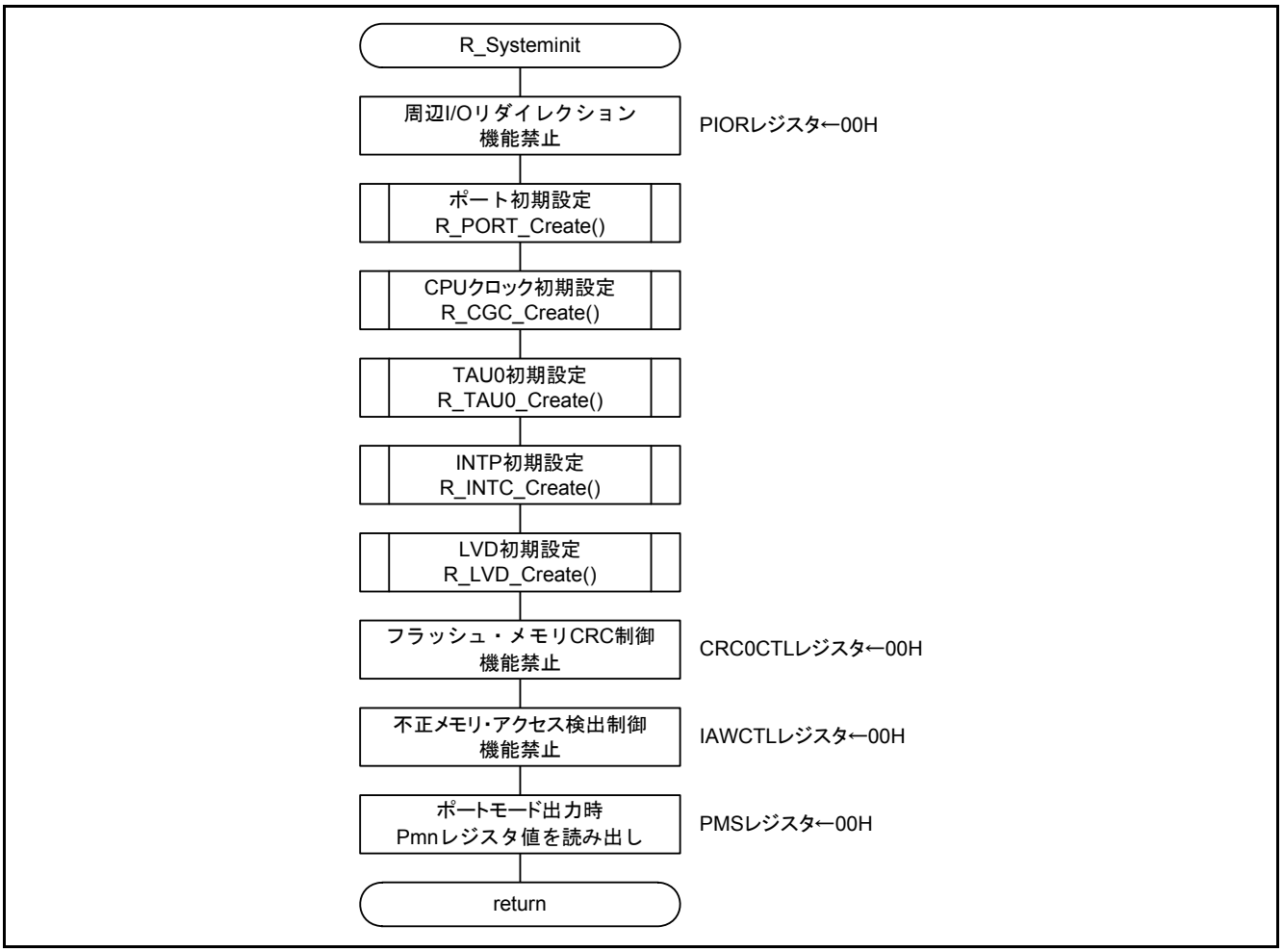


図 6.3 周辺機能初期設定

6.8.3　ポート初期設定

図 6.4にポート初期設定のフローチャートを示します。

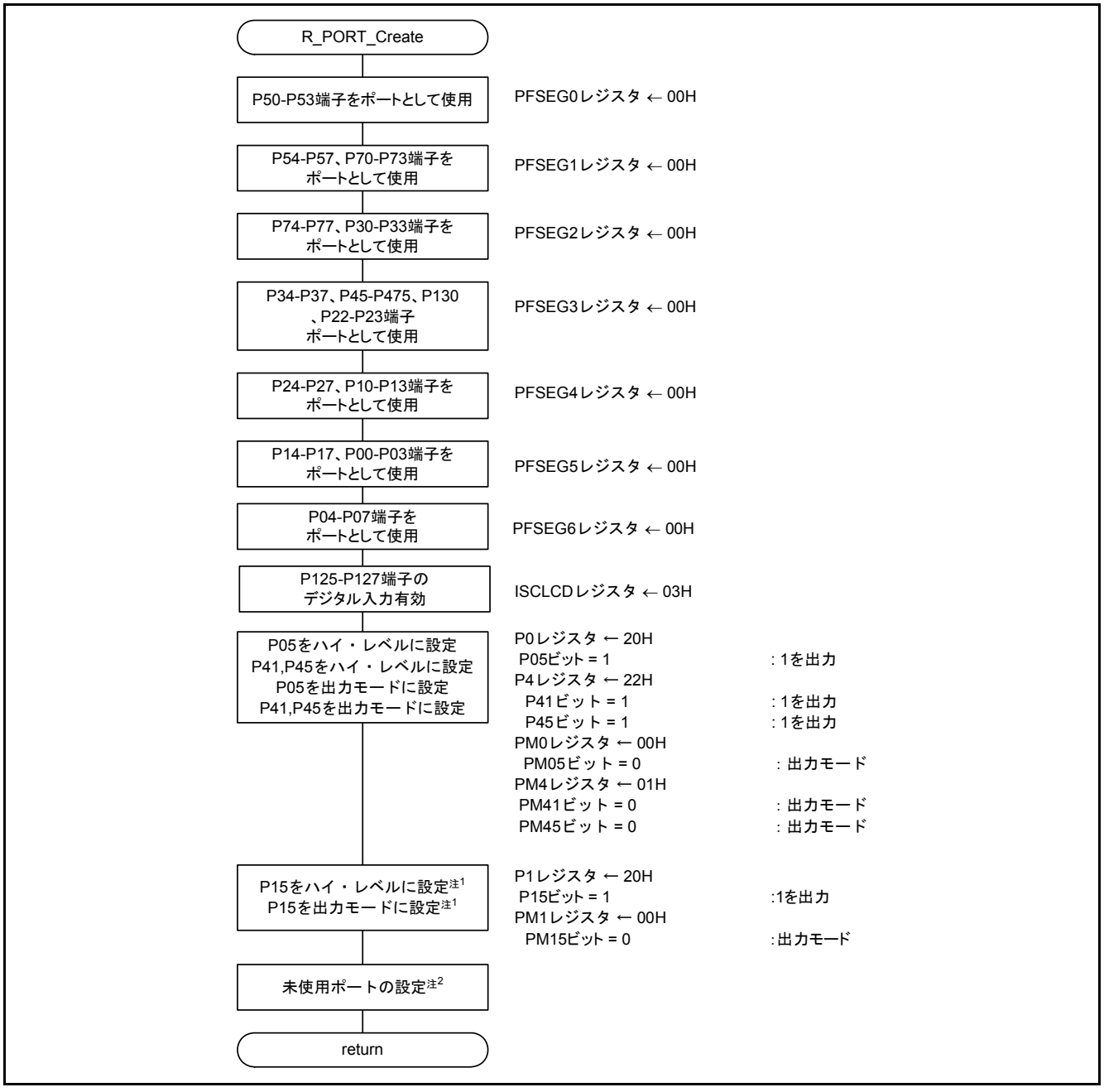


図 6.4 ポート初期設定

注 1　未使用 LED を消灯させる設定です。

注 2　未使用ポートの設定については、RL78/L13 ユーザーズマニュアル ハードウェア編を参照してください。

注意　未使用のポートは、端子処理などを適切に行い、電気的特性を満たすように設計してください。
また、未使用の入力専用ポートは個別に抵抗を介して V_{DD} 又は V_{SS} に接続してください。

6.8.4　CPU クロック初期設定

図 6.5にCPU クロック初期設定のフローチャートを示します。

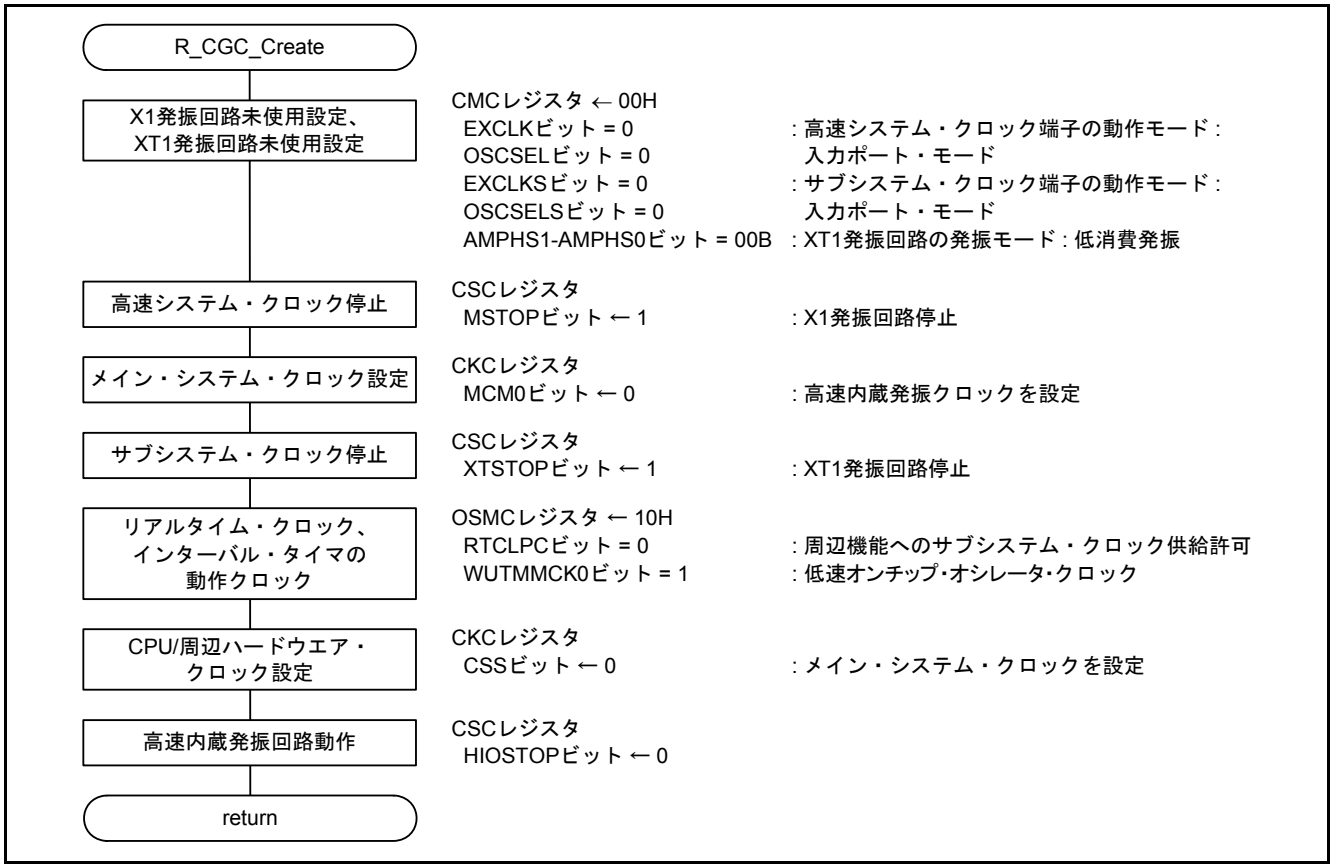


図 6.5 CPU クロック初期設定

6.8.5　TAU0 初期設定

図 6.6、図 6.7に TAU0 初期設定のフローチャートを示します。

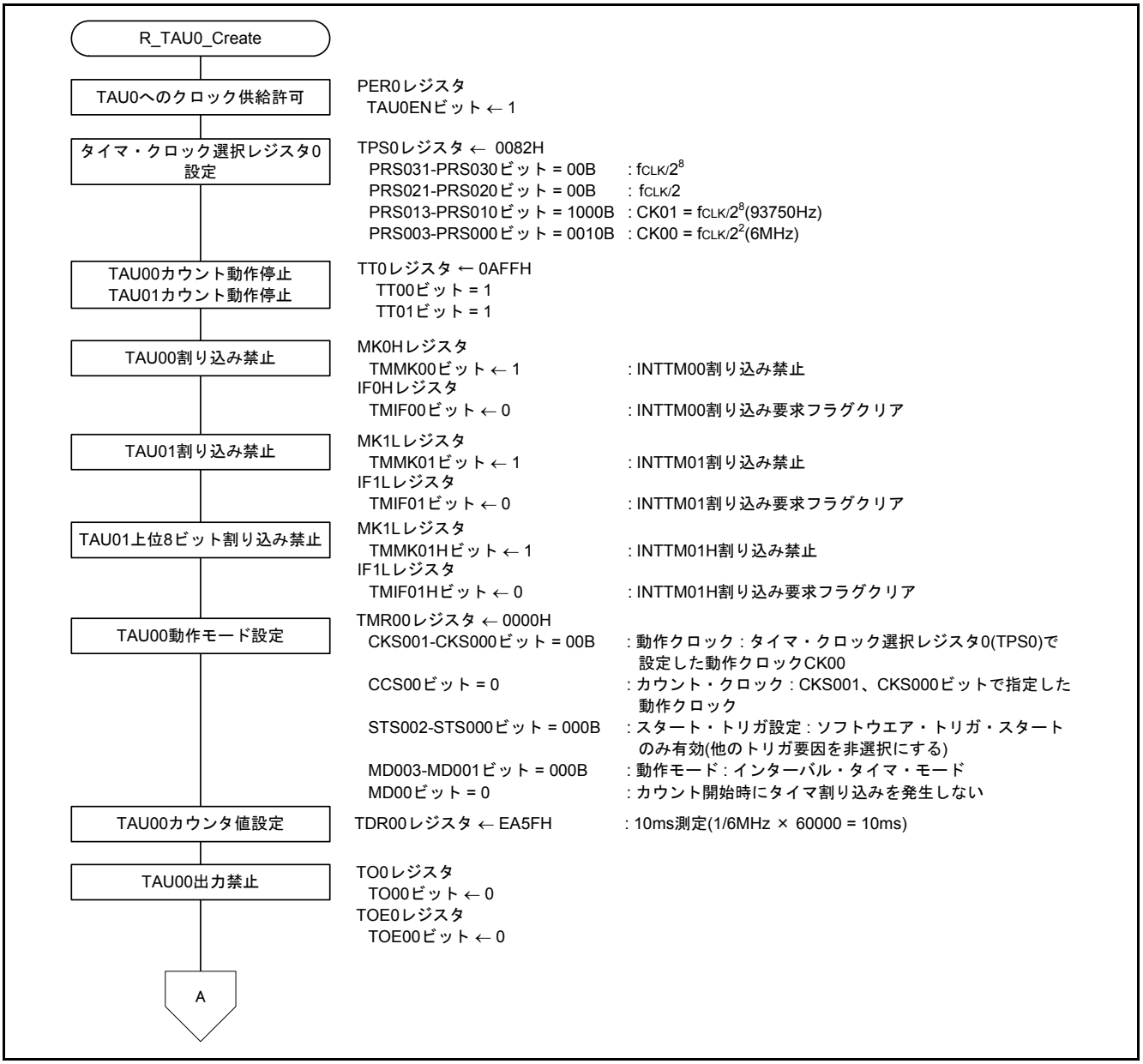


図 6.6 TAU0 初期設定(1/2)

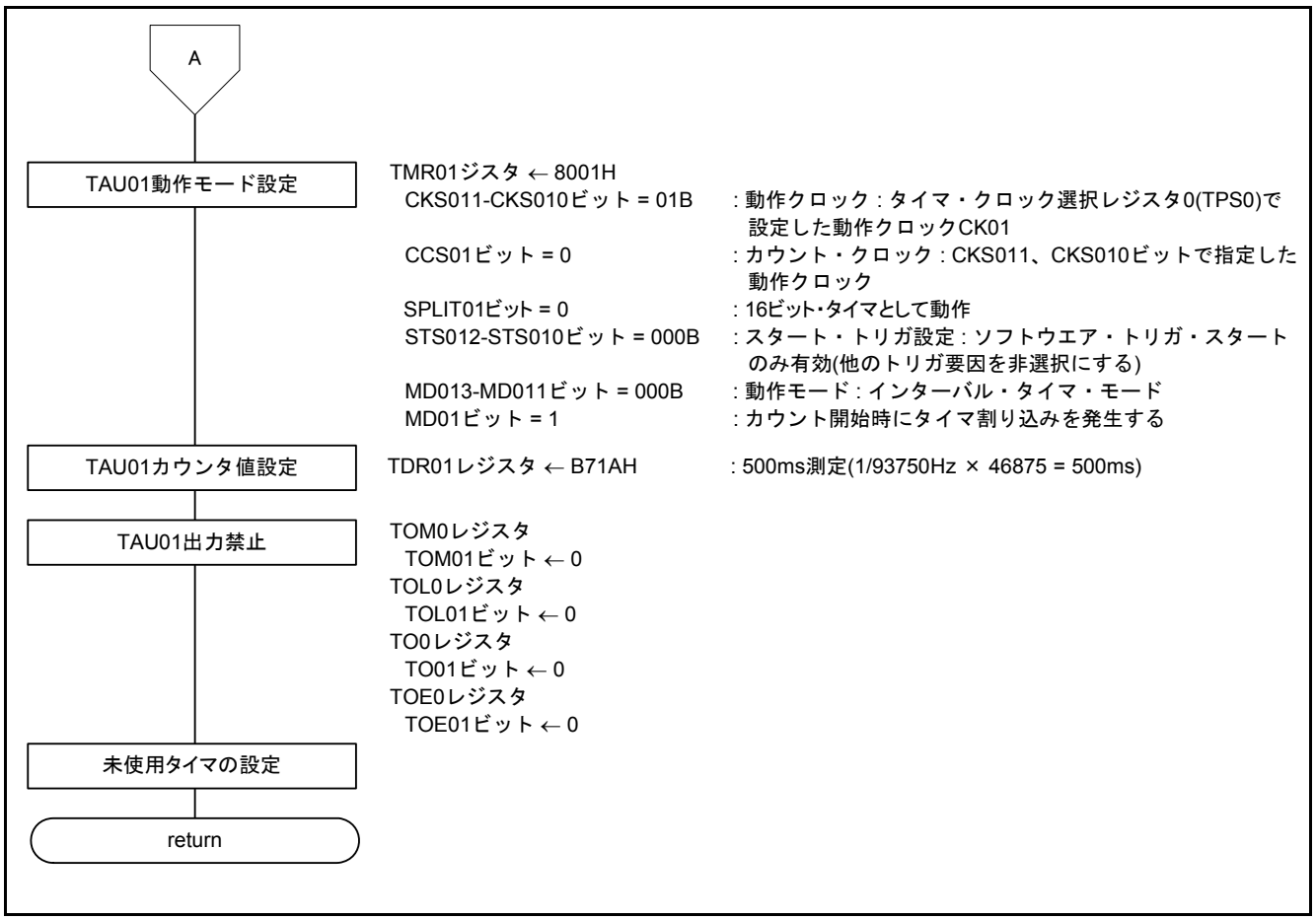


図 6.7 TAU0 初期設定(2/2)

6.8.6　INTP 初期設定

図 6.8にINTP 初期設定のフローチャートを示します。

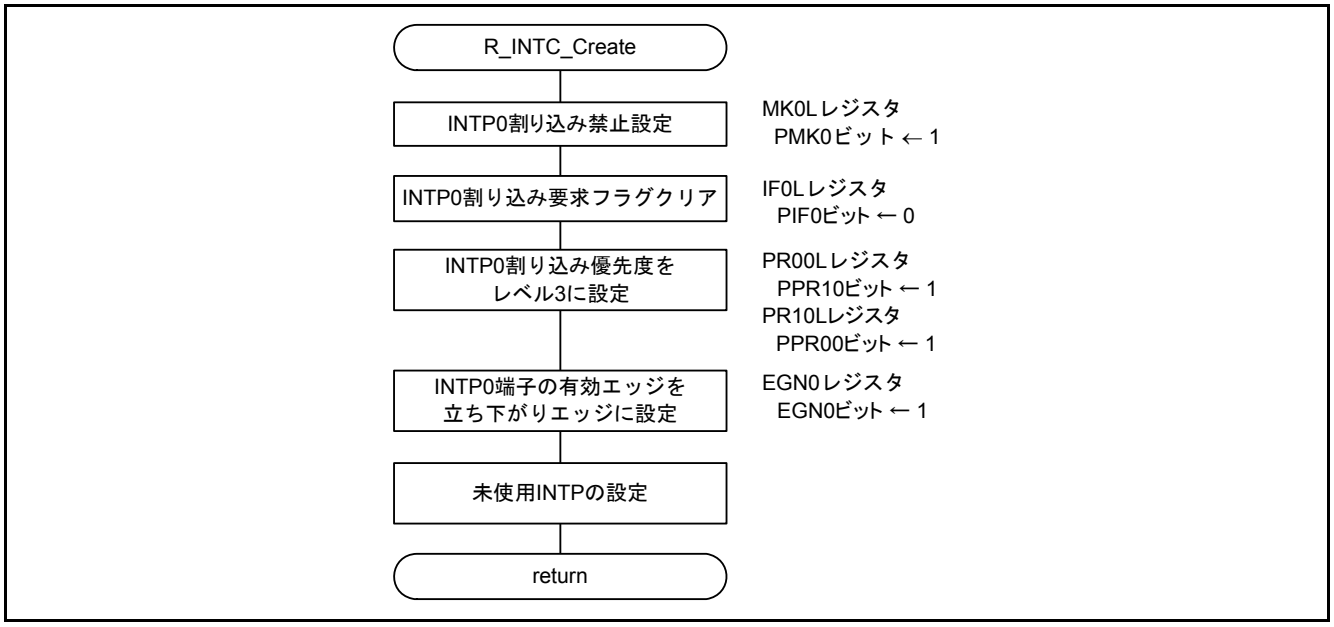


図 6.8 INTP 初期設定

6.8.7　LVD 初期設定

図 6.9にLVD 初期設定のフローチャートを示します。

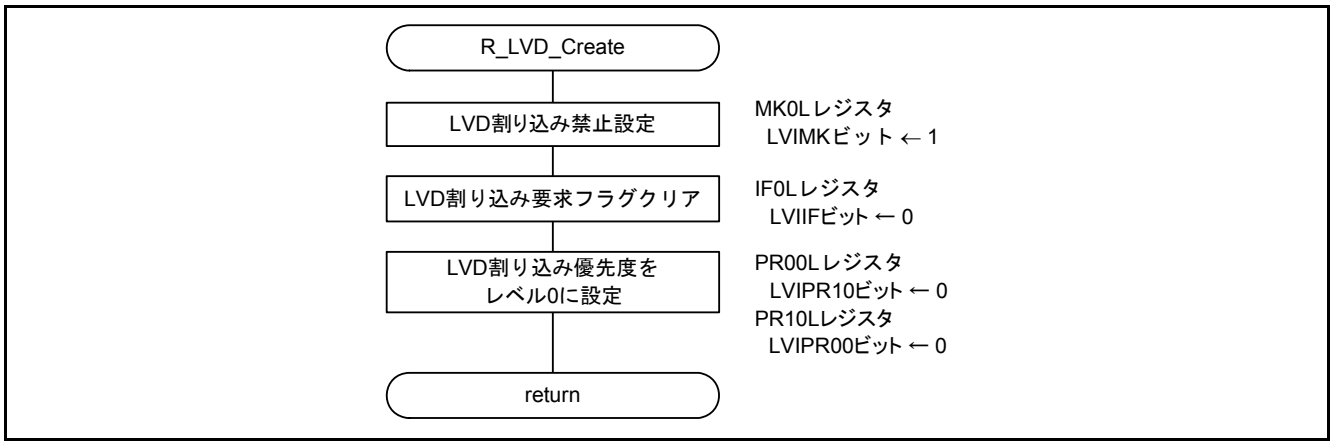


図 6.9 LVD 初期設定

6.8.8　メイン処理

図 6.10、図 6.11にメイン処理のフローチャートを示します。

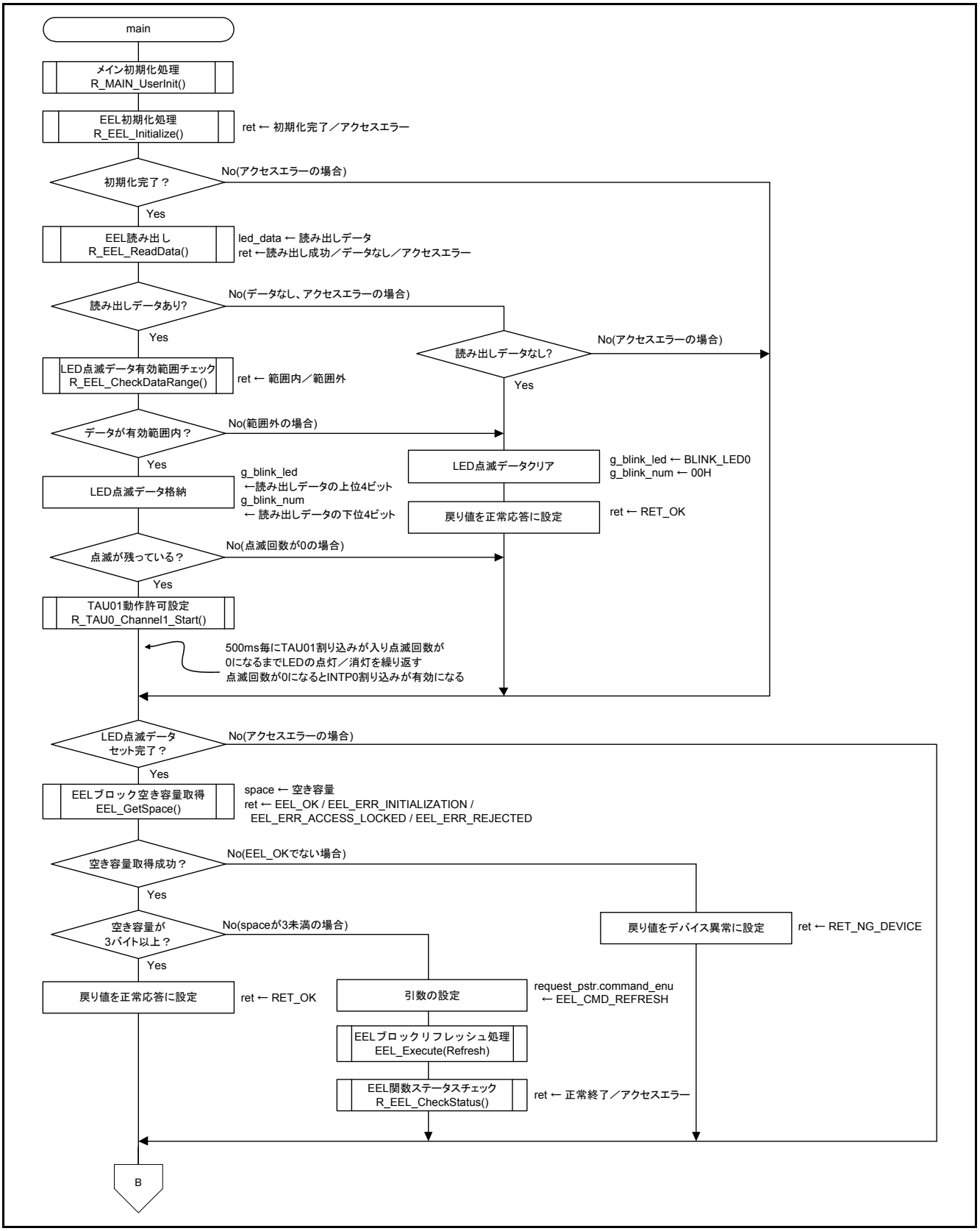


図 6.10 メイン処理(1/2)

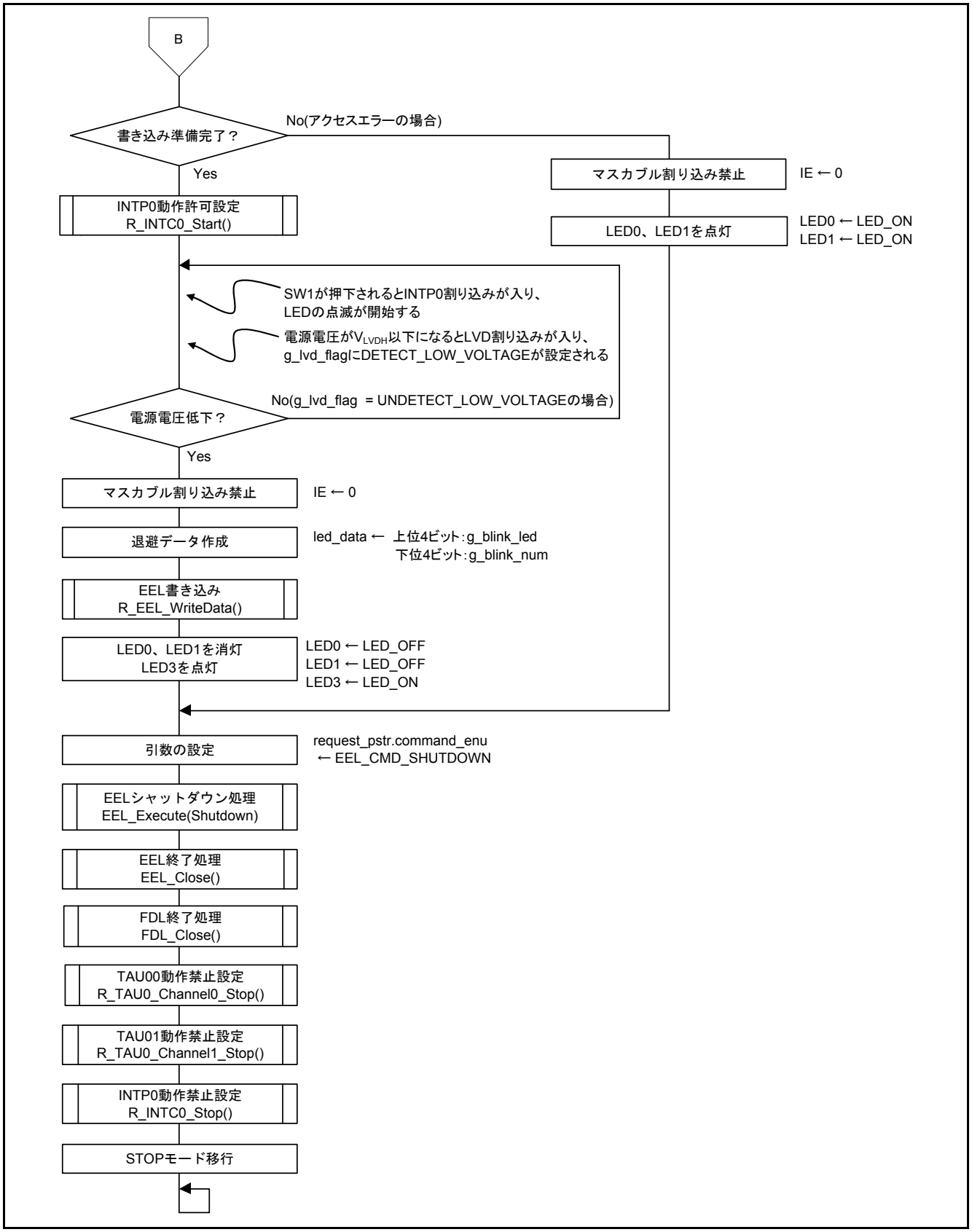


図 6.11 メイン処理(2/2)

6.8.9　メイン初期化処理

図 6.12にメイン初期化処理のフローチャートを示します。

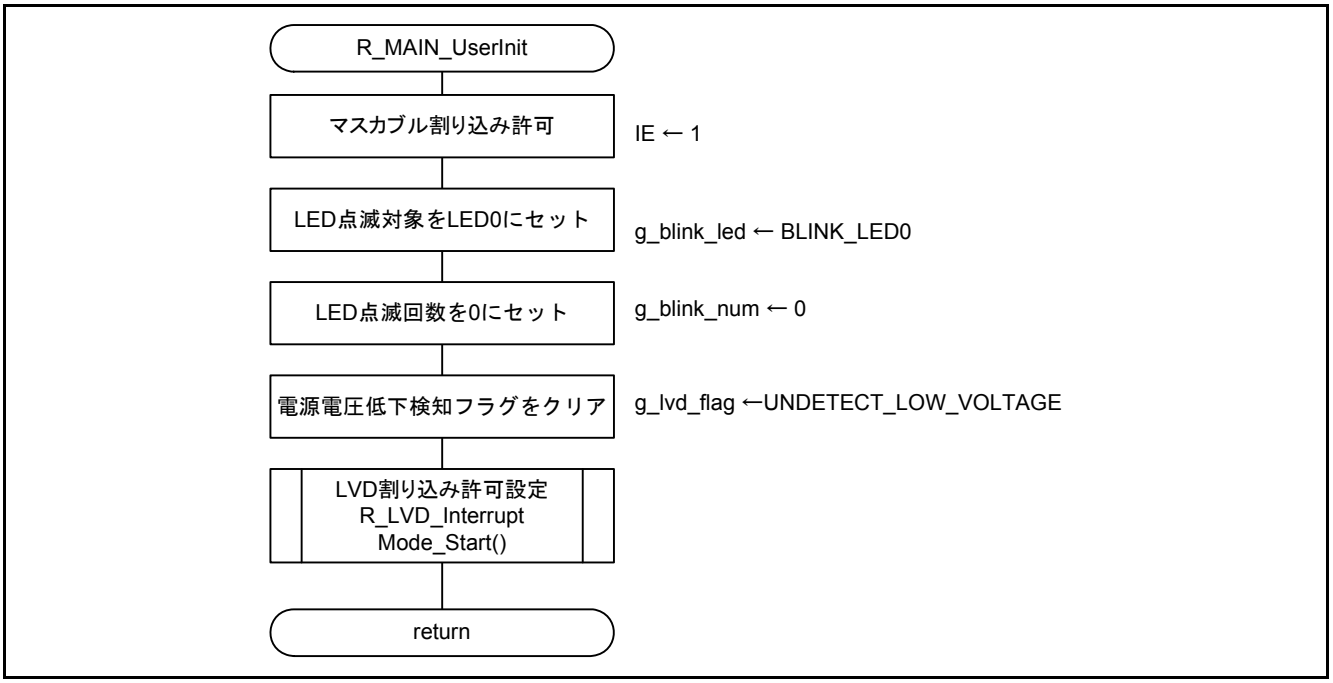


図 6.12 メイン初期化処理

6.8.10 EEL 初期化処理

図 6.13にEEL 初期化処理のフローチャートを示します。

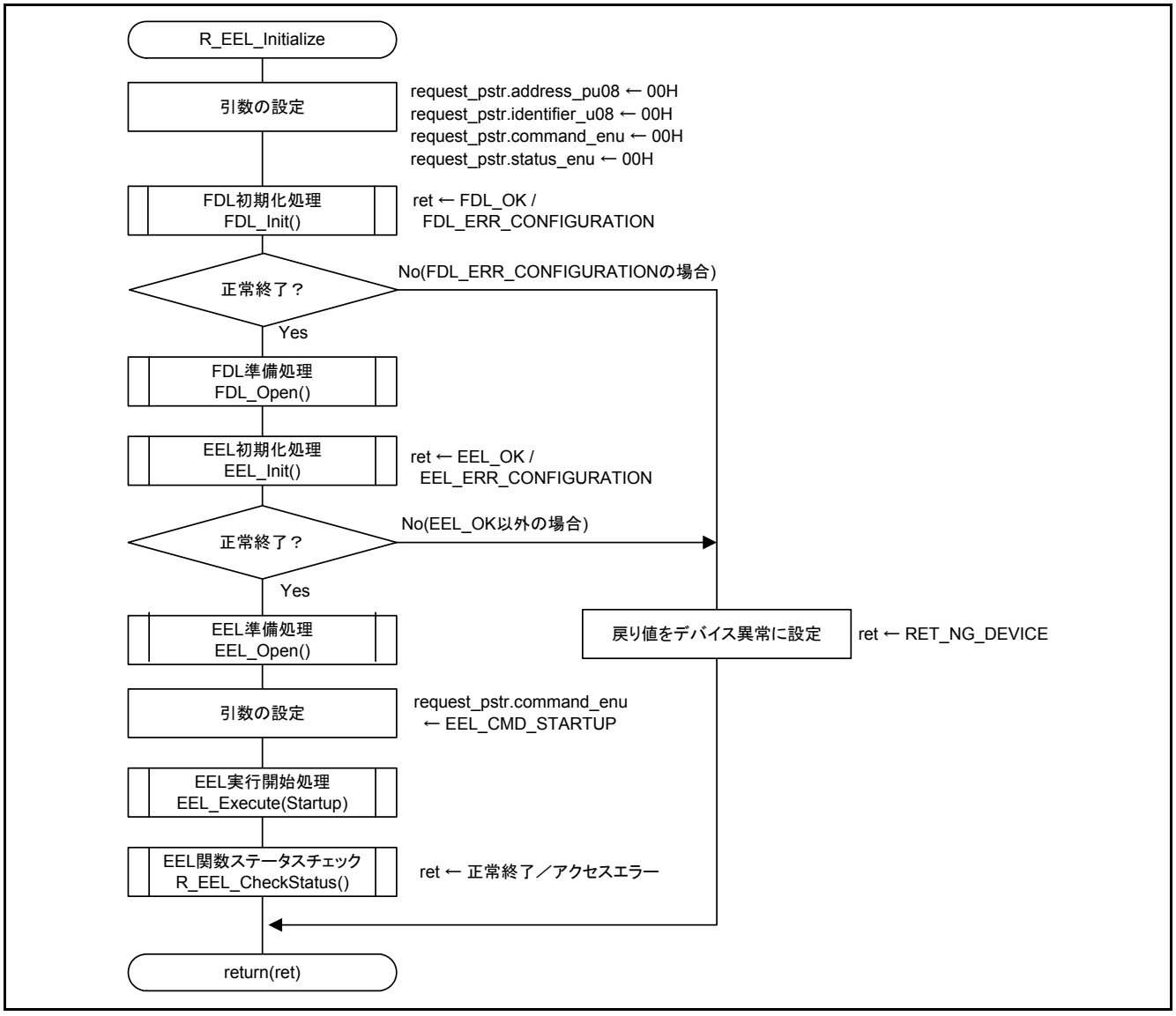


図 6.13 EEL 初期化処理

6.8.11　EEL 読み出し

図 6.14にEEL 読み出しのフローチャートを示します。

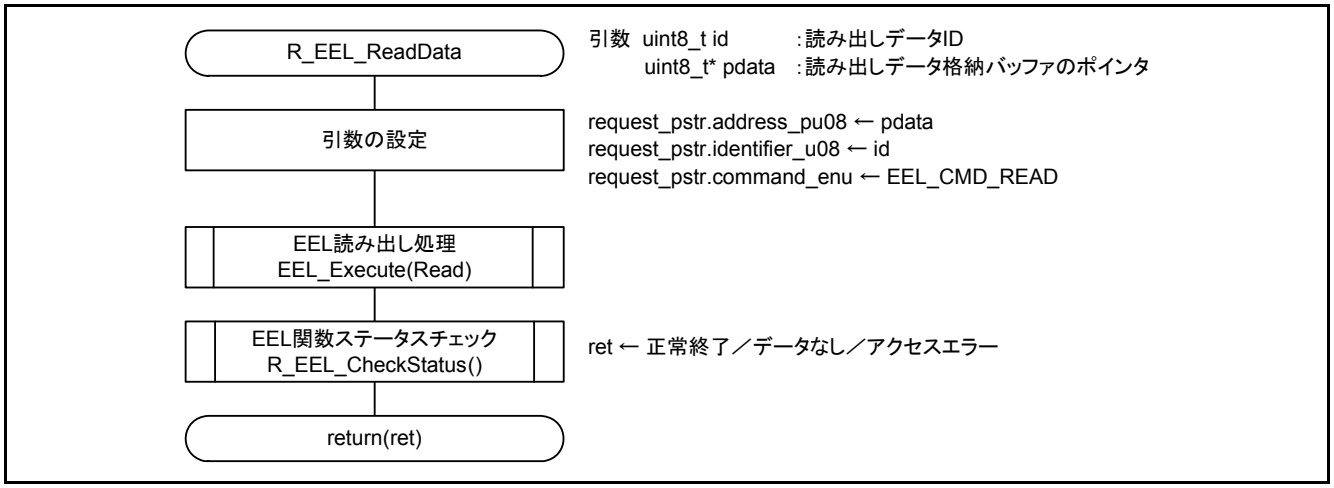


図 6.14 EEL 読み出し

6.8.12　LED 点滅データ有効範囲チェック

図 6.15にLED 点滅データ有効範囲チェックのフローチャートを示します。

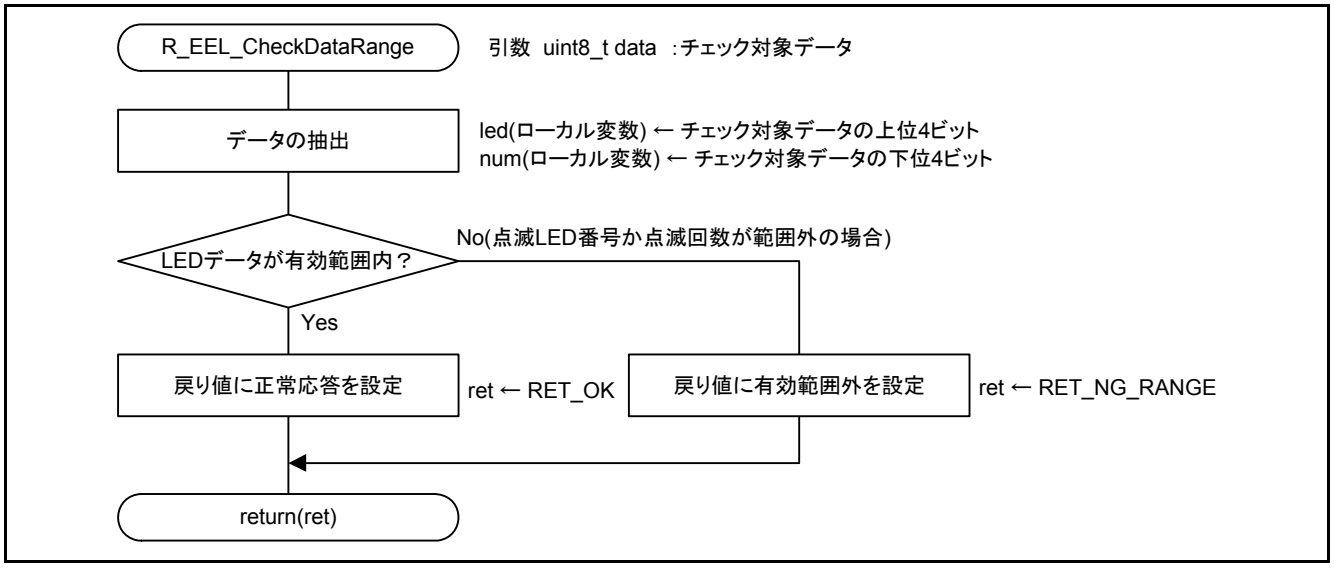


図 6.15 LED 点滅データ有効範囲チェック

6.8.13　EEL 関数ステータスチェック

図 6.16に EEL 関数ステータスチェックのフローチャートを示します。

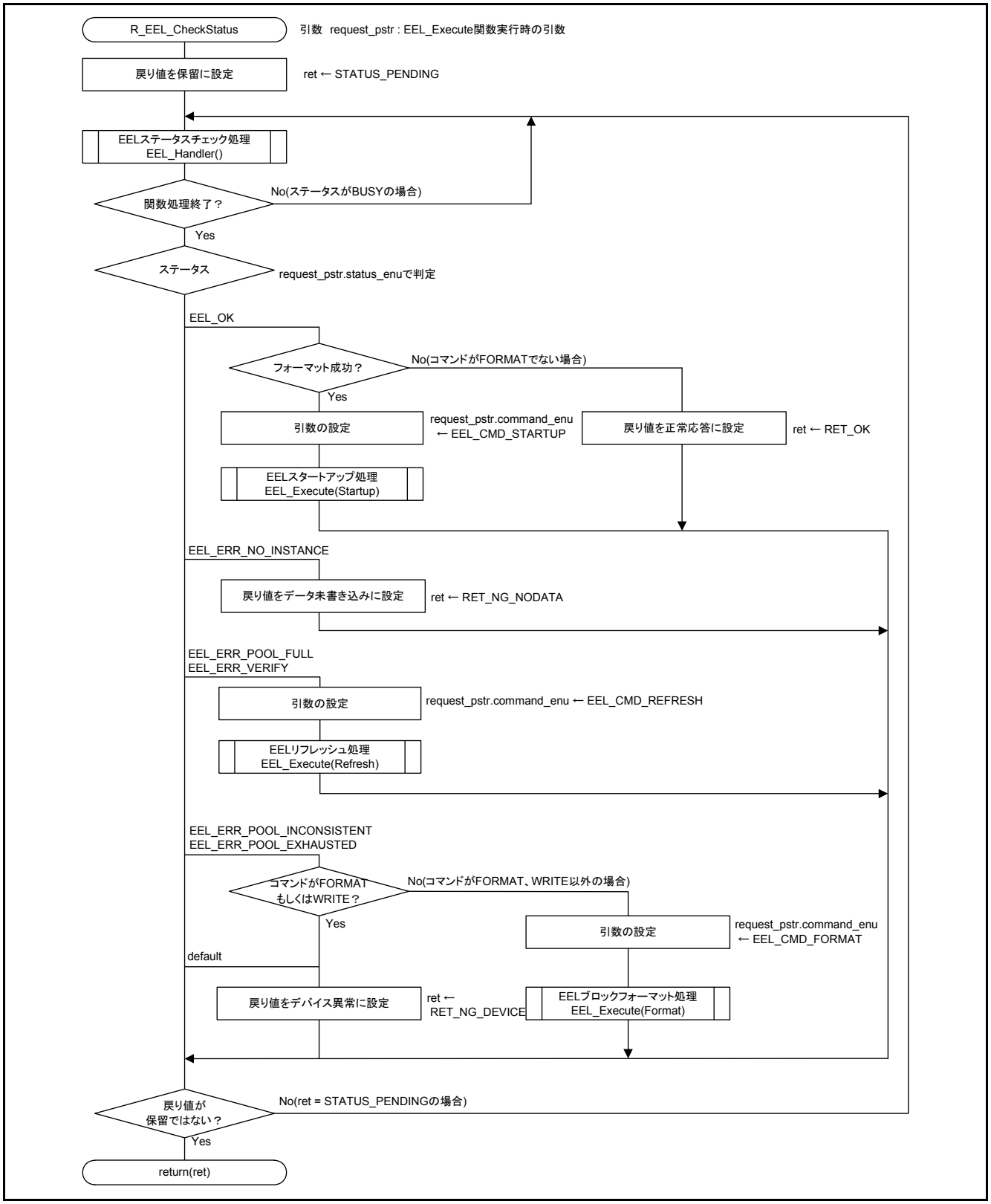


図 6.16 EEL 関数ステータスチェック

6.8.14　TAU01 動作許可設定

図 6.17にTAU01 動作許可設定のフローチャートを示します。

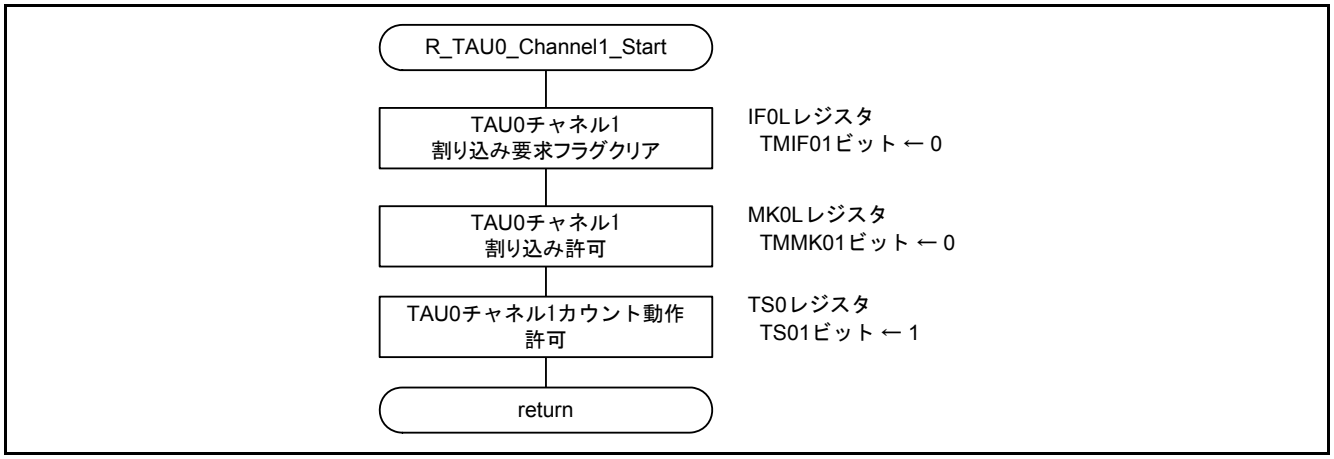


図 6.17 TAU01 動作許可設定

6.8.15 TAU01 割り込みハンドラ

図 6.18にTAU01 割り込みハンドラのフローチャートを示します。

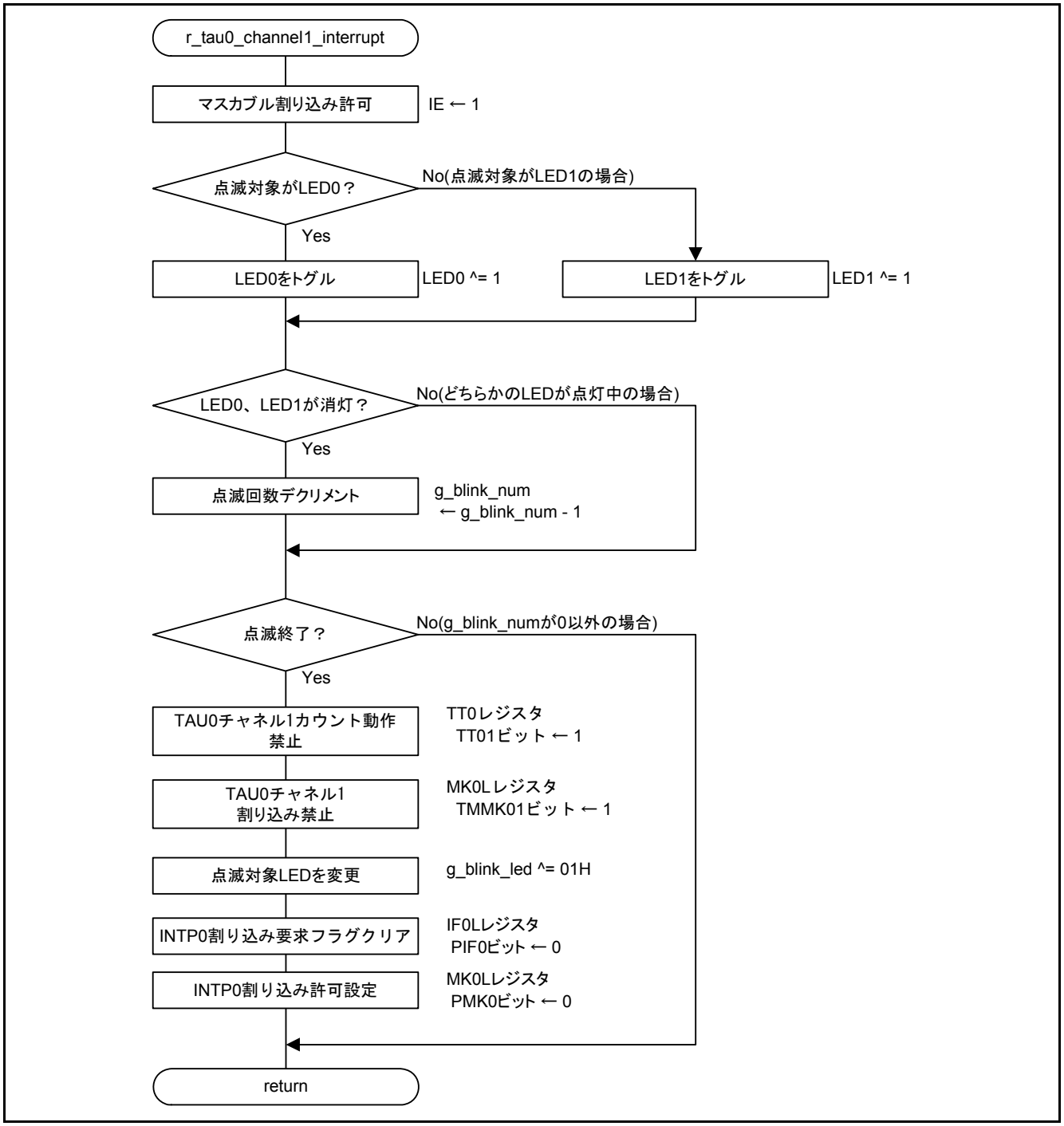


図 6.18 TAU01 割り込みハンドラ

6.8.16　TAU01 動作禁止設定

図 6.19にTAU01 動作禁止設定のフローチャートを示します。

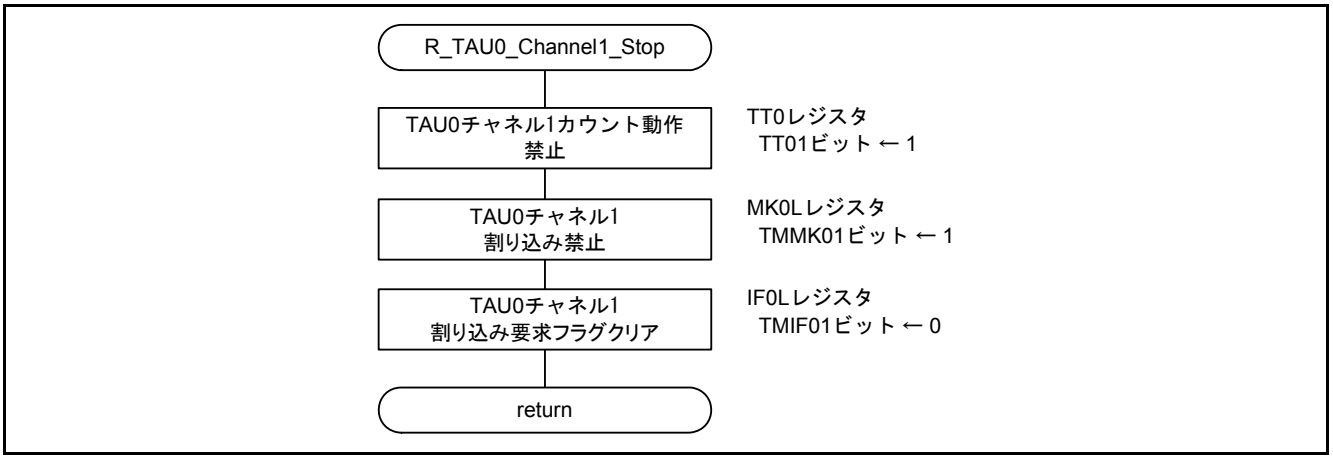


図 6.19 TAU01 動作禁止設定

6.8.17　INTP0 動作許可設定

図 6.20にINTP0 動作許可設定のフローチャートを示します。

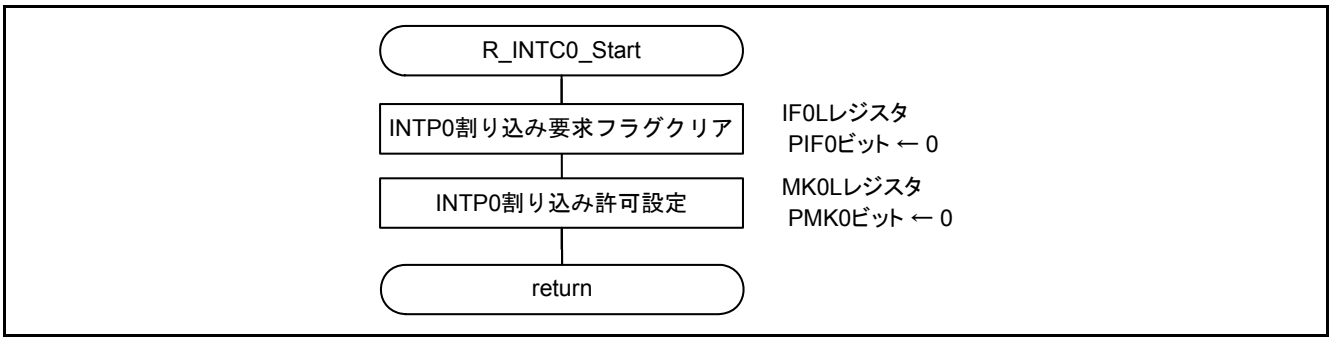


図 6.20 INTP0 動作許可設定

6.8.18 INTP0 割り込みハンドラ

図 6.21にINTP0 割り込みハンドラのフローチャートを示します。

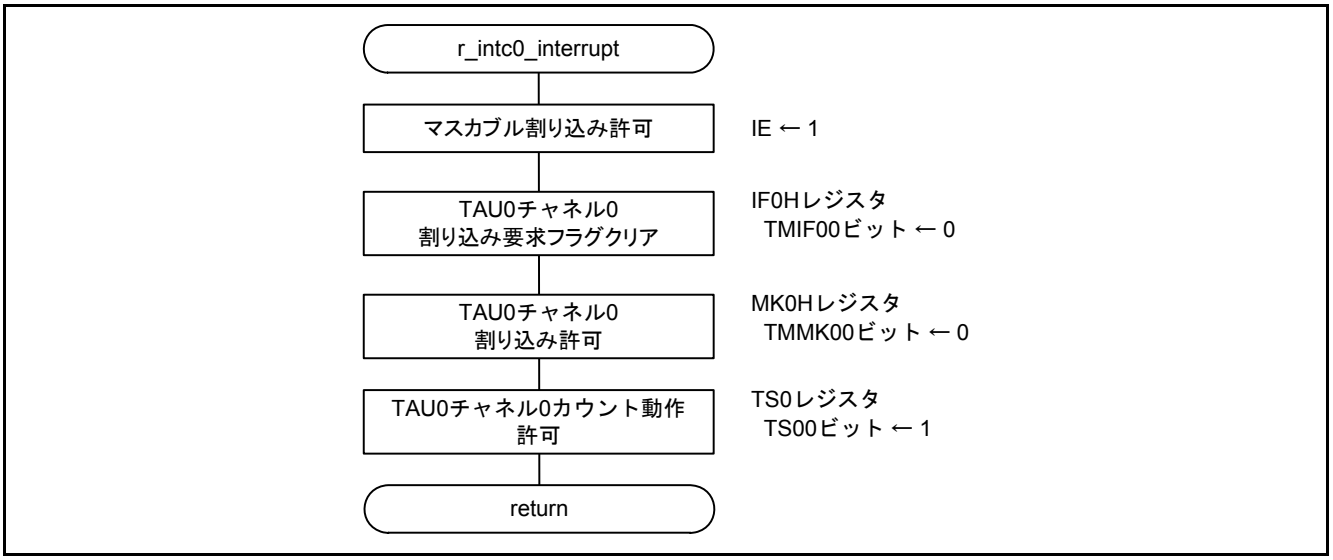


図 6.21 INTP0 割り込みハンドラ

6.8.19 TAU00 動作許可設定

図 6.22にTAU00 動作許可設定のフローチャートを示します。

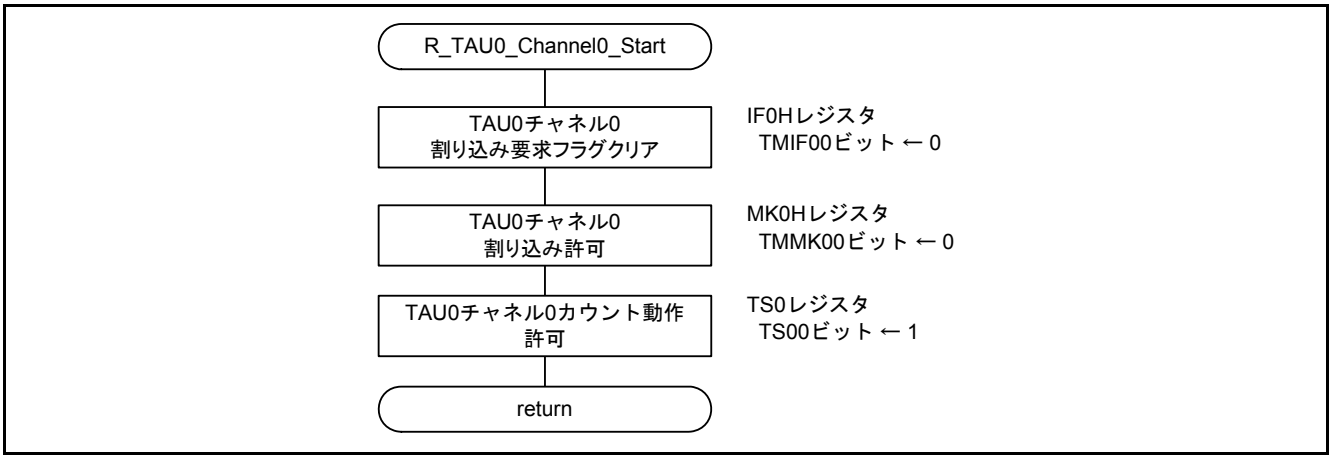


図 6.22 TAU00 動作許可設定

6.8.20　TAU00 割り込みハンドラ

図 6.23にTAU00 割り込みハンドラのフローチャートを示します。

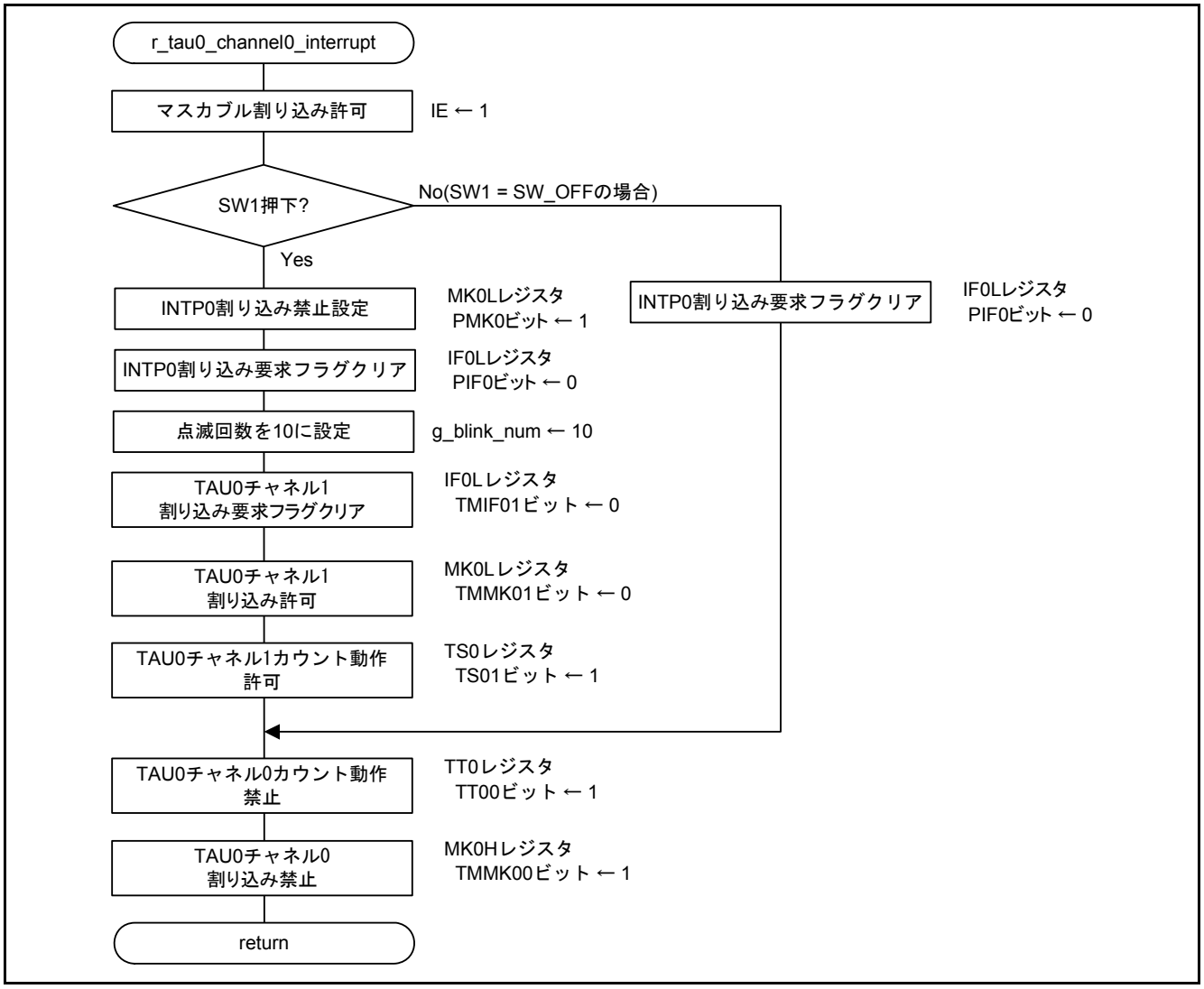


図 6.23 TAU00 割り込みハンドラ

6.8.21　EEL 書き込み

図 6.24にEEL 書き込みのフローチャートを示します。

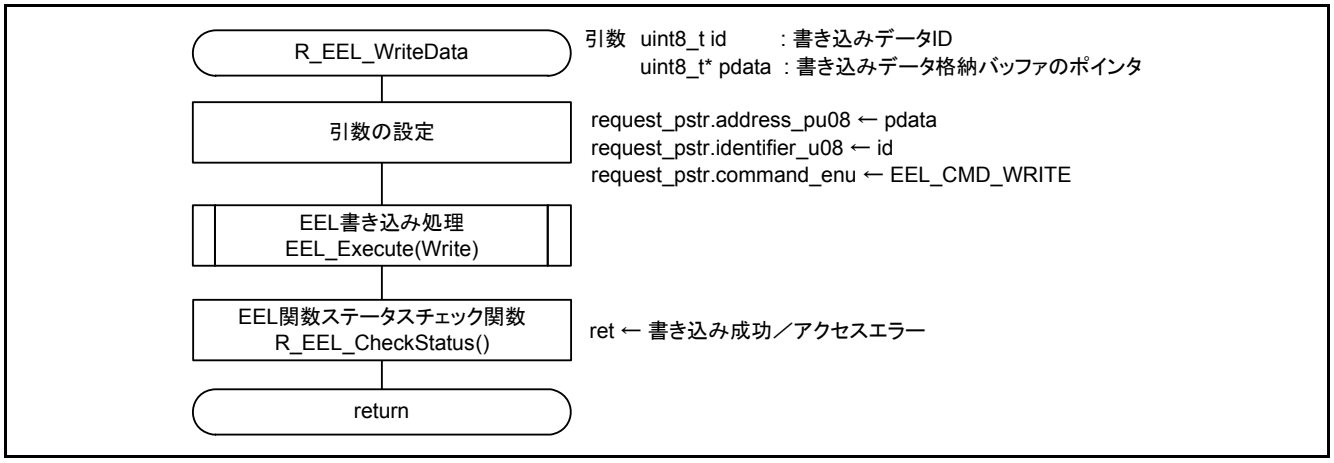


図 6.24 EEL 書き込み

6.8.22　TAU00 動作禁止設定

図 6.25にTAU00 動作禁止設定のフローチャートを示します。

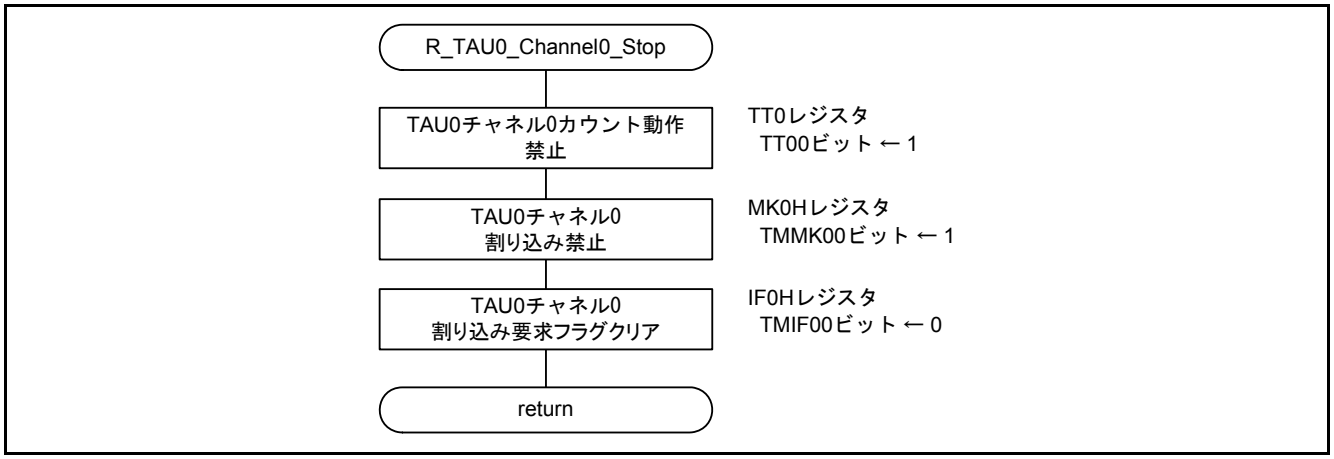


図 6.25 TAU00 動作禁止設定

6.8.23　INTP0 動作禁止設定

図 6.26にINTP0 動作禁止設定のフローチャートを示します。

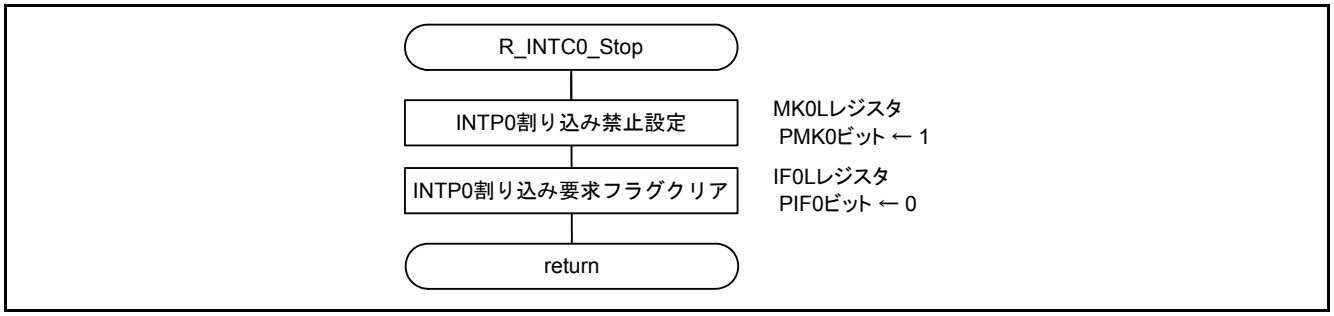


図 6.26 INTP0 動作禁止設定

6.8.24　LVD 割り込み許可設定

図 6.27にLVD 割り込み許可設定のフローチャートを示します。

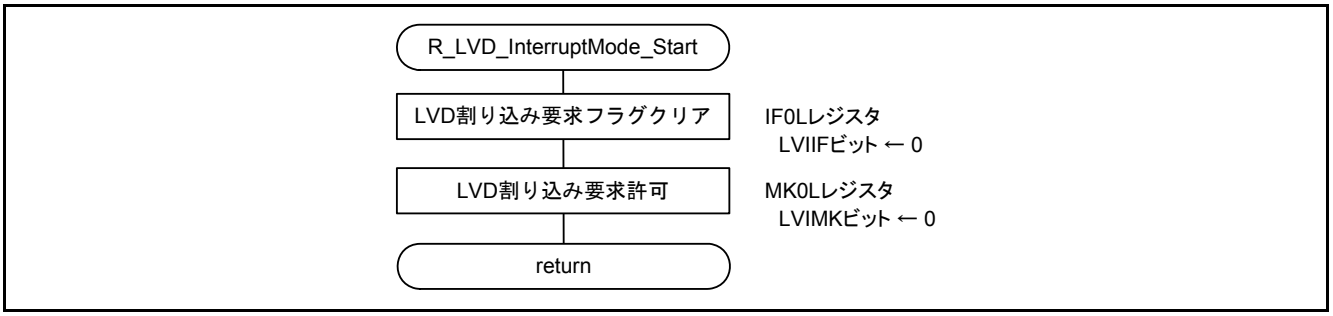


図 6.27 LVD 割り込み許可設定

6.8.25　LVD 割り込みハンドラ

図 6.28にLVD 割り込みハンドラのフローチャートを示します。

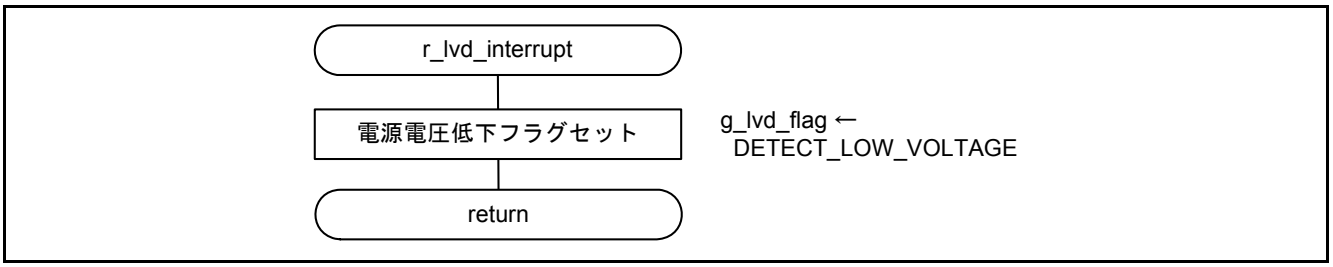


図 6.28 LVD 割り込みハンドラ

6.9 EEL の取り込み方

本アプリケーションで使用する EEL ファイルをプロジェクトへ取り込む方法を以下に記載します。

6.9.1 CubeSuite+版

(1) プロジェクトのルートディレクトリに以下のファイルをコピーする。
- fdl.h
- fdl_types.h
- fdl.lib
- eel.h
- eel_types.h
- eel.lib

(2) CubeSuite+のプロジェクトツリーで「ファイル」を右クリックし、「追加」→「既存のファイルを追加」で拡張子別にコピーしたファイルを選択する。（.c、.h、.lib）

注意 EEL に含まれるディレクトリ smprl78 内にあるファイルを取り込まないでください。必ず、サンプルコードに含まれているファイルを使用してください。（eel_descriptor.c 等）
ファイルを上書きしてしまった場合は、「2.3 EEL ユーザー設定初期値」に従い修正を行ってください。

6.9.2 IAR 版

(1) プロジェクト内、src ディレクトリに以下のファイルをコピーする。
- fdl.h
- fdl_types.h
- fdl.r87
- eel.h
- eel_types.h
- eel.r87

(2) e2studio のプロジェクト・エクスプローラーでプロジェクト名を右クリックし、「更新」を選択する。

注意 FDL/EEL に含まれるディレクトリ smp 内にあるファイルは取り込まないでください。必ず、サンプルコードに含まれているファイルを使用してください。（eel_descriptor.c 等）
ファイルを上書きしてしまった場合は、「2.3 EEL ユーザー設定初期値」に従い修正を行ってください。

6.10　サンプルコードの修正について

コード生成をやり直す場合は、以下のようにファイル及びプロジェクトの修正が必要になる場合があります。

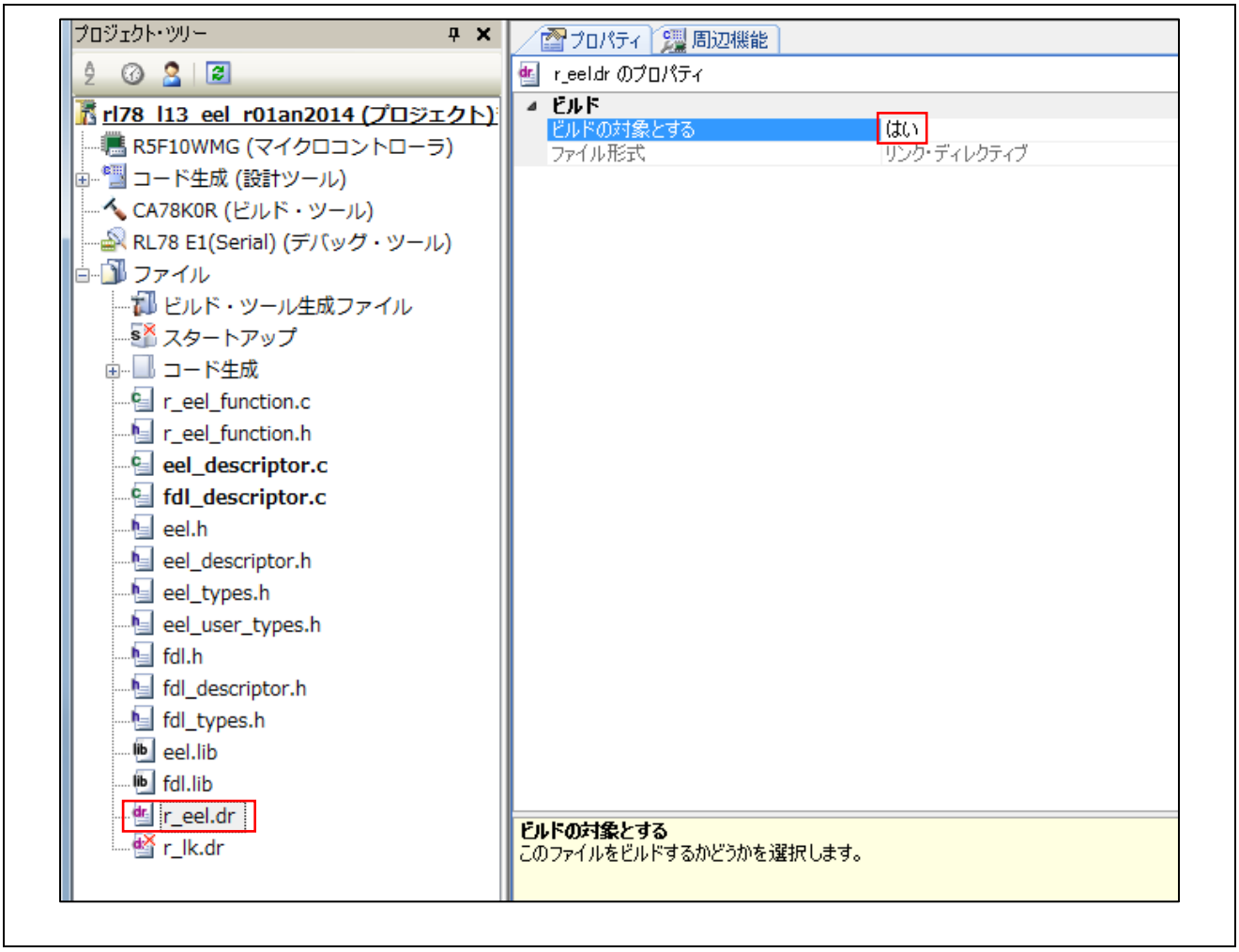
対象環境　：CubeSuite+版

ファイル名：r_eel.dr

修正箇所　：■ビルド対象設定

CubeSuite+のプロジェクトツリーから r_eel.dr を右クリックし、プロパティを開く。

「ビルド対象とする」を「いいえ」から「はい」に修正



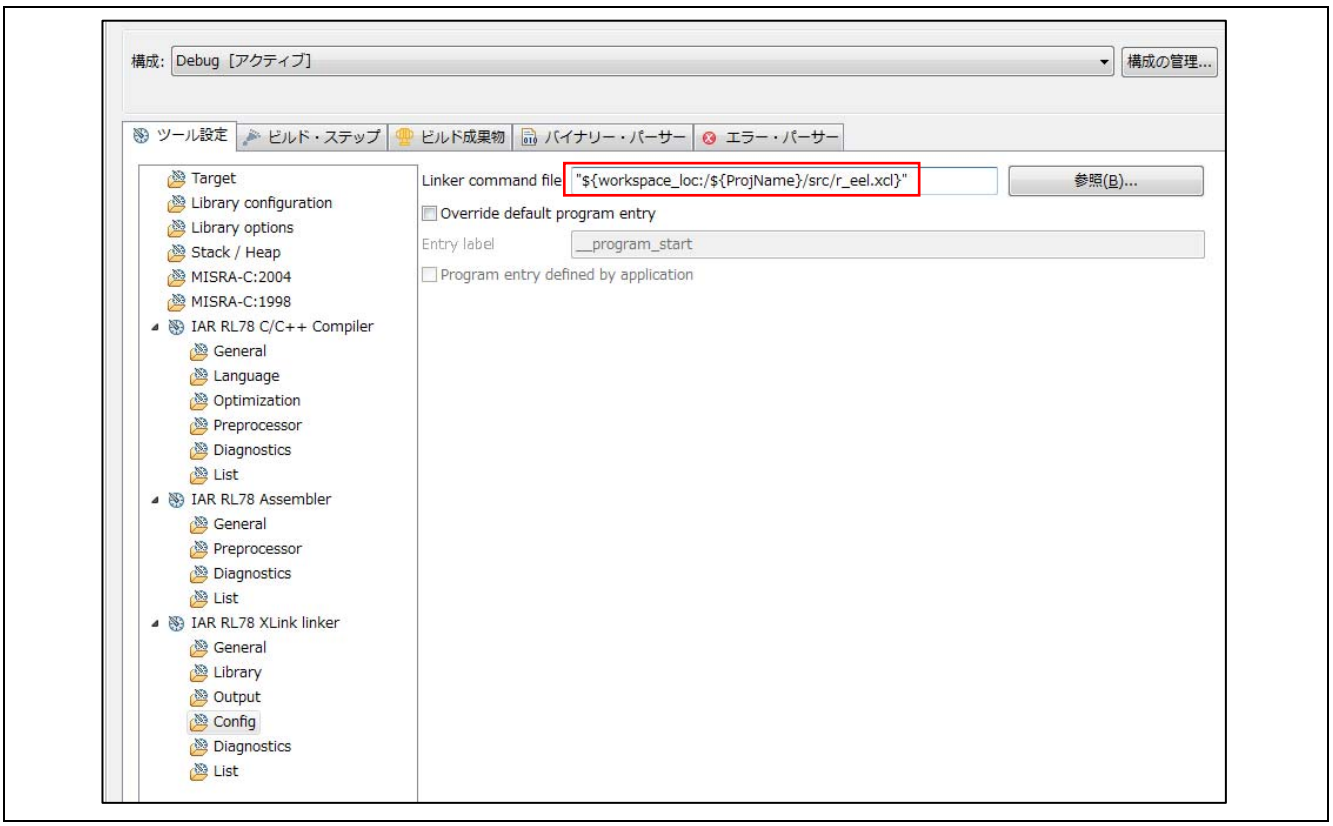
対象環境　：IAR 版

ファイル名：r_eel.xcl

修正箇所　：■リンカファイル設定

e2studio のプロジェクト・エクスプローラーからプロジェクト名を右クリックし、
プロパティを開く。

「C/C++ ビルド」→「設定」→「ツール設定」→「IAR RL78 XLink linker」→「Config」で
「Linker command file」の設定を「"${workspace_loc:/${ProjName}/src/r_eel.xcl}"」に修正



対象環境　：CubeSuite+版、IAR 版

ファイル名：r_cg_port.c

修正箇所　：■PFSEG3 レジスタ設定コード

“_04_PFDEG_DEFAULT”を”_00_PFDEG_PORT”に修正

■PFSEG6 レジスタ設定コード

ISCLED レジスタ設定コードの前に追加

```
/***********************************************************************************************************************
* Function Name: R_PORT_Create
* Description  : This function initializes the Port I/O.
* Arguments    : None
* Return Value : None
***********************************************************************************************************************/
void R_PORT_Create(void)
{
    PFSEG0 = _00_PFSEG07_PORT | _00_PFSEG06_PORT | _00_PFSEG05_PORT | _00_PFSEG04_PORT;
    PFSEG1 = _00_PFSEG15_PORT | _00_PFSEG14_PORT | _00_PFSEG13_PORT | _00_PFSEG12_PORT | _00_PFSEG11_PORT |
             _00_PFSEG10_PORT | _00_PFSEG09_PORT | _00_PFSEG08_PORT;
    PFSEG2 = _00_PFSEG23_PORT | _00_PFSEG22_PORT | _00_PFSEG21_PORT | _00_PFSEG20_PORT | _00_PFSEG19_PORT |
             _00_PFSEG18_PORT | _00_PFSEG17_PORT | _00_PFSEG16_PORT;
    PFSEG3 = _00_PFSEG30_PORT | _00_PFSEG29_PORT | _00_PFSEG28_PORT | _00_PFSEG27_PORT | _00_PFSEG26_PORT |
             _00_PFSEG25_PORT | _00_PFSEG24_PORT | _00_PFDEG_PORT;
    PFSEG4 = _00_PFSEG38_PORT | _00_PFSEG37_PORT | _00_PFSEG36_PORT | _00_PFSEG35_PORT | _00_PFSEG34_PORT |
             _00_PFSEG33_PORT | _00_PFSEG32_PORT | _00_PFSEG31_PORT;
    PFSEG5 = _00_PFSEG46_PORT | _00_PFSEG45_PORT | _00_PFSEG44_PORT | _00_PFSEG43_PORT | _00_PFSEG42_PORT |
             _00_PFSEG41_PORT | _00_PFSEG40_PORT | _00_PFSEG39_PORT;
    PFSEG6 = _00_PFSEG47_PORT | _00_PFSEG48_PORT | _00_PFSEG49_PORT | _00_PFSEG50_PORT;
    ISCLCD = _02_ISCVL3_VALID | _01_ISCCAP_VALID;
```

対象環境　：CubeSuite+版、IAR 版

ファイル名：r_cg_port.h

修正箇所　：■PFDEG 設定マクロ

“_04_PFDEG_DEFAULT”の前に”_00_PFDEG_PORT”を 0x00U でマクロ定義追加

```
/*
    LCD port function registers 03 (PFSEG03)
*/
/* Port (other than segment output)/segment outputs specification of Pmn pins (PFSEGxx) */
#define _00_PFSEG24_PORT        (0x00U) /* used the P34 pin as port (other than segment output) */
#define _01_PFSEG24_SEG         (0x01U) /* used the P34 pin as segment output */
#define _00_PFSEG25_PORT        (0x00U) /* used the P35 pin as port (other than segment output) */
#define _02_PFSEG25_SEG         (0x02U) /* used the P35 pin as segment output */
#define _00_PFSEG26_PORT        (0x00U) /* used the P46 pin as port (other than segment output) */
#define _08_PFSEG26_SEG         (0x08U) /* used the P46 pin as segment output */
#define _00_PFSEG27_PORT        (0x00U) /* used the P47 pin as port (other than segment output) */
#define _10_PFSEG27_SEG         (0x10U) /* used the P47 pin as segment output */
#define _00_PFSEG28_PORT        (0x00U) /* used the P130 pin as port (other than segment output) */
#define _20_PFSEG28_SEG         (0x20U) /* used the P130 pin as segment output */
#define _00_PFSEG29_PORT        (0x00U) /* used the P22 pin as port (other than segment output) */
#define _40_PFSEG29_SEG         (0x40U) /* used the P22 pin as segment output */
#define _00_PFSEG30_PORT        (0x00U) /* used the P23 pin as port (other than segment output) */
#define _80_PFSEG30_SEG         (0x80U) /* used the P23 pin as segment output */
#define _00_PFDEG_PORT          (0x00U) /* used the P45 pin as port (other than segment output) */
#define _04_PFDEG_DEFAULT       (0x04U) /* PFDEG default value */
```

7. サンプルコード

サンプルコードは、ルネサス エレクトロニクスホームページから入手してください。

8. 参考ドキュメント

RL78/L13 ユーザーズマニュアル ハードウェア編
RL78 ファミリ ユーザーズマニュアル ソフトウェア編
（最新版をルネサス エレクトロニクスホームページから入手してください。）

テクニカルアップデート
（最新の情報をルネサス エレクトロニクスホームページから入手してください。）

ホームページとサポート窓口

ルネサス エレクトロニクスホームページ
http://japan.renesas.com

お問合せ先
http://japan.renesas.com/contact/

改訂記録	RL78/L13　アプリケーションノート　セルフ・プログラミング・ライブラリを用いた外付け EEPROM IC の取り込み（EEPROM エミュレーション編）

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2014.03.25	—	初版発行

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. 未使用端子の処理
   【注意】未使用端子は、本文の「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。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。
2. 電源投入時の処置
   【注意】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. リザーブアドレス（予約領域）のアクセス禁止
   【注意】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、将来の機能拡張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。
4. クロックについて
   【注意】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。
5. 製品間の相違について
   【注意】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、内部 ROM、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、ノイズ輻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。

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、お客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
2. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
3. 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に関し、当社は、何らの責任を負うものではありません。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. 当社製品を改造、改変、複製等しないでください。かかる改造、改変、複製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）、交通用信号機器、
   防災・防犯装置、各種安全装置等
   当社製品は、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（原子力制御システム、軍事機器等）に使用されることを意図しておらず、使用することはできません。たとえ、意図しない用途に当社製品を使用したことによりお客様または第三者に損害が生じても、当社は一切その責任を負いません。なお、ご不明点がある場合は、当社営業にお問い合わせください。
6. 当社製品をご使用の際は、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他の保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障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、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、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。お客様が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. 本資料に記載されている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。また、当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途に使用しないでください。当社製品または技術を輸出する場合は、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。
10. お客様の転売等により、本ご注意書き記載の諸条件に抵触して当社製品が使用され、その使用から損害が生じた場合、当社は何らの責任も負わず、お客様にてご負担して頂きますのでご了承ください。
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。

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その総株主の議決権の過半数を直接または間接に保有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。

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口 http://www.renesas.com

※営業お問合せ窓口の住所は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス株式会社　〒100-0004　千代田区大手町2-6-2（日本ビル）

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：http://japan.renesas.com/contact/

Colophon 3.0