RL78/L12

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（フラッシュ・データ・ライブラリ編） CC-RL

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/L12 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 FDL 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 FDL 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（FDL 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、FDL 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 FDL の取り込み方」を参照してください。

FDL には Renesas CC-RL、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 FDL 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 FDL を確認してください。また、FDL を使用する前に、FDL 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。

表1.1 コンパイラと FDL の対応関係

コンパイラの種類	対応する FDL	ダウンロード元
CS+版	RL78 ファミリ データフラシュライブラリ Type04 パッケージ Ver.2.00	https://www.renesas.com/jp/ja/software-tool/data-flash-libraries#overview
e2studio 版	RL78 ファミリ データフラシュライブラリ Type04 パッケージ Ver.2.00	https://www.renesas.com/jp/ja/software-tool/data-flash-libraries#overview
IAR 版	RL78 ファミリ データフラシュライブラリ Type04 パッケージ Ver.2.00	https://www.renesas.com/jp/ja/software-tool/data-flash-libraries#overview

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/L12 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 FDL 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/G12、RL78/G13、RL78/G14、RL78/G1A、RL78/G1C、RL78/G1E、RL78/I1A、RL78/L13、RL78/L1C

FDL の対応デバイスについては、最新の FDL 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

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、FDL の概要を1.1に、EEL 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 FDL について説明します。

表1.1 セルフ・プログラミング・ライブラリー覧

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　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。

1.2　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）。

1.3　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 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為、ユーザ設定等は必要ありません。

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覧ください。

(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 CS+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、FDL 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、データ構造などに依存します。）
  ※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。
- 保存するデータに合わせて管理領域を簡素化するなど、状況に応じて最適化することができます。

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。

- データ・フラッシュ・メモリにデータ管理領域が必要となるため、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。
- EEPROM IC との通信プログラムに代わり、FDL とデータ管理用プログラムが必要になります。

  データ・フラッシュ・メモリの同一アドレスに上書きをする場合、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。

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 FDL 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。その後、データ退避先アドレスを取得します。

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、FDL 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。なお、データには終端記号として 0x00 が付加されます。

また、FDL 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。

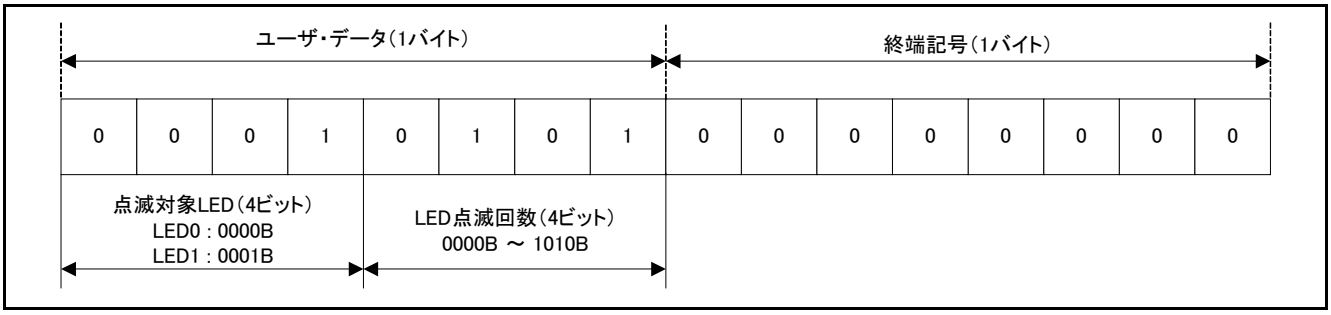


図2.1　格納データ

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
P30	LED 点灯制御(LED0)
P42	LED 点灯制御(LED1)
P52	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）

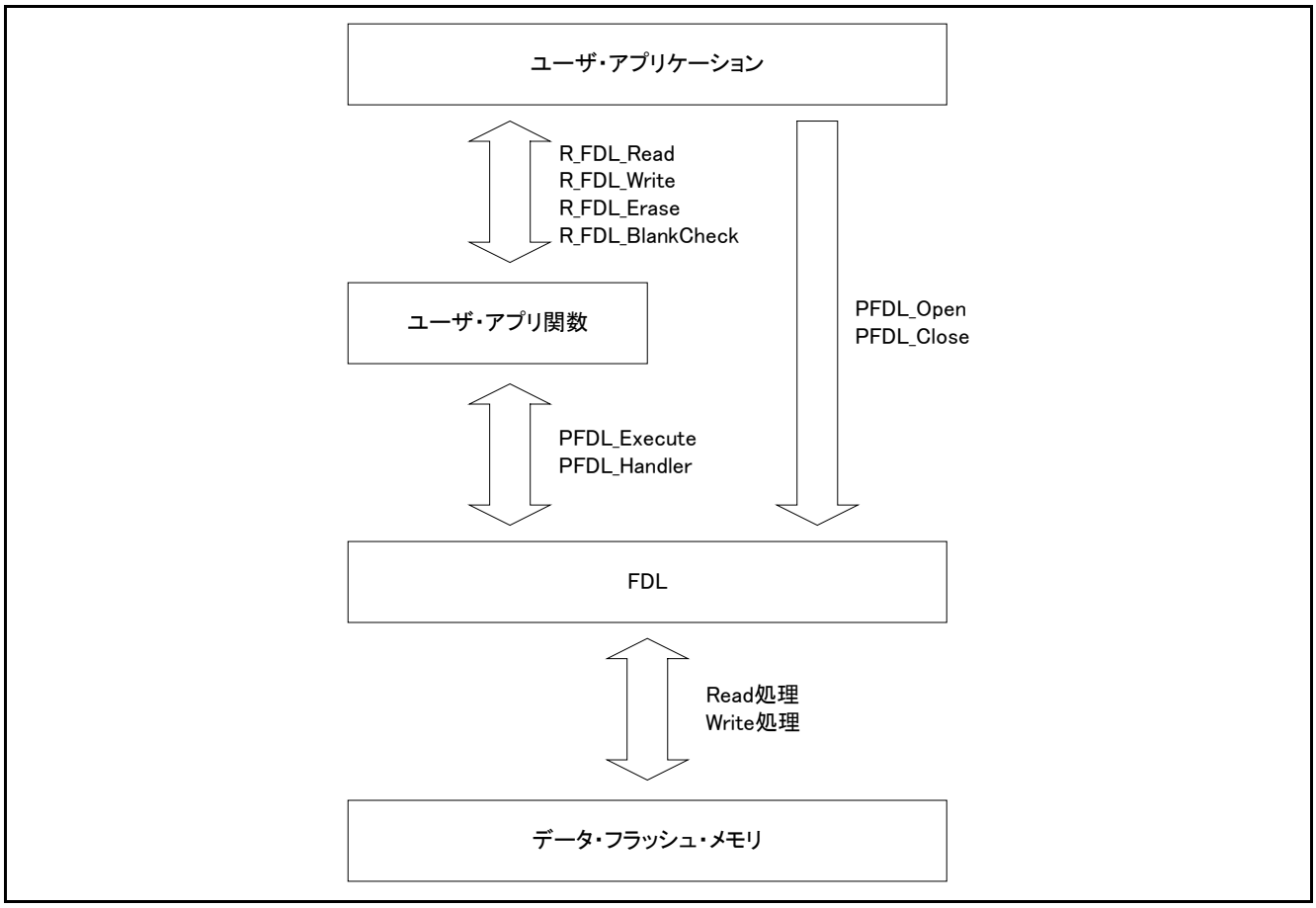


図2.2　アプリケーション全体像

ユーザ・アプリケーションから PFDL_Open 関数、PFDL_Close 関数でデータ・フラッシュ・メモリへのアクセスを許可／禁止にします。データ・フラッシュ・メモリへのアクセスを許可した状態でユーザ・アプリ関数を呼び出すことで、間接的に FDL 関数を実行します。実行した FDL 関数でデータ・フラッシュ・メモリに読み出しや書き込みを行います。

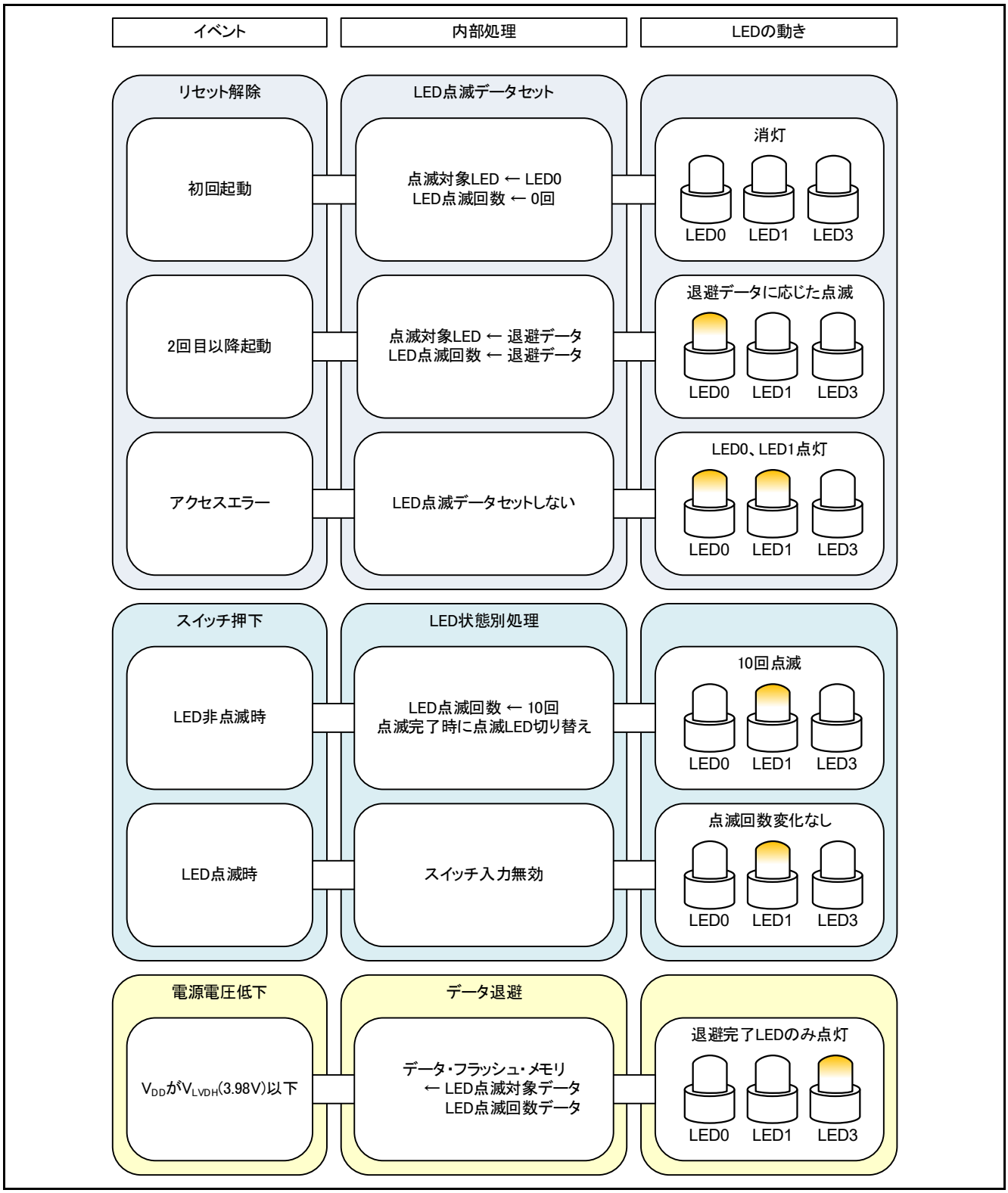

イベント
内部処理
LEDの動き
リセット解除
初回起動
LED点滅データセット
点滅対象LED ← LED0
LED点滅回数 ← 0回
消灯
LED0 LED1 LED3
2回目以降起動
点滅対象LED ← 退避データ
LED点滅回数 ← 退避データ
退避データに応じた点滅
LED0 LED1 LED3
アクセスエラー
LED点滅データセットしない
LED0、LED1点灯
LED0 LED1 LED3
スイッチ押下
LED非点滅時
LED状態別処理
LED点滅回数 ← 10回
点滅完了時に点滅LED切り替え
10回点滅
LED0 LED1 LED3
LED点滅時
スイッチ入力無効
点滅回数変化なし
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　FDL 書き込み時間の短縮

ユーザ・アプリケーションから FDL 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 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。FDL では、アクセスを開始する際に PFDL_Open 関数という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。

また、本アプリケーションでは、データを書き込むアドレスを取得する必要があり、R_FDL_GetWriteAddr 関数を実行することで実現しています。

ただ、電源電圧低下時に PFDL_Open 関数を実行し、R_FDL_GetWriteAddr 関数でアドレスを取得すると、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、FDL 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際のデータ退避処理を図2.4に、処理を分割して行った際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 187.4[μs]、分割処理で 127.6[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 CS+上で FDL RL78 Type04 Ver1.05 を動作させたときの実測値です。

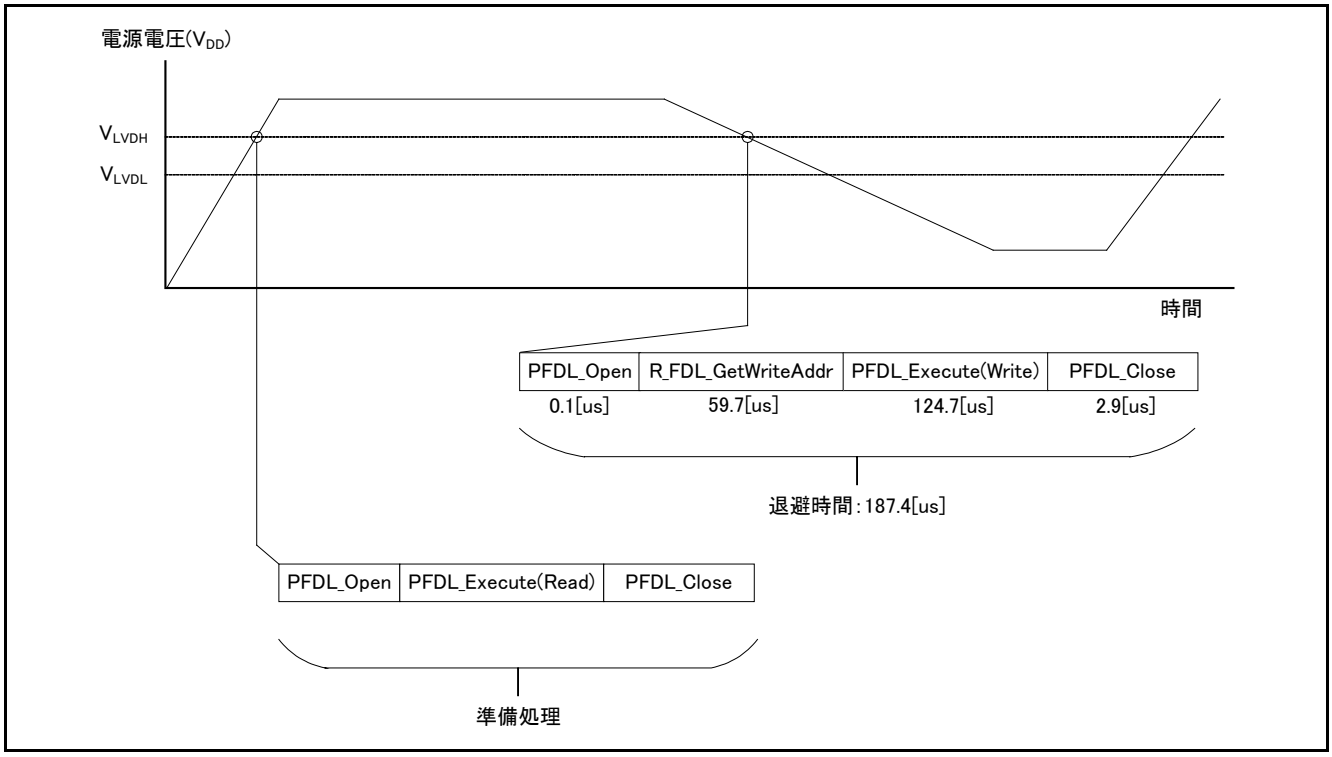


図2.4　データ退避処理（一括）

備考. データ・フラッシュ・メモリからのデータ読み出しは、DFLEN ビットをセットすることでも実現できますが、一例として FDL 関数を用いています。

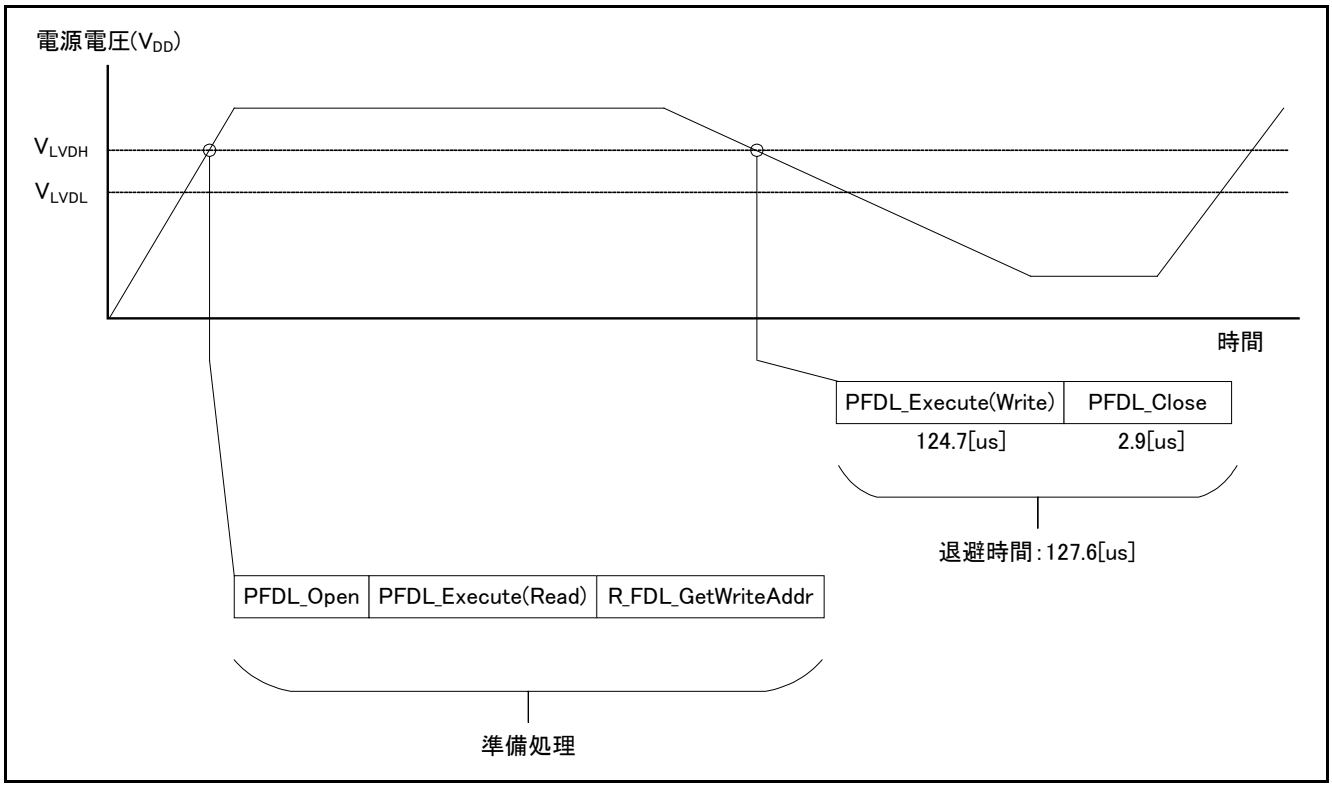


図2.5　データ退避処理（分割）

2.2　データ・フラッシュ・メモリの使い方

本アプリケーションノートではデータ・フラッシュ・メモリを図2.6のように使用します。

データ・フラッシュ・メモリを 1K バイト単位のブロックで管理し、各ブロックの先頭 2 バイトを管理用データ領域として使用します。管理用データはブロック有効フラグとブロック無効フラグから構成され、フラグの状態によってブロックの状態を判別します。表2.2にブロック状態を示します。

データ・フラッシュ・メモリは、ブランク状態（未使用状態）では 0xFF が読み出されるため、ブロック有効フラグとブロック無効フラグが共に 0xFF である場合は、『未使用』ブロックであることを示しています。ブロックの使用を開始するときにブロック有効フラグに 0x00 を書き込み、状態を『有効』に変更します。『有効』状態のブロックに対し、LED 点滅データを書き込んでいき、ブロックの最後に達するとブロック無効フラグに 0x00 を書き込み、『無効』状態に変更します。その後、『未使用』ブロックを検索し、『未使用』ブロックがなければ『無効』ブロックを消去して使用します。

ブロック消去発生時は、準備処理時間が通常時に比べて 5.77[ms]長くなります。

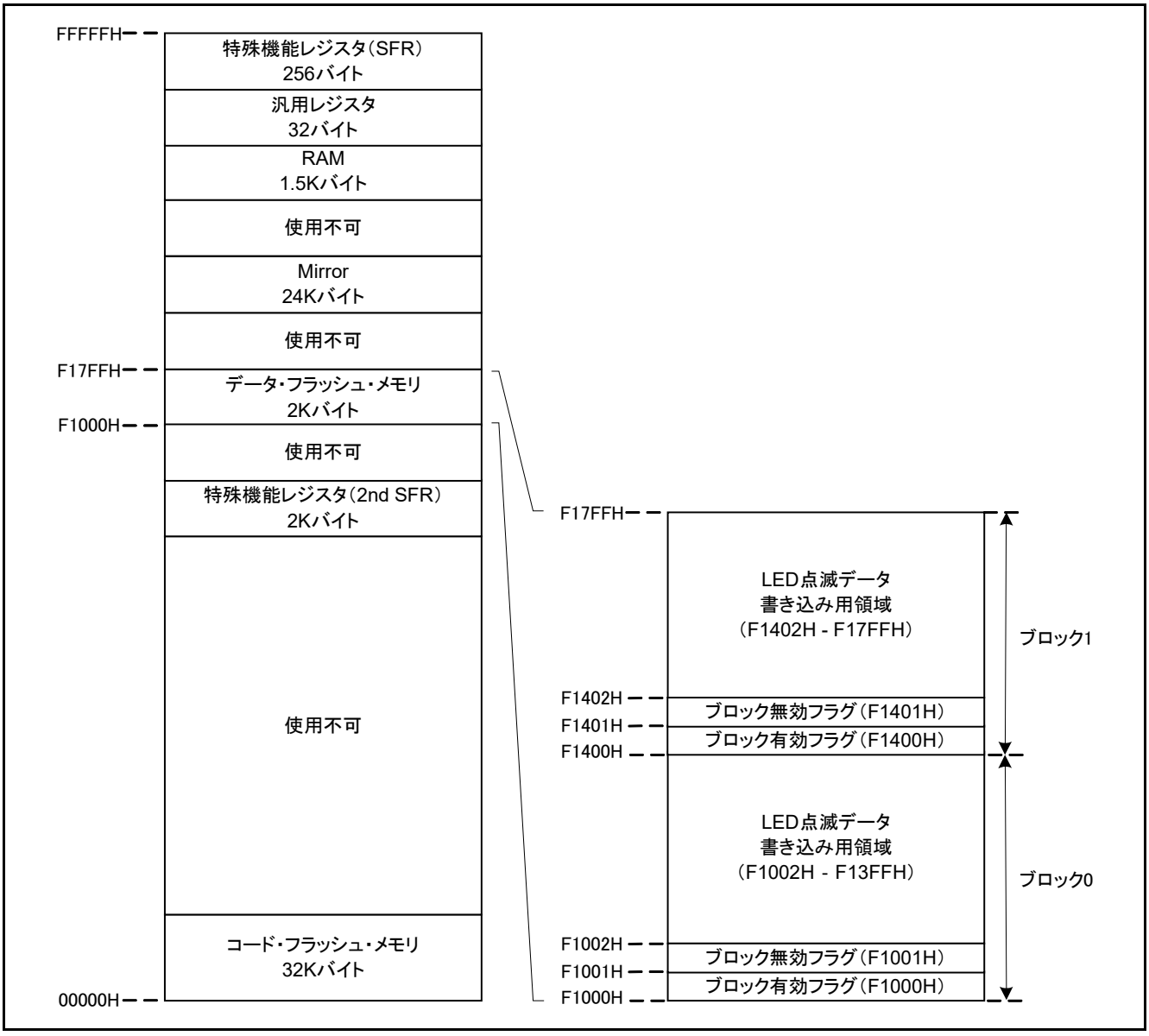


図2.6　データ・フラッシュ・メモリの使い方（RL78/L12(R5F10RLC)の例）

表2.2 ブロック状態

ブロック状態	管理用データ領域	
	ブロック有効フラグ （F1000H:ブロック 0／F1400H:ブロック 1）	ブロック無効フラグ （F1001H：ブロック 0／F1401H：ブロック 1）
未使用	FFH	FFH
有効	00H	FFH
無効	00H	00H

2.3　ブロック内データ検索のアルゴリズム

本アプリケーションノートでのデータ検索のアルゴリズムについて説明します。

データは LED 点滅データ書き込み用領域の先頭から格納されるため、LED 点滅データ書き込み用領域の末尾から格納されるデータ・サイズ（2 バイト）単位でブランクチェックを行い、初めてブランク状態でない（書き込み不可能）と判定されたアドレスに最新データが格納されていると判定しています。

また、最新データが書かれているアドレスがブロックの末尾でない場合は、最新データが書かれているアドレスの次のアドレスがブランク状態（書き込み可能）であるため、容易に書き込みアドレスを決定できます。最新データが書かれているアドレスがブロックの末尾である場合は、ブロックチェンジが発生するため、LED 点滅データ書き込み用領域の先頭に書き込みアドレスを設定します。

注意　データ・フラッシュ・メモリに書き込めるかどうかの判断はブランクチェックで行ってください。読み出し値が FFH であっても消去状態とは判断できません。

上書きを行うと、データ・フラッシュ・メモリにダメージを与えることがあります。書き込みを行う場合は、必ず、ブランクチェックで書き込み可能であることを確認してください。

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/L12(R5F10RLC)
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)
統合開発環境 (CS+)	ルネサス エレクトロニクス製 CS+ for CC V8.07.00
C コンパイラ(CS+)	ルネサス エレクトロニクス製 CC-RL V1.11.00
統合開発環境 (e^2 studio)	ルネサス エレクトロニクス製 e^2 studio V2022-01 (22.1.0)
C コンパイラ (e^2 studio)	ルネサス エレクトロニクス製 CC-RL V1.11.00
統合開発環境 (IAR)	IAR Systems 製 IAR Embedded Workbench for Renesas RL78 V4.21.3
C コンパイラ (IAR)	IAR Systems 製 IAR C/C++ Compiler for Renesas RL78 V4.21.3.2447
FDL	FDL RL78 Type04 Ver1.05 注 Type04 パッケージ Ver.2.00
使用ボード	RL78/L12 ターゲット・ボード (QB-R5F10RLC-TB)

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。

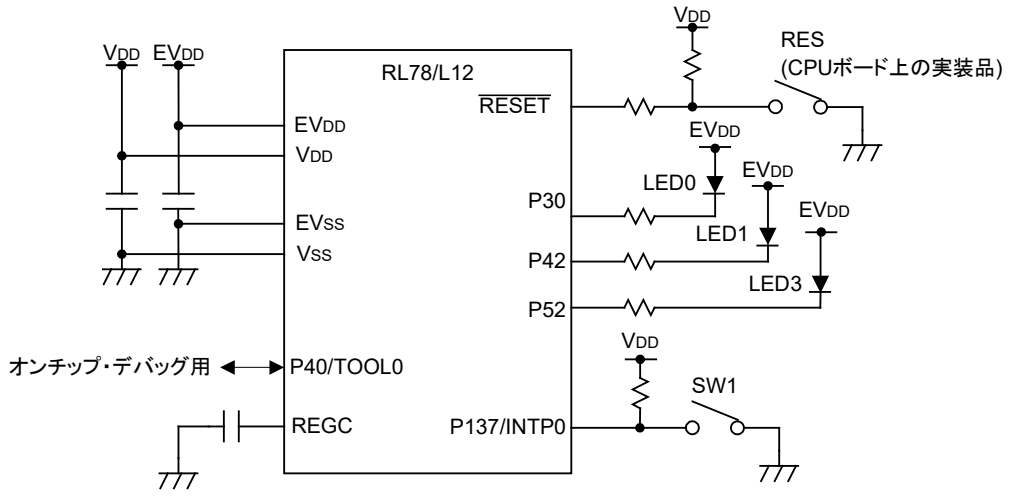
RL78 ファミリ データ・フラッシュ・ライブラリ Type04（R01US0049J）ユーザーズ・マニュアル

Data Flash Access Library (Type T04 (Pico), European Release) (R01US0055ED0120) 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で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
P30	出力	LED 点灯（LED0）制御ポート
P42	出力	LED 点灯（LED1）制御ポート
P52	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 FDL 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。その後、データ退避先アドレスを取得します。

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。

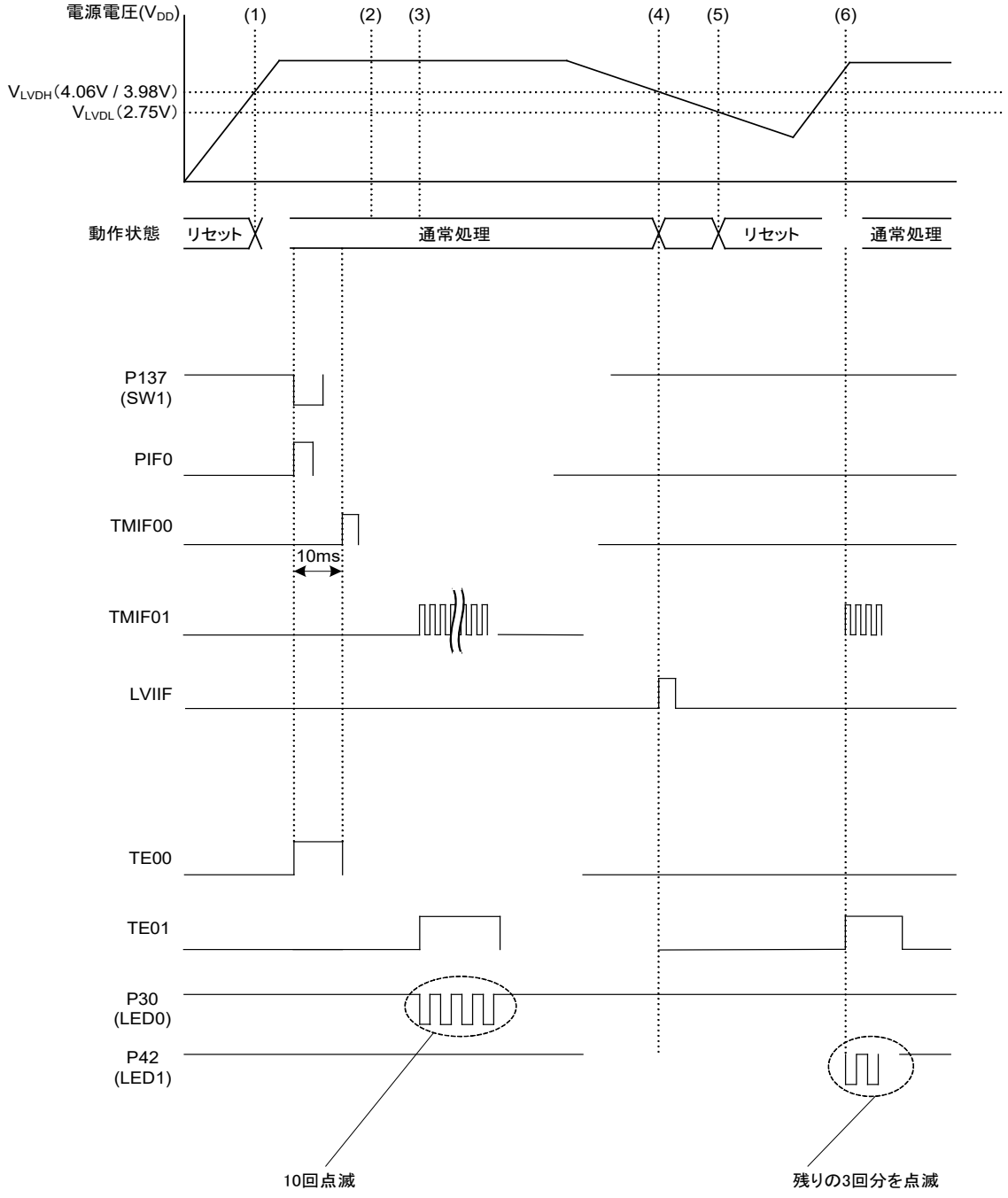
電源電圧の低下は LVD 機能で検出します。電源電圧の低下を検知すると、FDL を用いて LED 点滅状態のデータ（点滅対象 LED と点滅回数）をデータ・フラッシュ・メモリに退避し、データの退避完了を示す LED3 を点灯させて STOP モードに入ります。なお、データには終端記号として 0x00 が付加されます。

また、FDL 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。

1. 入出力ポートを設定します。
   - LED 点灯制御（LED0、LED1、LED3）：P30、P42、P52 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 で使用する RAM の初期化と開始をします。具体的には、PFDL_Open 関数をコールします。
3. データ・フラッシュ・メモリから有効ブロックを探します。
   - 有効ブロックは各ブロックの管理領域（先頭 2 バイト）が 00H、FFH のブロックです。
     読み出しには PFDL_Execute(Read)関数を用います。
   - 有効ブロックがない場合は、データ・フラッシュ・メモリの先頭ブロックを消去して、
     有効ブロックとして扱います。消去には、PFDL_Execute(Erase)関数を用います。
4. 最新の LED 点滅データを読み出して、データに応じて対象の LED を 500ms 間隔で点滅させます。
   - 最新のデータは有効ブロックを末尾から 2 バイトずつブランクチェックを行い、初めてブランク状態でないと判定されたメモリアドレスの偶数番地に書かれています。ブランクチェックには PFDL_Execute(Blankcheck)関数を、読み出しには PFDL_Execute(Read)関数を用います。
   - 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。
   - 読み出したデータに応じた点滅を開始します。
5. 電源電圧低下時の書き込みアドレスを取得します。
   - 読み出しアドレスから次回の書き込みアドレスを取得します。具体的には R_FDL_GetWriteAddr 関数をコールします。
   - 読み出しアドレスと書き込みアドレスのブロック番号が異なる場合は、ブロックチェンジが発生します。
   - 書き込みアドレスを取得した後は、スイッチの押下待ち状態となります。
6. 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。

7. 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 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。
   PFDL_Execute(Write)、PFDL_Close
8. FDL 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 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、PFDL_Close 関数をコールしたあとに STOP 命令を実行します。
9. 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FDL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。
退避データの書き込み先を取得します。
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、点滅用の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 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、電源電圧低下時にデータを書き込むアドレスを取得します。

(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）とデータの終端文字（00H）をデータ・フラッシュ・メモリに書き込んで 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（SW 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データに応じた 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_fdl_function.c	データ退避用関数のソースファイル	追加関数： R_FDL_Read R_FDL_Write R_FDL_Erase R_FDL_BlankCheck R_FDL_EnableBlock R_FDL_DisableBlock R_FDL_ChangeBlock R_FDL_ReadManageData R_FDL_ReadLedData R_FDL_SearchEnableBlock R_FDL_SearchReadAddr R_FDL_CheckDataRange R_FDL_GetWriteAddr R_FDL_WriteLedData
r_fdl_function.h	データ退避用関数のヘッダファイル	-
pfdl.h 注1	FDL のヘッダファイル	各コンパイラ共通
pfdl_types.h 注1	FDL 型定義ヘッダファイル	各コンパイラ共通
pfdl.lib 注1	FDL	CS+版、e2studio 版
pfdl.a 注1	FDL	IAR 版
r_fdl.dr 注2	リンク・ディレクティブ・ファイル	CS+版、e2studio 版
trio_lnkR5F10RLC.icf 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 FDL 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、表6.4に定数を示します。

表6.3　定数(1/2)

定数名	設定値	内容
BLOCK_NUM	0x0002	データ・フラッシュ・メモリのブロック数
RET_OK	0x00	正常応答
RET_NG	0x01	異常応答
RET_NG_DEVICE	0x02	FDL アクセスエラー
RET_BLOCK_UNUSED	0x03	未使用ブロック
RET_BLOCK_ENABLE	0x04	有効ブロック
RET_BLOCK_DISABLE	0x05	無効ブロック
RET_BLOCK_UNEXPECTED	0x06	異常ブロック
RET_CHECK_BLANK	0x07	ブランク状態
RET_CHECK_FILL	0x08	書き込み済み状態
STA_ADDR_SEARCH	0x10	格納アドレス検索中のステータス
STA_ADDR_FOUND	0x11	格納アドレス発見のステータス
STA_ADDR_NOTFOUND	0x12	格納アドレスなしのステータス
STA_ACCESS_ERROR	0x13	FDL アクセスエラー
SHIFT_NUM	0x04	LED 点滅データ用ビットシフト数
BLINK_LED_MAX	0x01	LED 点滅番号最大値
BLINK_NUM_MAX	0x0A	LED 点滅回数最大値
BLOCK_SIZE	0x0400	1 ブロックのサイズ
ENABLE_AREA_SIZE	0x0001	ブロック有効フラグのサイズ
DISABLE_AREA_SIZE	0x0001	ブロック無効フラグのサイズ
MANAGE_AREA_SIZE	0x0002	管理領域のサイズ
BLANK_DATA	0xFF	ブランク状態のデータ
ENABLE_DATA	0x00	ブロック有効状態のデータ
DISABLE_DATA	0x00	ブロック無効状態のデータ
LED_DATA_SIZE	0x02	LED 点滅データのサイズ
TERMINAL_SYMBOL	0x00	LED 点滅データの終端記号
ENABLE_AREA_ADDR	0x0000	有効ブロック確認領域のアドレス
DISABLE_AREA_ADDR	0x0001	無効ブロック確認領域のアドレス
FIRST_DATA_ADDR	0x0002	LED 点滅データ格納領域の先頭格納アドレス
LAST_DATA_ADDR	0x03FE	LED 点滅データ格納領域の最終格納アドレス

表6.4　定数(2/2)

定数名	設定値	内容
BLINK_LED0	0x00	点滅対象 LED0
BLINK_LED1	0x01	点滅対象 LED1
LED0	P3_bit.no0	LED0 制御ポート
LED1	P4_bit.no2	LED1 制御ポート
LED3	P5_bit.no2	LED3 制御ポート
LED_ON	0	LED 点灯レベル
LED_OFF	1	LED 消灯レベル
SW1	P13_bit.no7	SW1 制御ポート
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6.5にグローバル変数を示します。

表6.5　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6.6に関数を示します。

表6.6　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_FDL_SearchEnableBlock	有効ブロック検索
R_FDL_SearchReadAddr	読み出しアドレス検索
R_FDL_EnableBlock	ブロック有効化
R_FDL_ReadLedData	LED 点滅データ読み出し
R_FDL_CheckDataRange	LED 点滅データ有効範囲チェック
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	INTP 割り込み許可
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_FDL_GetWriteAddr	書き込みアドレス取得
R_FDL_WriteLedData	LED 点滅データ書き込み
R_INTC0_Stop	INTP0 動作禁止設定
R_TAU0_Channel0_Stop	TAU00 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
R_FDL_ReadManageData	管理データ読み出し
R_FDL_Read	FDL 読み出し
R_FDL_BlankCheck	FDL ブランクチェック
R_FDL_Write	FDL 書き込み
R_FDL_ChangeBlock	ブロックチェンジ
R_FDL_DisableBlock	ブロック無効化
R_FDL_Erase	ブロック消去
PFDL_Open	FDL の開始
PFDL_Close	FDL 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_fdl_function.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_fdl_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_FDL_SearchEnableBlock

概　要	有効ブロック検索
ヘッダ	r_fdl_function.h
宣　言	uint8_t R_FDL_SearchEnableBlock(uint16_t* pblock)
説　明	有効ブロックの番号を取得します。
引　数	uint16_t* pblock　　有効ブロック番号のポインタ
リターン値	・有効ブロック発見　　：RET_OK ・有効ブロックなし　　：RET_NG ・FDL アクセスエラー　　：RET_NG_DEVICE

R_FDL_SearchReadAddr

概　要	読み出しアドレス検索	
ヘッダ	r_fdl_function.h	
宣　言	uint8_t R_FDL_SearchReadAddr(uint16_t block, uint16_t* paddr)	
説　明	指定したブロック内でデータが格納されているアドレスを検索します。 データが格納されていた場合、そのアドレスを取得します。 データが格納されていない（ブロック全体がブランク）場合、データ領域の先頭アドレスを取得します。	
引　数	uint16_t block	有効ブロック番号
	uint16_t* paddr	格納アドレスのポインタ
リターン値	・アドレス取得	: RET_OK
	・FDL アクセスエラー	: RET_NG_DEVICE

R_FDL_EnableBlock

概　要	ブロック有効化	
ヘッダ	r_fdl_function.h	
宣　言	uint8_t R_FDL_EnableBlock(uint16_t block)	
説　明	指定した未使用ブロックを有効化します。	
引　数	uint16_t block	有効化するブロック番号
リターン値	・ブロック有効化成功	: RET_OK
	・FDL アクセスエラー	: RET_NG_DEVICE

R_FDL_ReadLedData

概　要	LED 点滅データ読み出し	
ヘッダ	r_fdl_function.h	
宣　言	uint8_t R_FDL_ReadLedData(uint16_t addr, uint8_t* pdata)	
説　明	指定のアドレスから LED の点滅データを読み出します。	
引　数	uint16_t addr	読み出しアドレス
	uint8_t* pdata	読み出しデータ格納バッファのポインタ
リターン値	・読み出し成功	: RET_OK
	・FDL アクセスエラー	: RET_NG_DEVICE

R_FDL_CheckDataRange

概　要	LED 点滅データ有効範囲チェック	
ヘッダ	r_fdl_function.h	
宣　言	uint8_t R_FDL_CheckDataRange(uint8_t* pdata)	
説　明	LED 点滅データが有効なデータかをチェックします。	
引　数	uint8_t* pdata	LED 点滅データのポインタ
リターン値	・有効範囲内	: RET_OK
	・有効範囲外	: RET_NG

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_fdl_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_fdl_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_FDL_GetWriteAddr

概　要	書き込みアドレス取得	
ヘッダ	r_fdl_function.h	
宣　言	uint8_t R_FDL_GetWriteAddr(uint16_t* pblock , uint16_t* paddr)	
説　明	引数に与えた読み出しアドレスより書き込みアドレスを計算します。 計算中にブロックチェンジが発生した場合は、ブロック番号が更新されます。	
引　数	uint16_t* pblock	読み出し／書き込みブロック番号のポインタ
	uint16_t* paddr	読み出し／書き込みアドレスのポインタ
リターン値	・アドレス取得成功	: RET_OK
	・FDL アクセスエラー	: RET_NG_DEVICE

R_FDL_WriteLedData

概　要	LED 点滅データ書き込み	
ヘッダ	r_fdl_function.h	
宣　言	uint8_t R_FDL_WriteLedData(uint16_t addr, uint8_t data)	
説　明	LED 点滅データを指定のアドレスに書き込みます。	
引　数	uint16_t addr	書き込み先アドレス
	uint8_t data	書き込みデータ
リターン値	・データ書き込み成功	: RET_OK
	・FDL アクセスエラー	: RET_NG_DEVICE

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_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_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_fdl_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

R_FDL_ReadManageData

概　要	管理データ読み出し	
ヘッダ	r_fdl_function.h	
宣　言	uint8_t R_FDL_ReadManageData(uint16_t block)	
説　明	指定ブロックの管理領域(先頭 2byte)のデータを読み出し、ブロックの状態を取得します。	
引　数	uint16_t block	読み出しブロック番号
リターン値	・未使用ブロック	: RET_BLOCK_UNUSED
	・有効ブロック	: RET_BLOCK_ENABLE
	・無効ブロック	: RET_BLOCK_DISABLE
	・異常ブロック	: RET_BLOCK_UNEXPECTED
	・FDL アクセスエラー	: RET_NG_DEVICE

R_FDL_Read

概　要	FDL 読み出し	
ヘッダ	r_fdl_function.h	
宣　言	uint8_t R_FDL_Read(uint16_t addr, uint8_t* pdata, uint16_t size)	
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	uint16_t addr	読み出し先頭アドレス
	uint8_t* pdata	読み出しデータ格納バッファのポインタ
	uint16_t size	読み出しデータ・サイズ
リターン値	・読み出し成功	: RET_OK
	・FDL アクセスエラー	: RET_NG_DEVICE

R_FDL_BlankCheck

概　要	FDL ブランクチェック	
ヘッダ	r_fdl_function.h	
宣　言	uint8_t R_FDL_BlankCheck(uint16_t addr, uint16_t size)	
説　明	指定した範囲がブランク状態かをチェックします。	
引　数	uint16_t addr	チェック開始アドレス
	uint16_t size	チェックバイト数
リターン値	・書き込み状態	: RET_CHECK_FILL
	・ブランク状態	: RET_CHECK_BLANK
	・FDL アクセスエラー	: RET_NG_DEVICE

R_FDL_Write

概　要	FDL 書き込み	
ヘッダ	r_fdl_function.h	
宣　言	uint8_t R_FDL_Write(uint16_t addr, uint8_t* pdata, uint16_t size)	
説　明	指定したアドレスにデータを書き込みます。	
引　数	uint16_t addr	書き込み先アドレス
	uint8_t* pdata	書き込みデータ
	uint16_t size	書き込みデータ数
リターン値	・書き込み成功	: RET_OK
	・FDL アクセスエラー	: RET_NG_DEVICE

R_FDL_ChangeBlock

概　要	ブロックチェンジ	
ヘッダ	r_fdl_function.h	
宣　言	uint8_t R_FDL_ChangeBlock(uint16_t* pblock)	
説　明	引数で与えたブロックを無効にし、新しいブロックを有効にします。 本関数実行後には引数 pblock にチェンジ後ブロック番号が入ります。	
引　数	uint16_t* pblock	[IN]　チェンジ前ブロック番号 [OUT]チェンジ後ブロック番号
リターン値	・チェンジ成功	: RET_OK
	・チェンジ失敗	: RET_NG
	・FDL アクセスエラー	: RET_NG_DEVICE

R_FDL_DisableBlock

概　要	ブロック無効化	
ヘッダ	r_fdl_function.h	
宣　言	uint8_t R_FDL_DisableBlock(uint16_t block)	
説　明	指定したブロックを無効化します。	
引　数	uint16_t block	無効化するブロック番号
リターン値	・ブロック無効化成功	: RET_OK
	・FDL アクセスエラー	: RET_NG_DEVICE

R_FDL_Erase

概　要	FDL ブロック消去	
ヘッダ	r_fdl_function.h	
宣　言	uint8_t R_FDL_Erase(uint16_t block)	
説　明	指定したブロックを消去します。	
引　数	uint16_t block	消去ブロック
リターン値	・消去成功	: RET_OK
	・FDL アクセスエラー	: RET_NG_DEVICE

PFDL_Open

概　要	FDL 開始	
ヘッダ	pfdl.h	
宣　言	pfdl_status_t __far PFDL_Open(__near pfdl_descriptor_t* descriptor_pstr)	
説　明	FDL で使用する RAM の初期化と開始をします。 ※FDL のライブラリ関数です。	
引　数	__near pfdl_descriptor_t* descriptor_pstr	FDL の初期設定値
リターン値	・正常終了	: PFDL_OK

PFDL_Close

概　要	FDL 終了
ヘッダ	pfdl.h
宣　言	void PFDL_Close(void)
説　明	FDL を終了します。 ※FDL のライブラリ関数です。
引　数	なし
リターン値	なし

PFDL_Execute

概　要	データ・フラッシュ・メモリへの制御実行	
ヘッダ	pfdl.h	
宣　言	pfdl_status_t __far PFDL_Execute(__near pfdl_request_t* request_pstr)	
説　明	コマンドに応じ、データ・フラッシュ・メモリへの制御を実行します。 ※FDL のライブラリ関数です。	
引　数	__near pfdl_request_t* request_pstr	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）
リターン値	・正常終了	：PFDL_OK
	・消去エラー	：PFDL_ERR_ERASE
	・ブランク・チェック・エラー or 内部ベリファイ・エラー	：PFDL_ERR_MARGIN
	・書き込みエラー	：PFDL_ERR_WRITE
	・指定したコマンドの実行開始	：PFDL_BUSY

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）	
ヘッダ	pfdl.h	
宣　言	pfdl_status_t __far PFDL_Handler(void)	
説　明	直前に実行した PFDL_Execute 関数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。 ※FDL のライブラリ関数です。	
引　数	なし	
リターン値	・正常終了	：PFDL_OK
	・消去エラー	：PFDL_ERR_ERASE
	・ブランク・チェック・エラー or 内部ベリファイ・エラー	：PFDL_ERR_MARGIN
	・書き込みエラー	：PFDL_ERR_WRITE
	・アイドル状態	：PFDL_IDLE
	・コマンド実行中	：PFDL_BUSY

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。

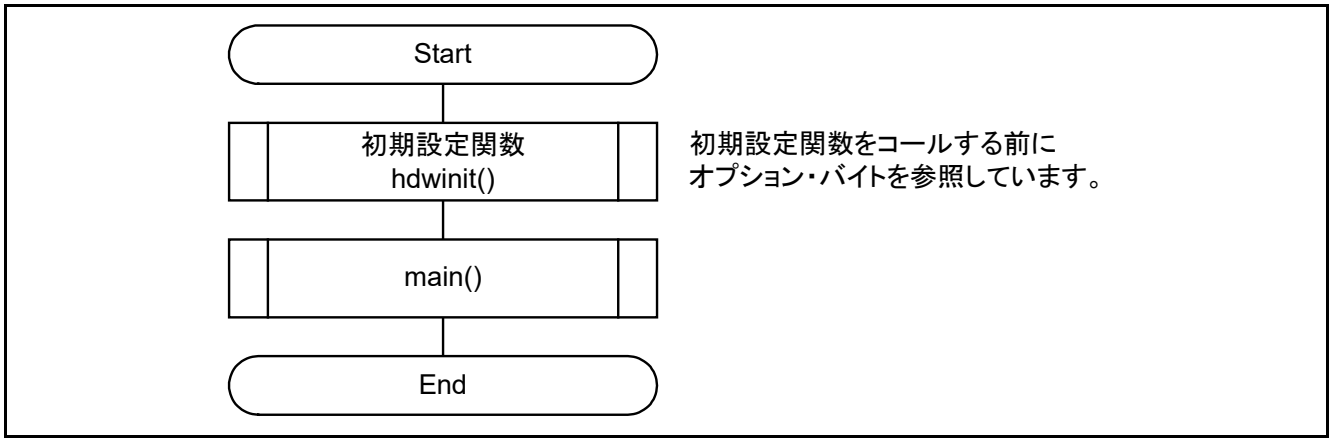


図 6.2 全体フローチャート

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。

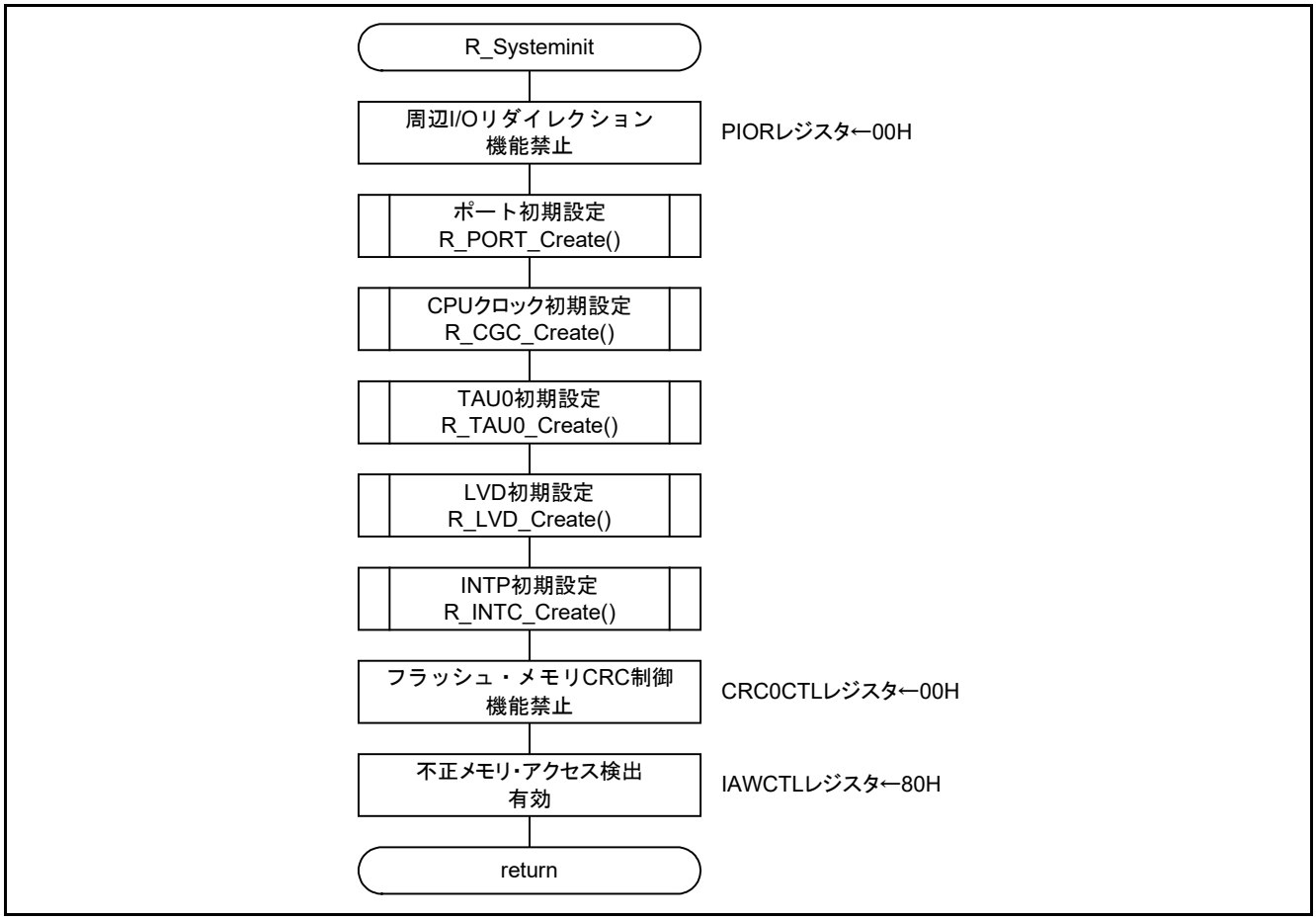


図 6.3 周辺機能初期設定

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。

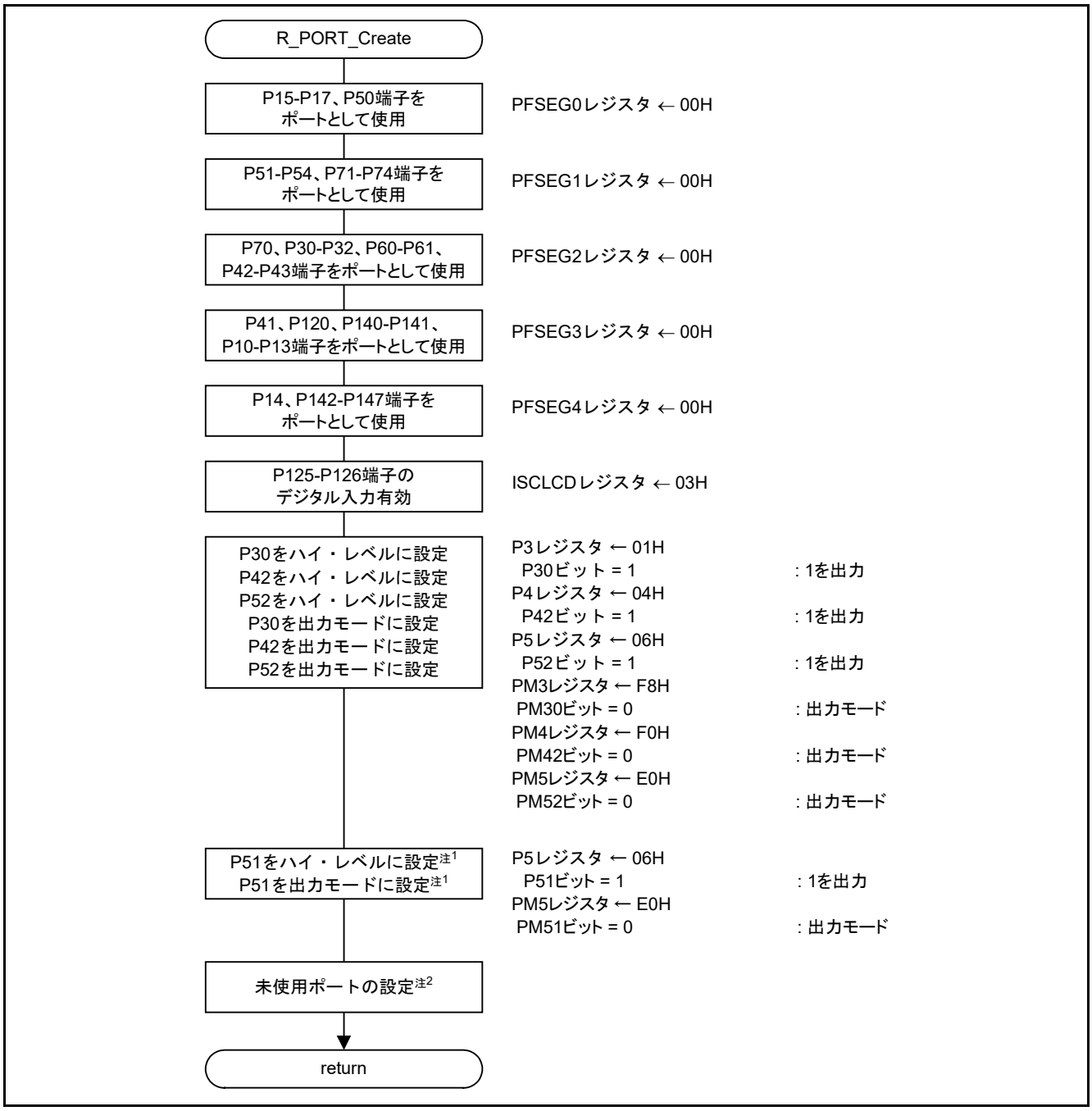


図 6.4 ポート初期設定

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/L12 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。

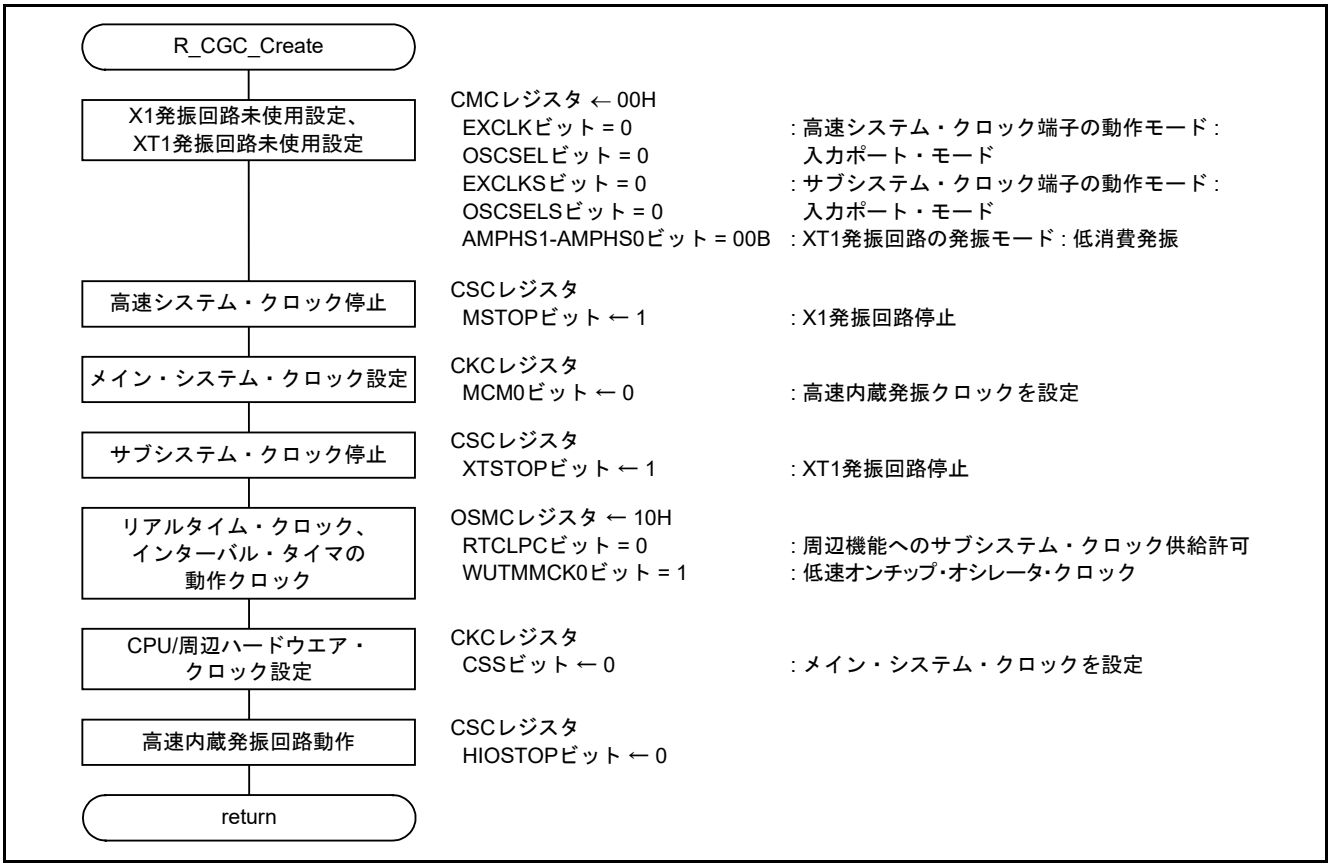


図 6.5 CPU クロック初期設定

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。

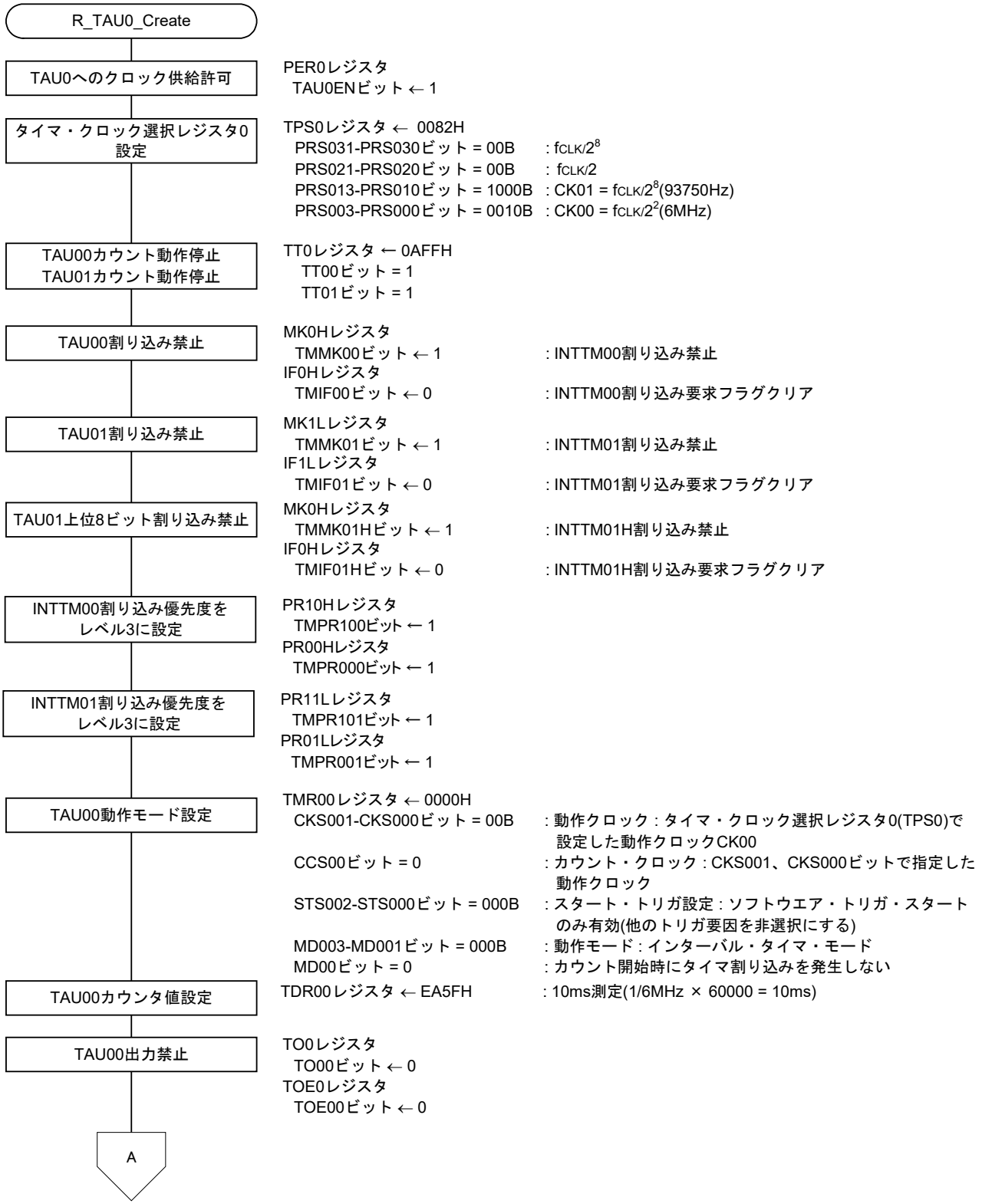


図 6.6 TAU0 初期設定(1/2)

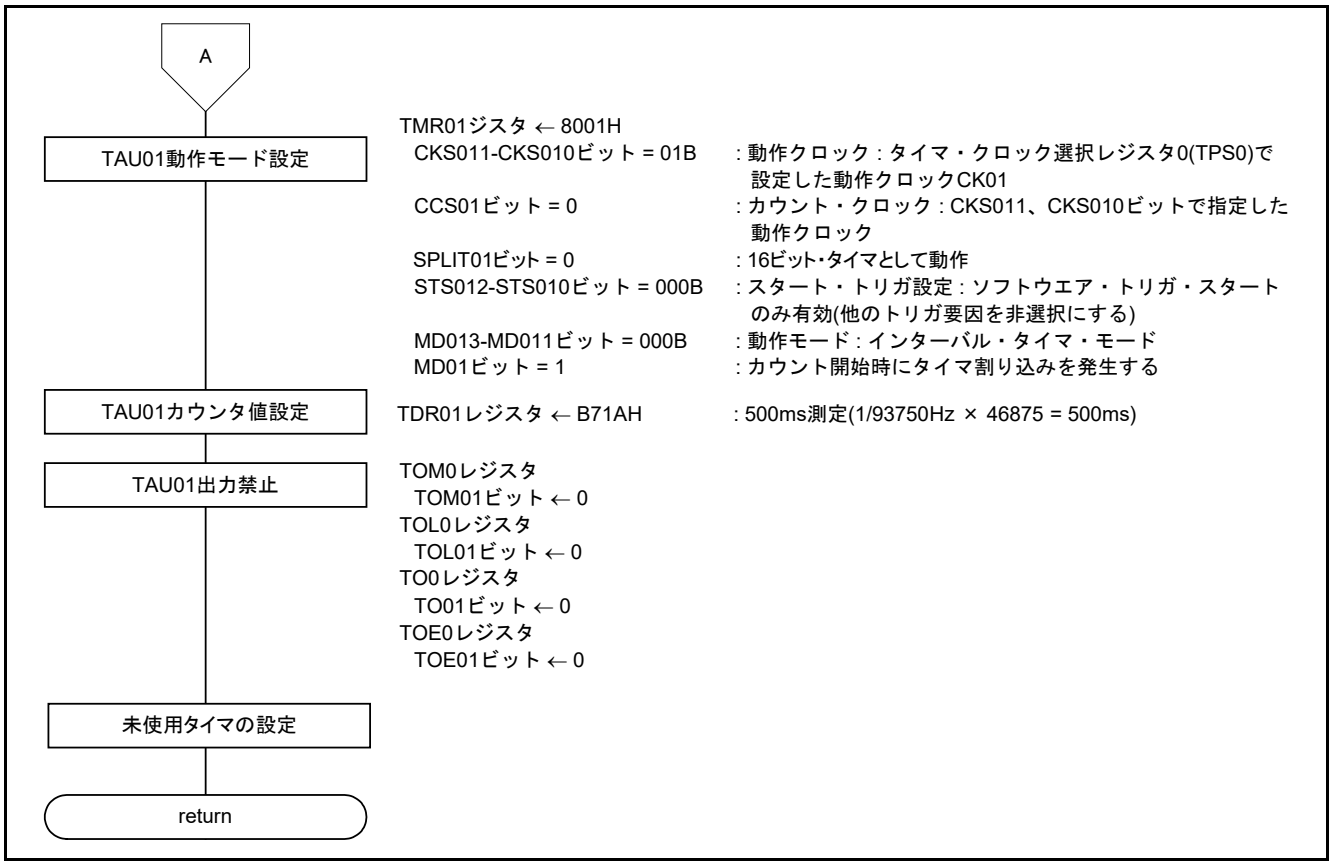


図 6.7 TAU0 初期設定(2/2)

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。

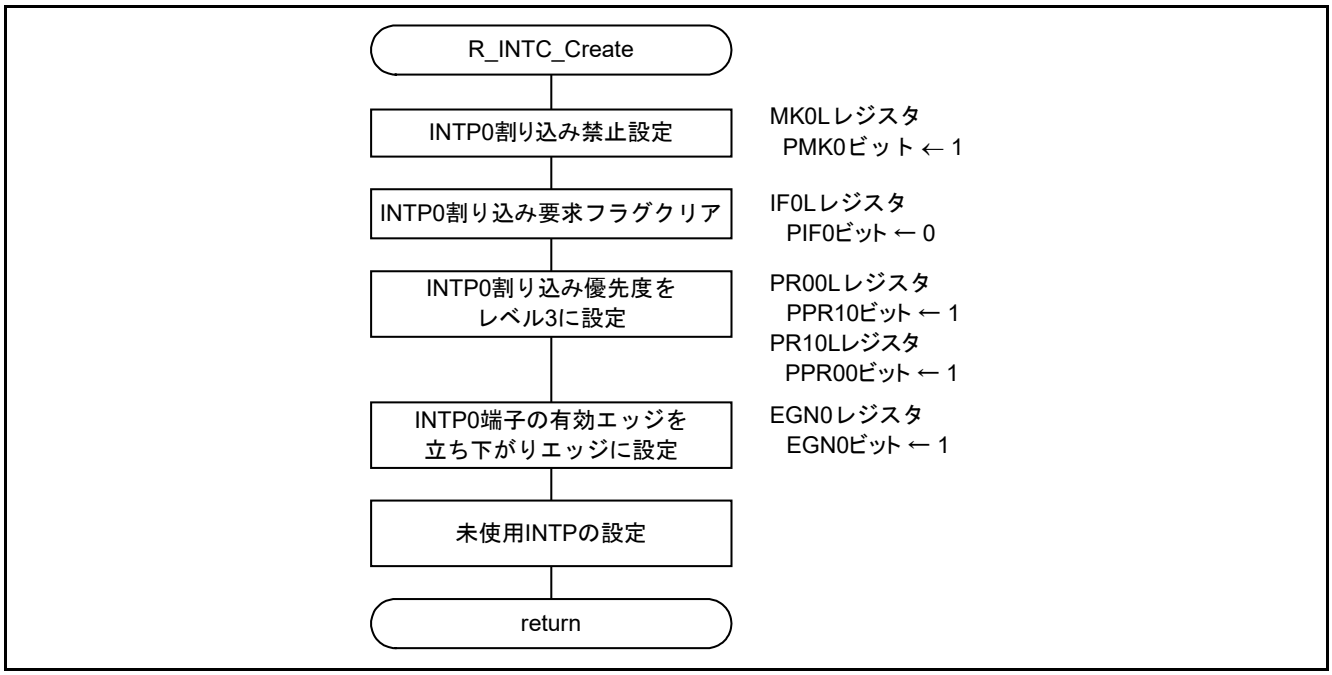


図 6.8 INTP 初期設定

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。

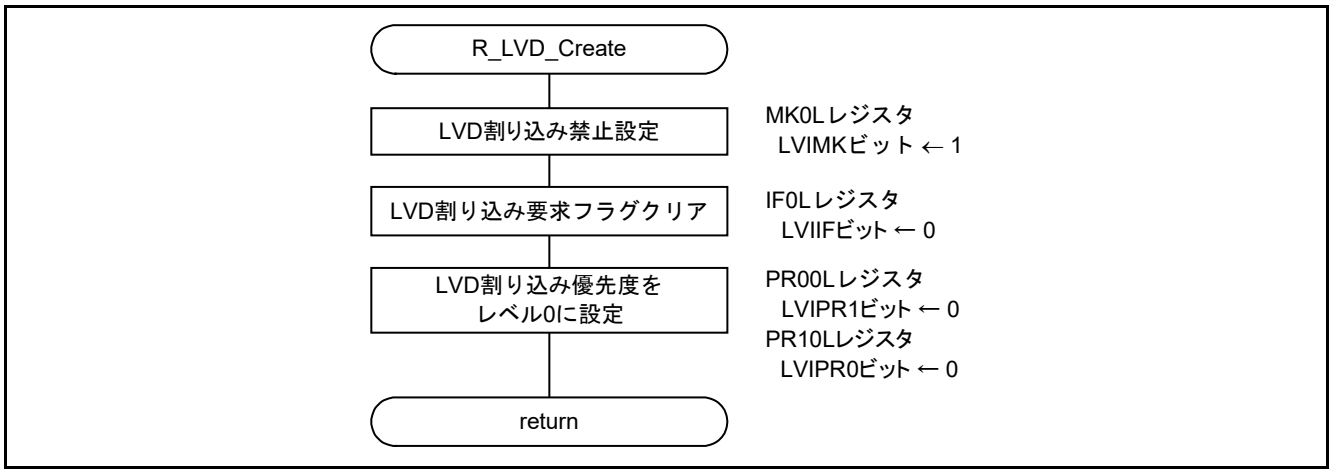


図 6.9 LVD 初期設定

6.8.8　　メイン処理

図 6.10、図 6.11、図 6.12にメイン処理のフローチャートを示します。

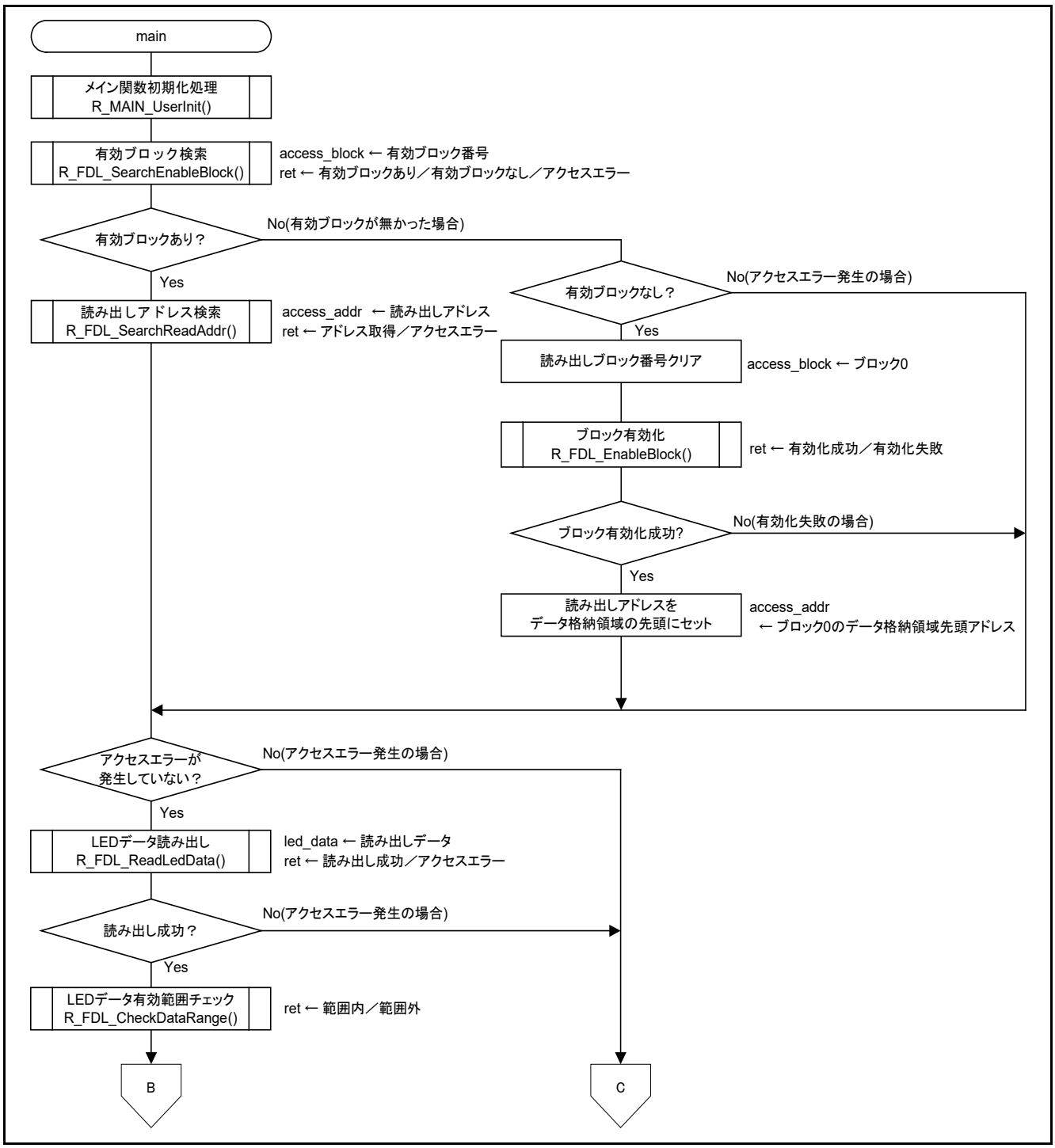


図 6.10 メイン処理(1/3)

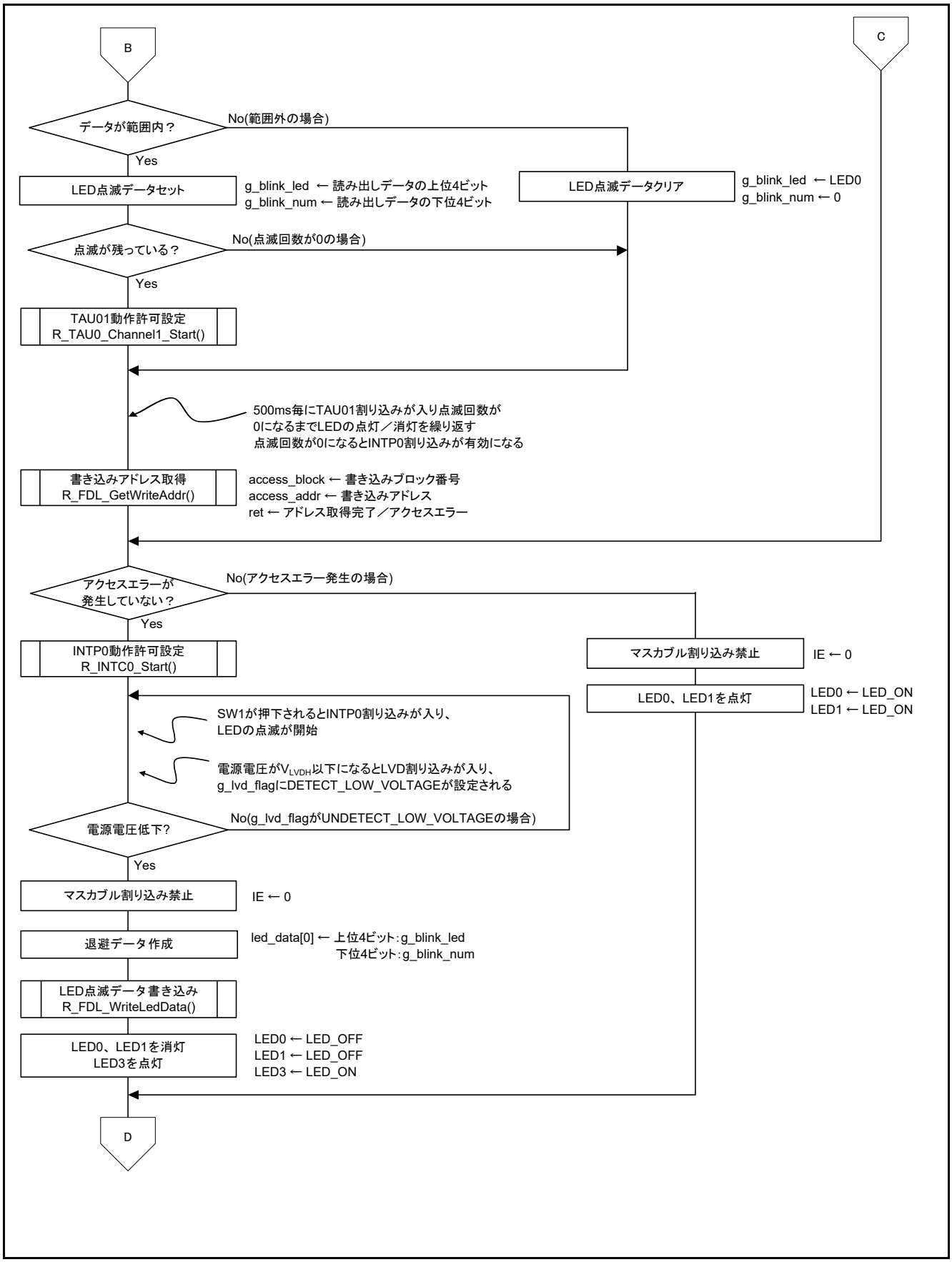


図 6.11 メイン処理(2/3)

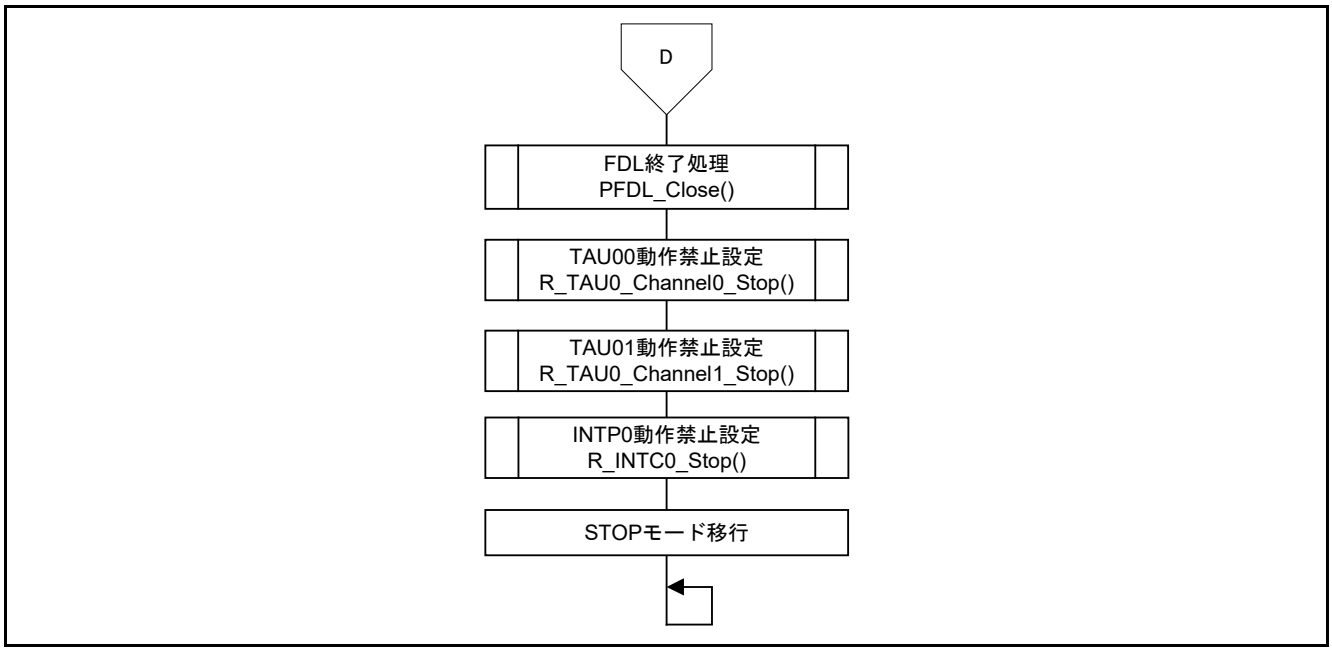


図 6.12 メイン処理(3/3)

6.8.9　　メイン初期化処理

図 6.13にメイン初期化処理のフローチャートを示します。

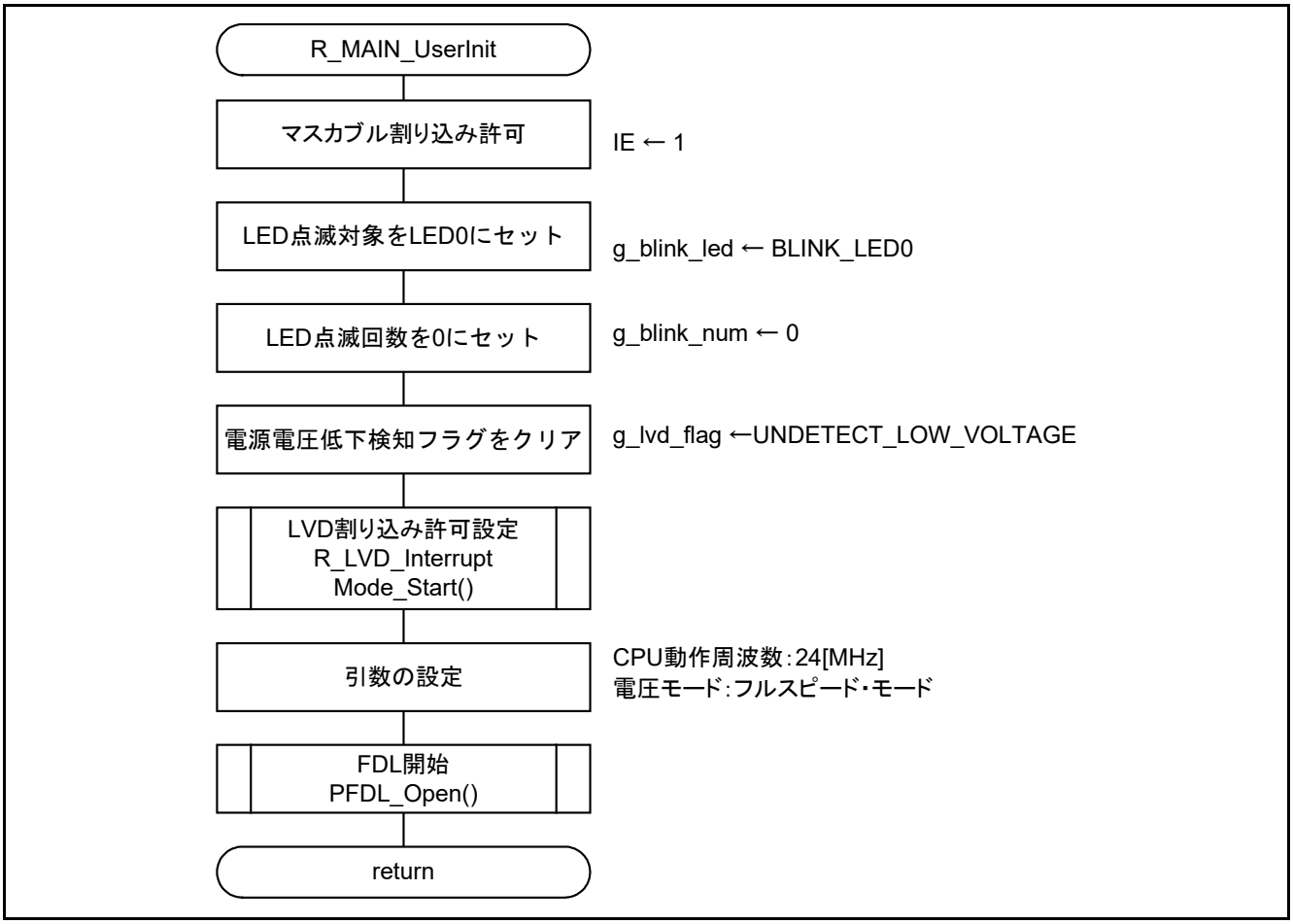


図 6.13 メイン初期化処理

6.8.10　有効ブロック検索

図 6.14に有効ブロック検索のフローチャートを示します。

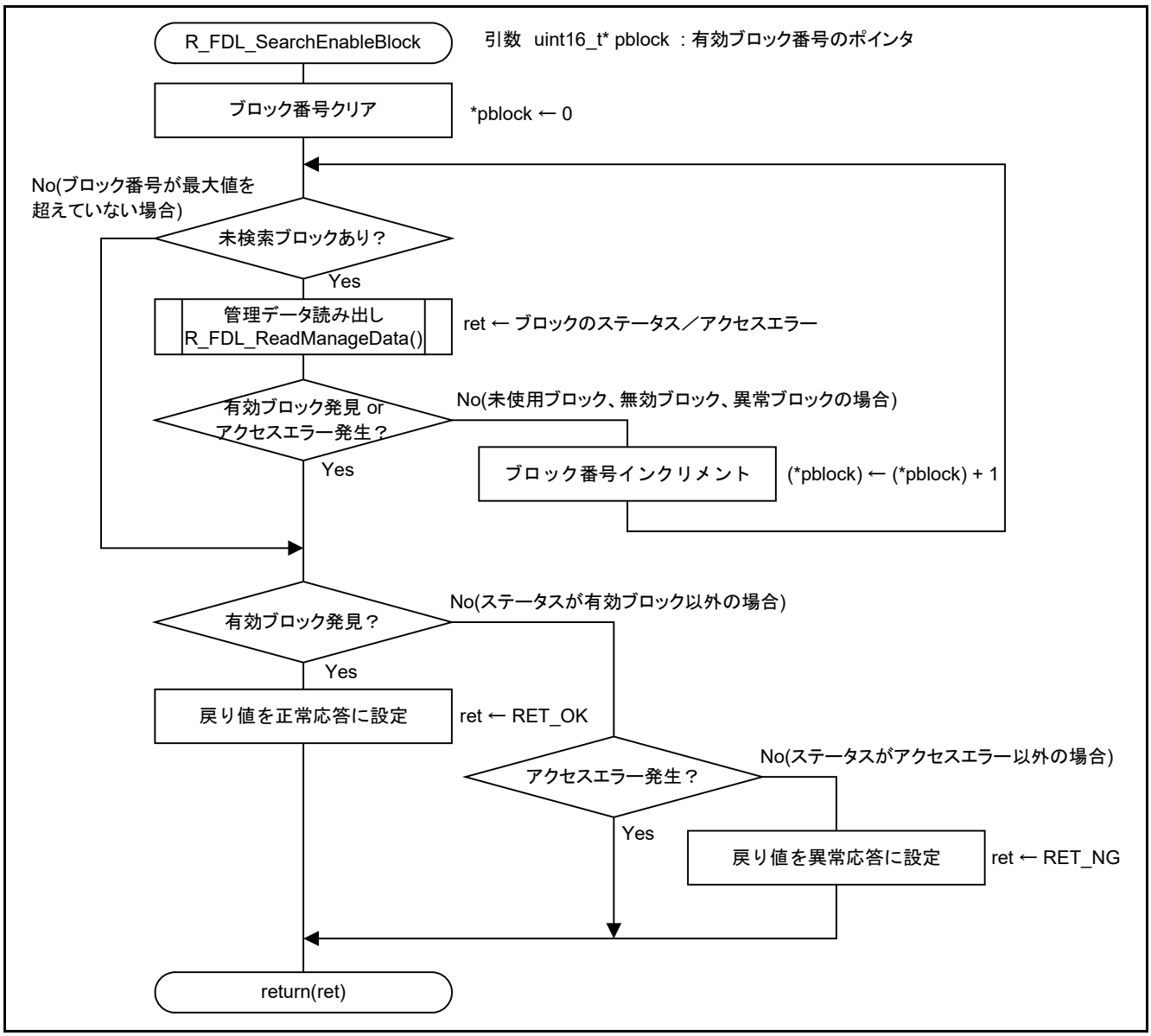


図 6.14 有効ブロック検索

6.8.11　読み出しアドレス検索

図 6.15に読み出しアドレス検索のフローチャートを示します。

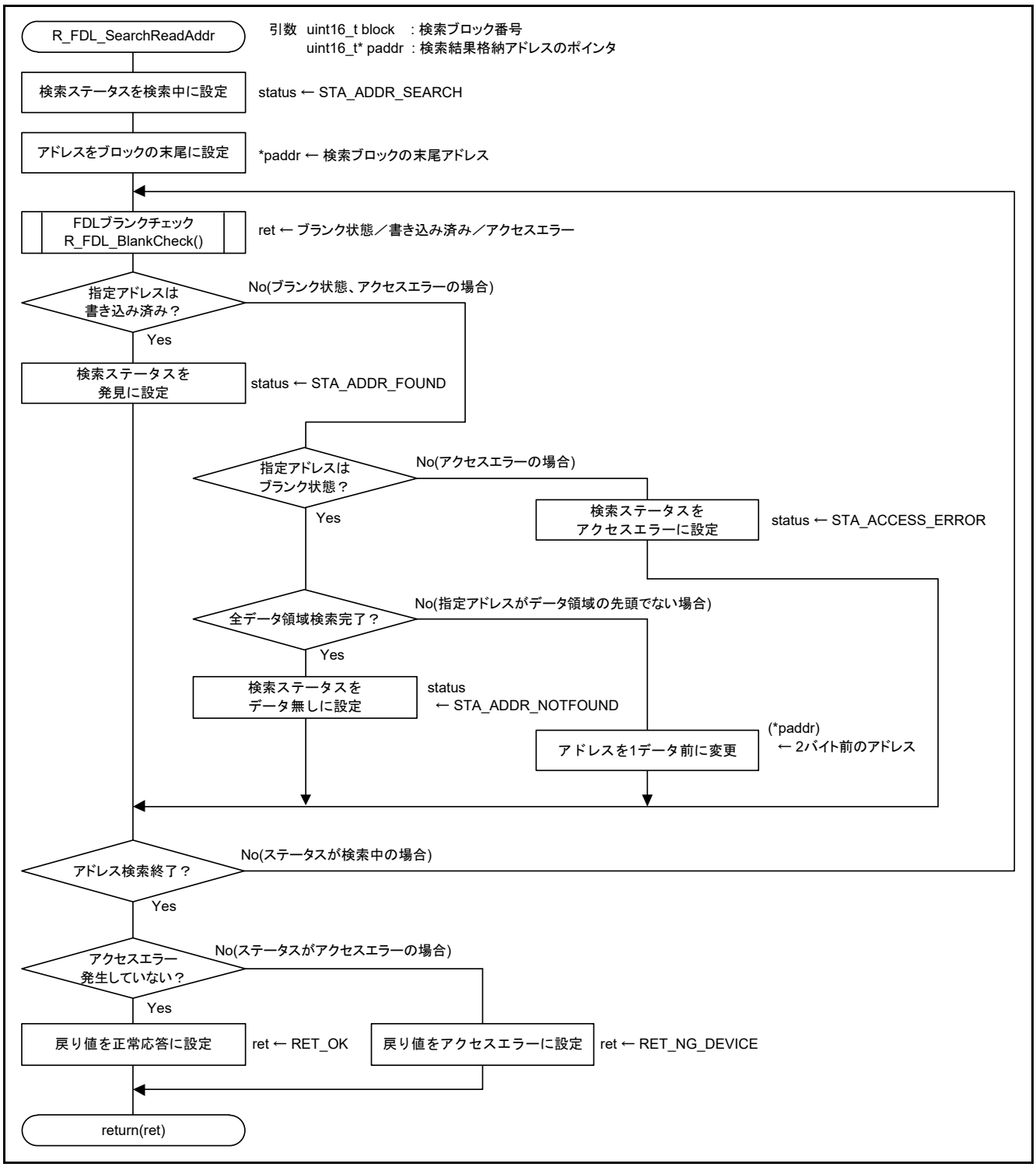


図 6.15 読み出しアドレス検索

6.8.12　ブロック有効化

図 6.16にブロック有効化のフローチャートを示します。

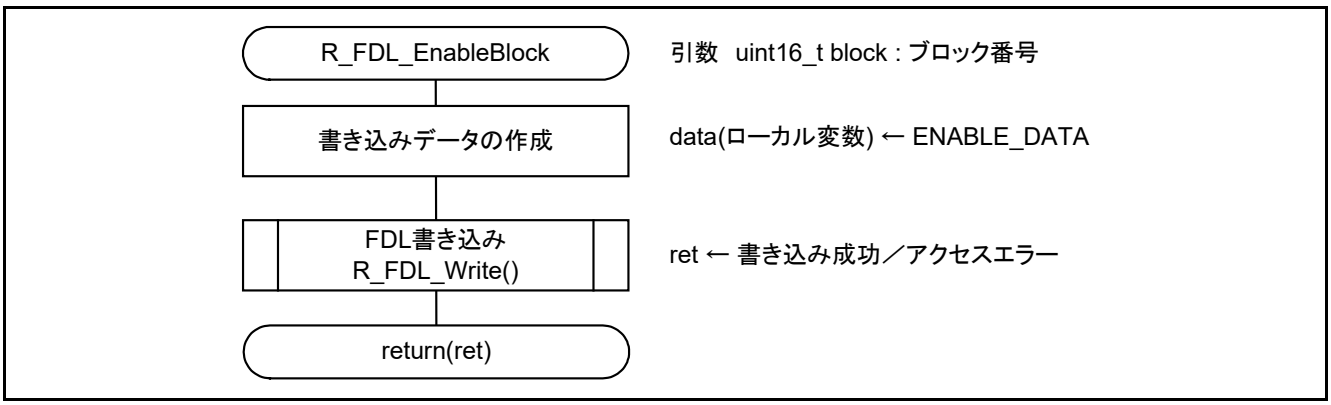


図 6.16 ブロック有効化

6.8.13　LED 点滅データ読み出し

図 6.17にLED点滅データ読み出しのフローチャートを示します。

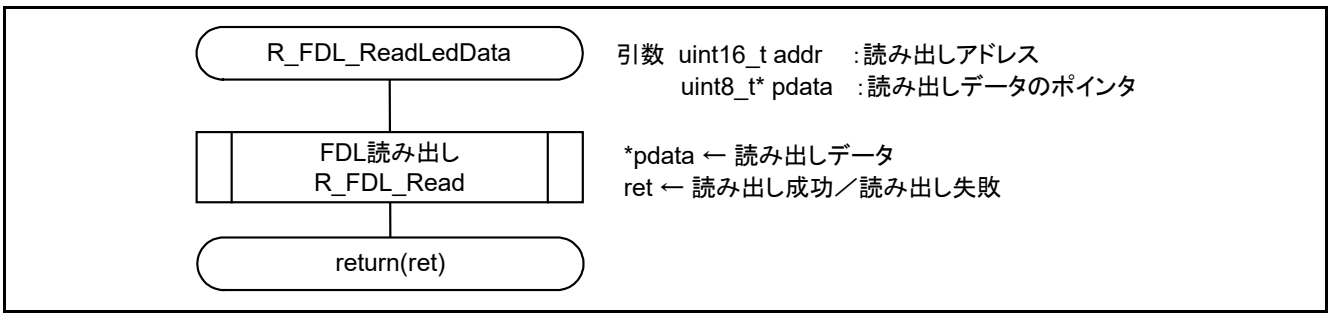


図 6.17 LED 点滅データ読み出し

6.8.14　LED 点滅データ有効範囲チェック

図 6.18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。

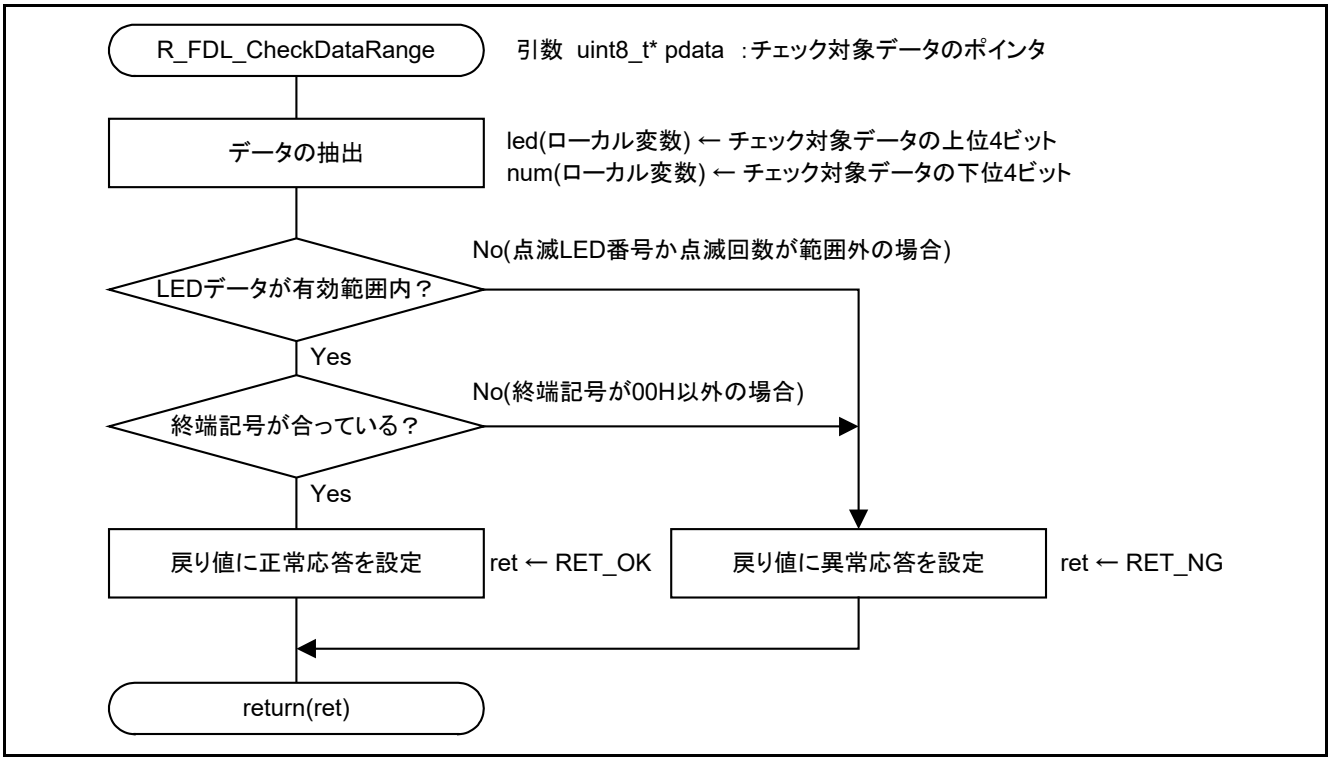


図 6.18 LED 点滅データ有効範囲チェック

6.8.15　TAU01 動作許可設定

図 6.19にTAU01動作許可設定のフローチャートを示します。

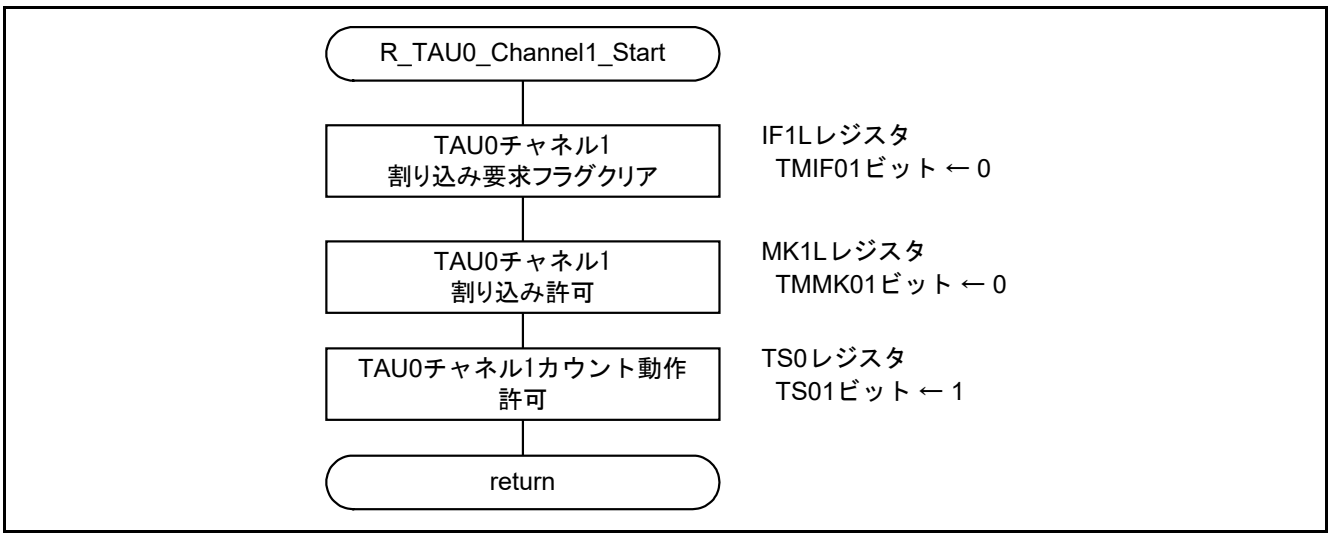


図 6.19 TAU01 動作許可設定

6.8.16　TAU01 割り込みハンドラ

図 6.20にTAU01割り込みハンドラのフローチャートを示します。

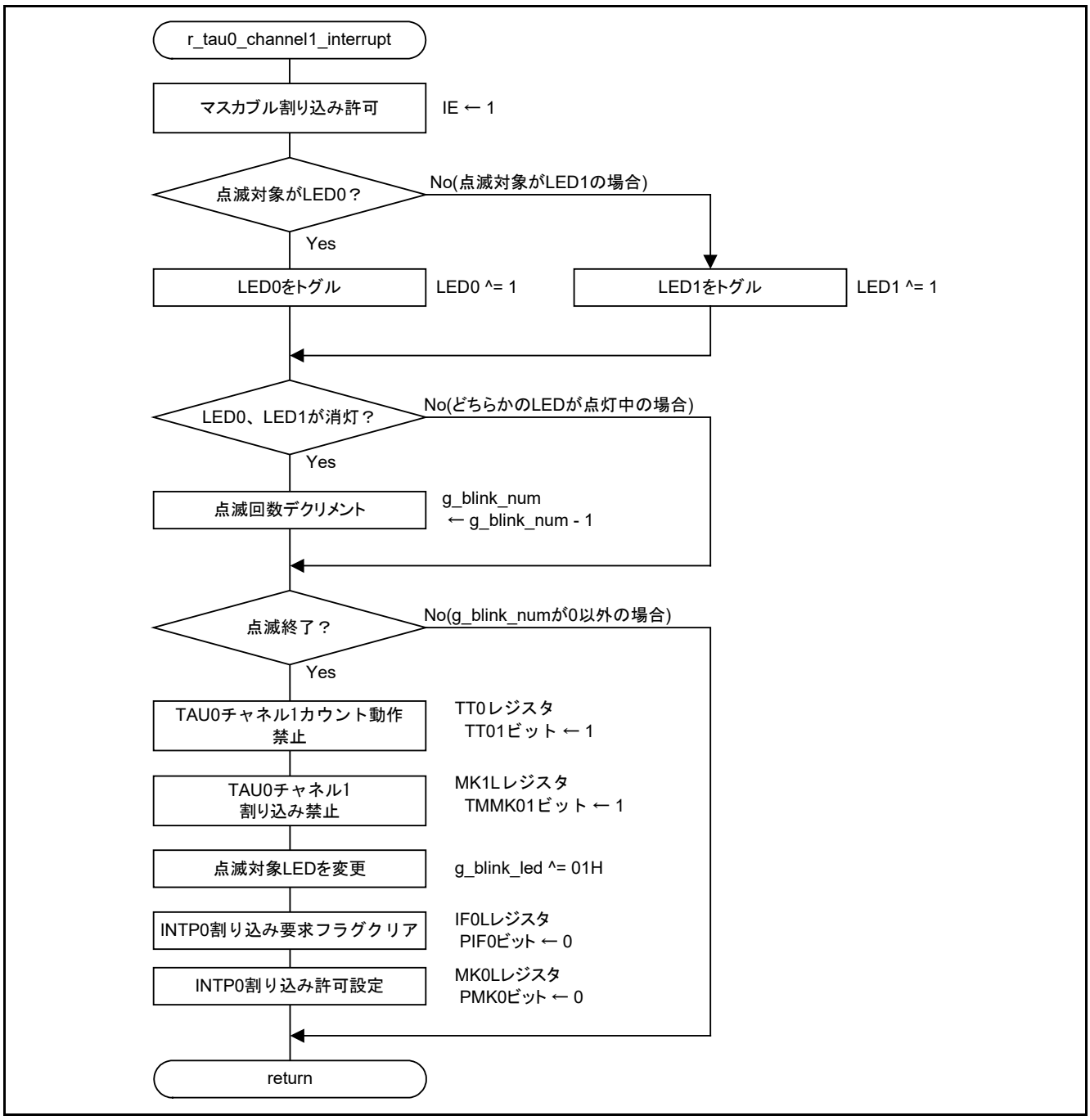


図 6.20 TAU01 割り込みハンドラ

6.8.17　TAU01 動作禁止設定

図 6.21にTAU01動作禁止設定のフローチャートを示します。

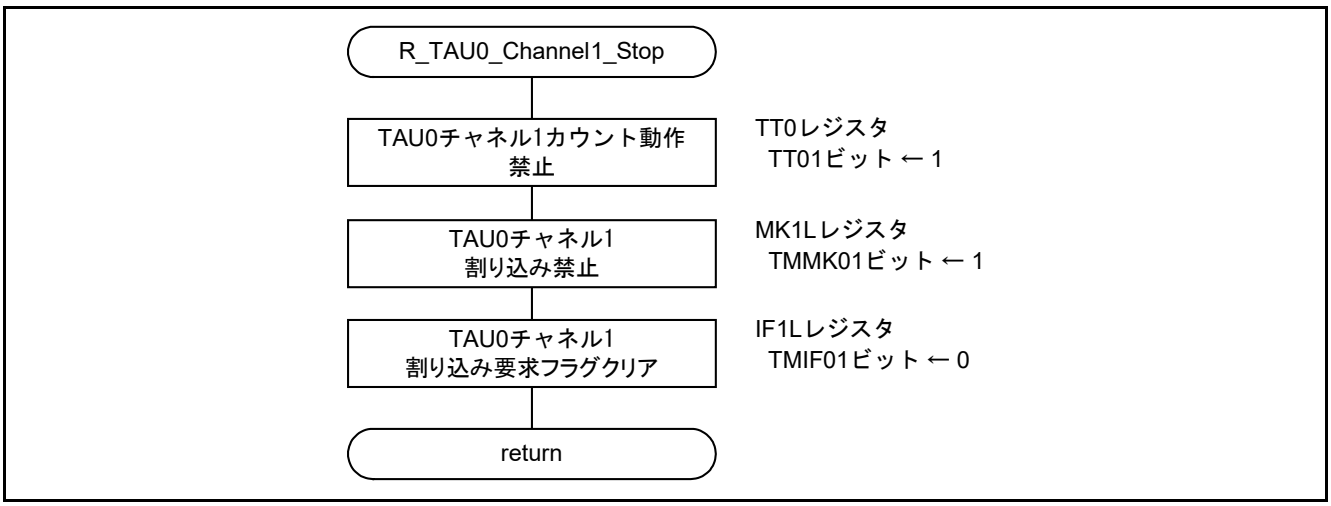


図 6.21 TAU01 動作禁止設定

6.8.18　INTP0 動作許可設定

図 6.22にINTP0動作許可設定のフローチャートを示します。

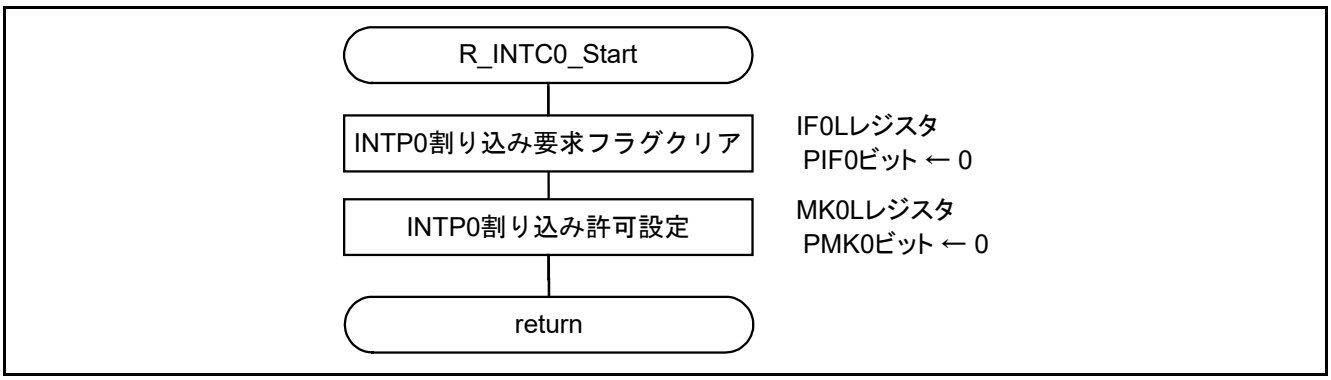


図 6.22 INTP0 動作許可設定

6.8.19　INTP0 割り込みハンドラ

図 6.23にINTP0割り込みハンドラのフローチャートを示します。

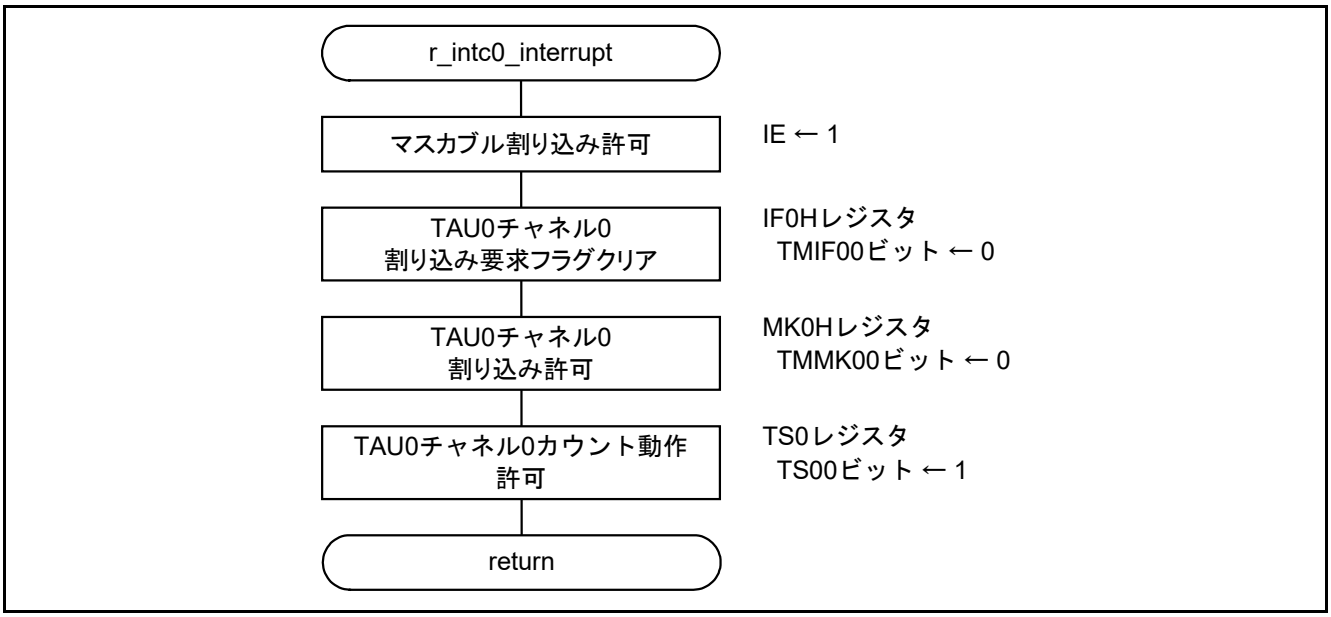


図 6.23 INTP0 割り込みハンドラ

6.8.20　TAU00 動作許可設定

図 6.24にTAU00動作許可設定のフローチャートを示します。

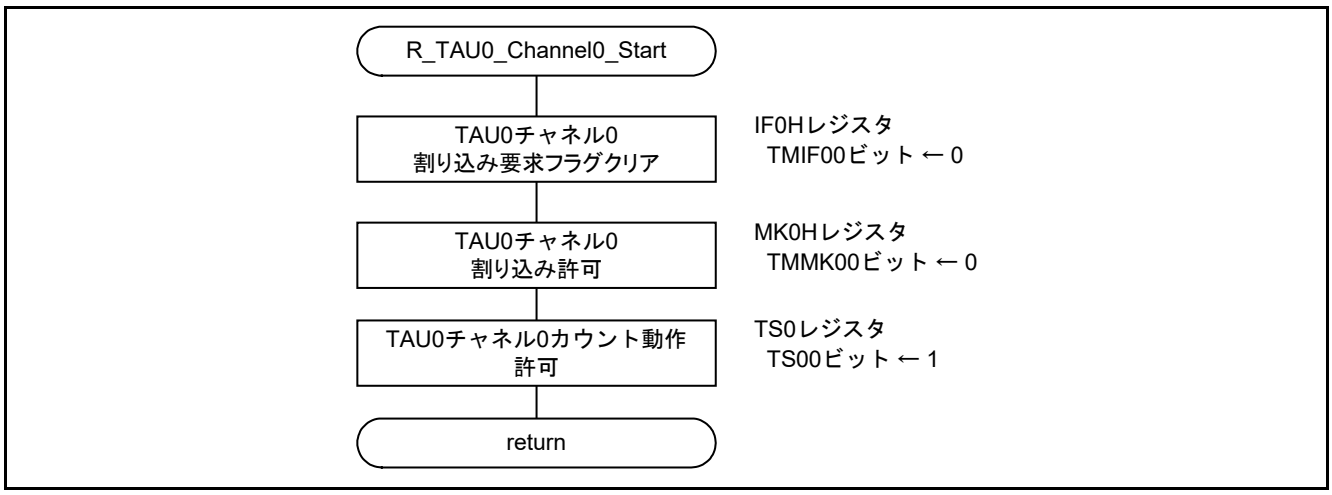


図 6.24 TAU00 動作許可設定

6.8.21　TAU00 割り込みハンドラ

図 6.25にTAU00割り込みハンドラのフローチャートを示します。

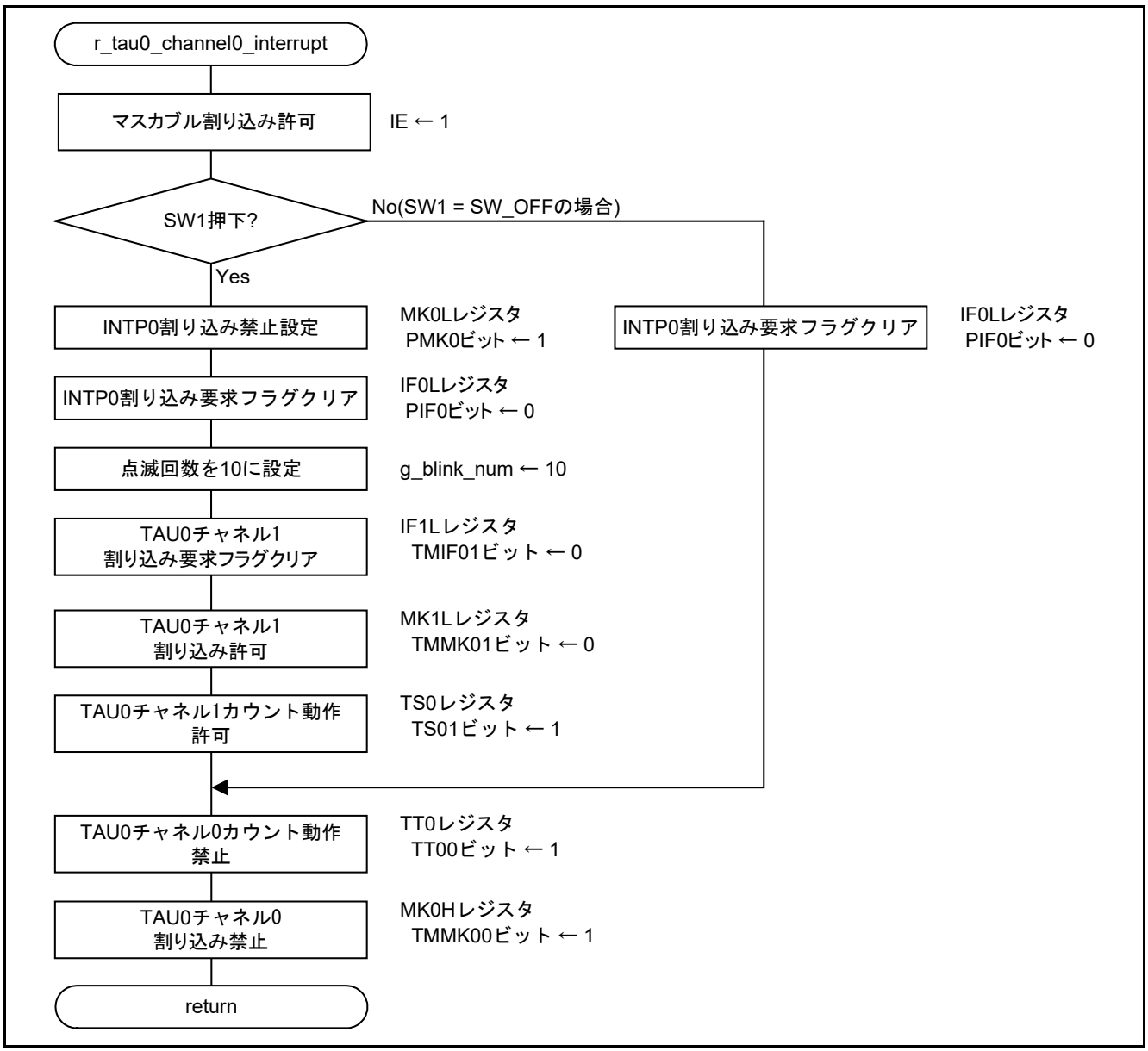


図 6.25 TAU00 割り込みハンドラ

6.8.22　書き込みアドレス取得

図 6.26に書き込みアドレス取得のフローチャートを示します。

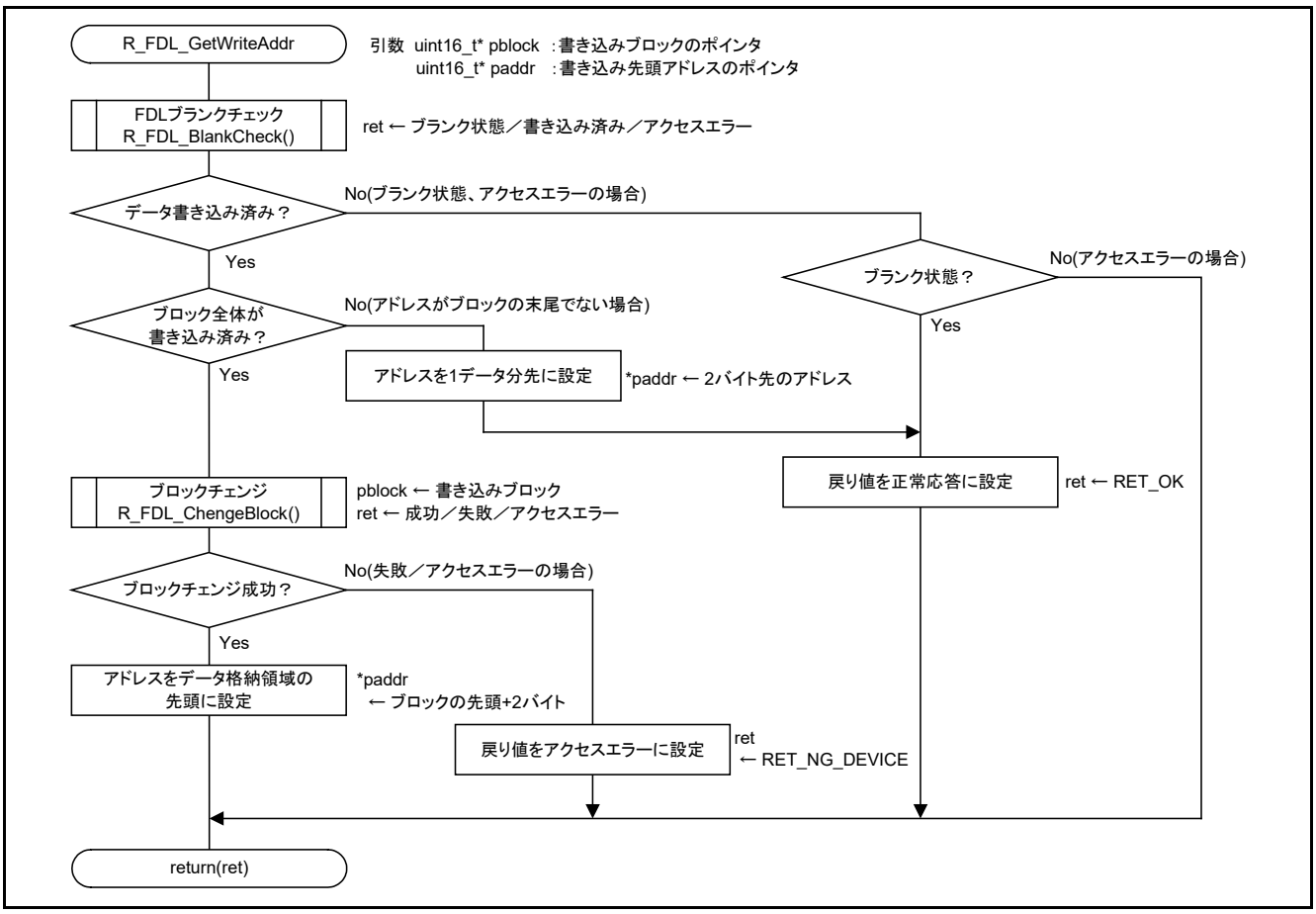


図 6.26 書き込みアドレス取得

6.8.23　LED 点滅データ書き込み

図 6.27にLED点滅データ書き込みのフローチャートを示します。

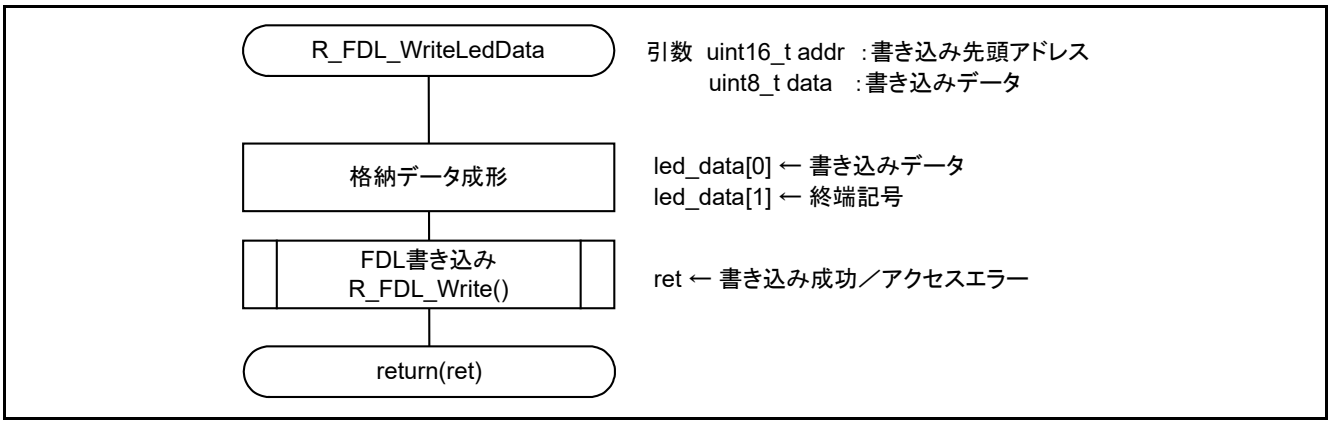


図 6.27 LED 点滅データ書き込み

6.8.24　INTP0 動作禁止設定

図 6.28にINTP0動作禁止設定のフローチャートを示します。

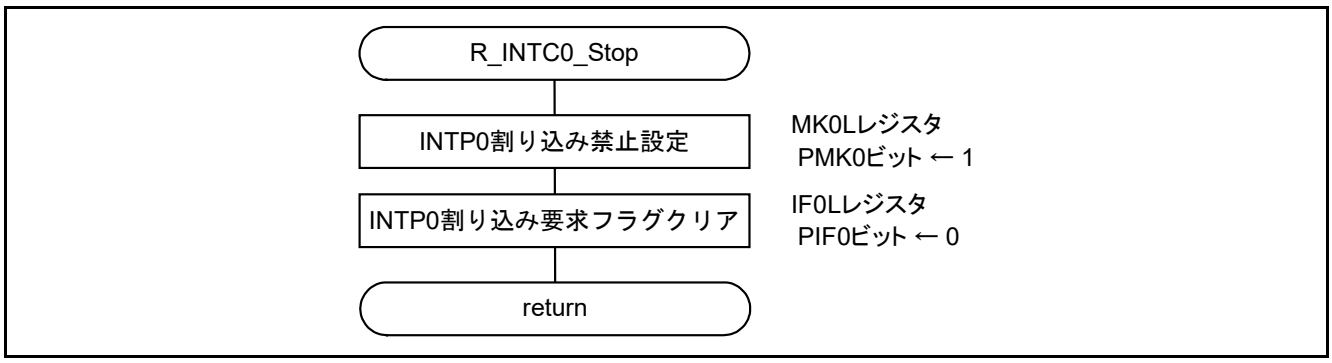


図 6.28 INTP0 動作禁止設定

6.8.25　TAU00 動作禁止設定

図 6.29にTAU00動作禁止設定のフローチャートを示します。

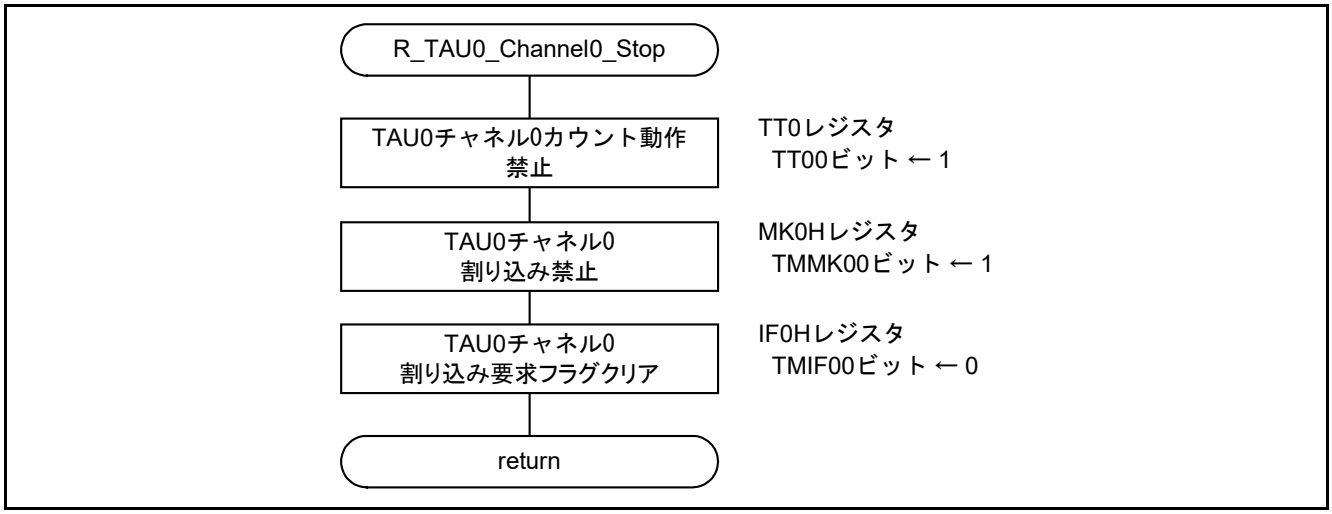


図 6.29 TAU00 動作禁止設定

6.8.26　LVD 割り込み許可設定

図 6.30にLVD割り込み許可設定のフローチャートを示します。

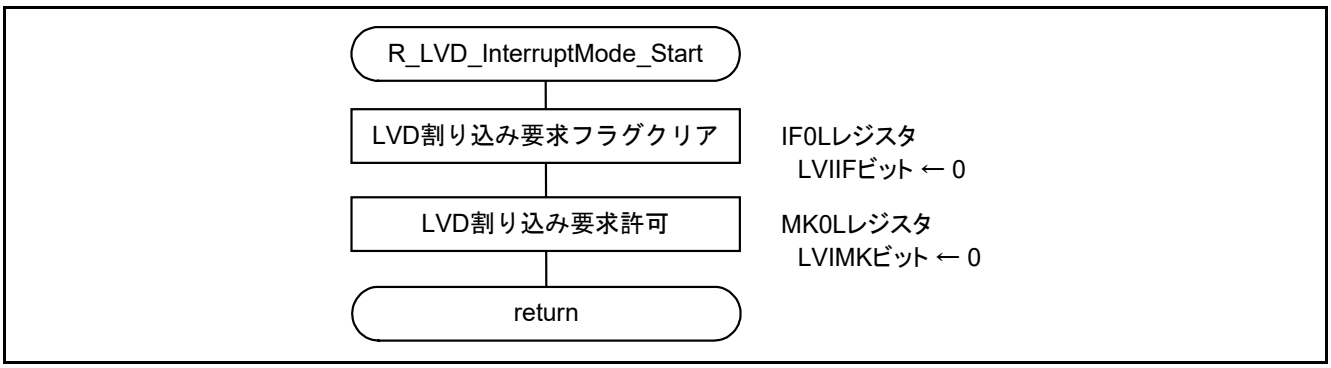


図 6.30 LVD 割り込み許可設定

6.8.27　LVD 割り込みハンドラ

図 6.31にLVD割り込みハンドラのフローチャートを示します。

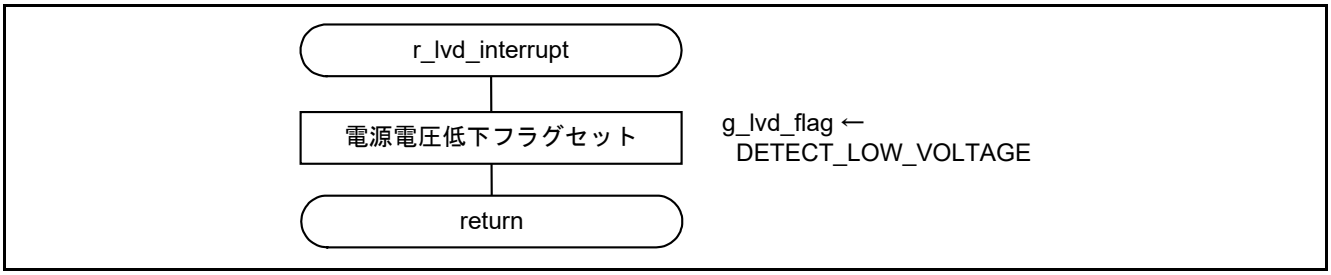


図 6.31 LVD 割り込みハンドラ

6.8.28　管理データ読み出し

図 6.32、図 6.33に管理データ読み出しのフローチャートを示します。

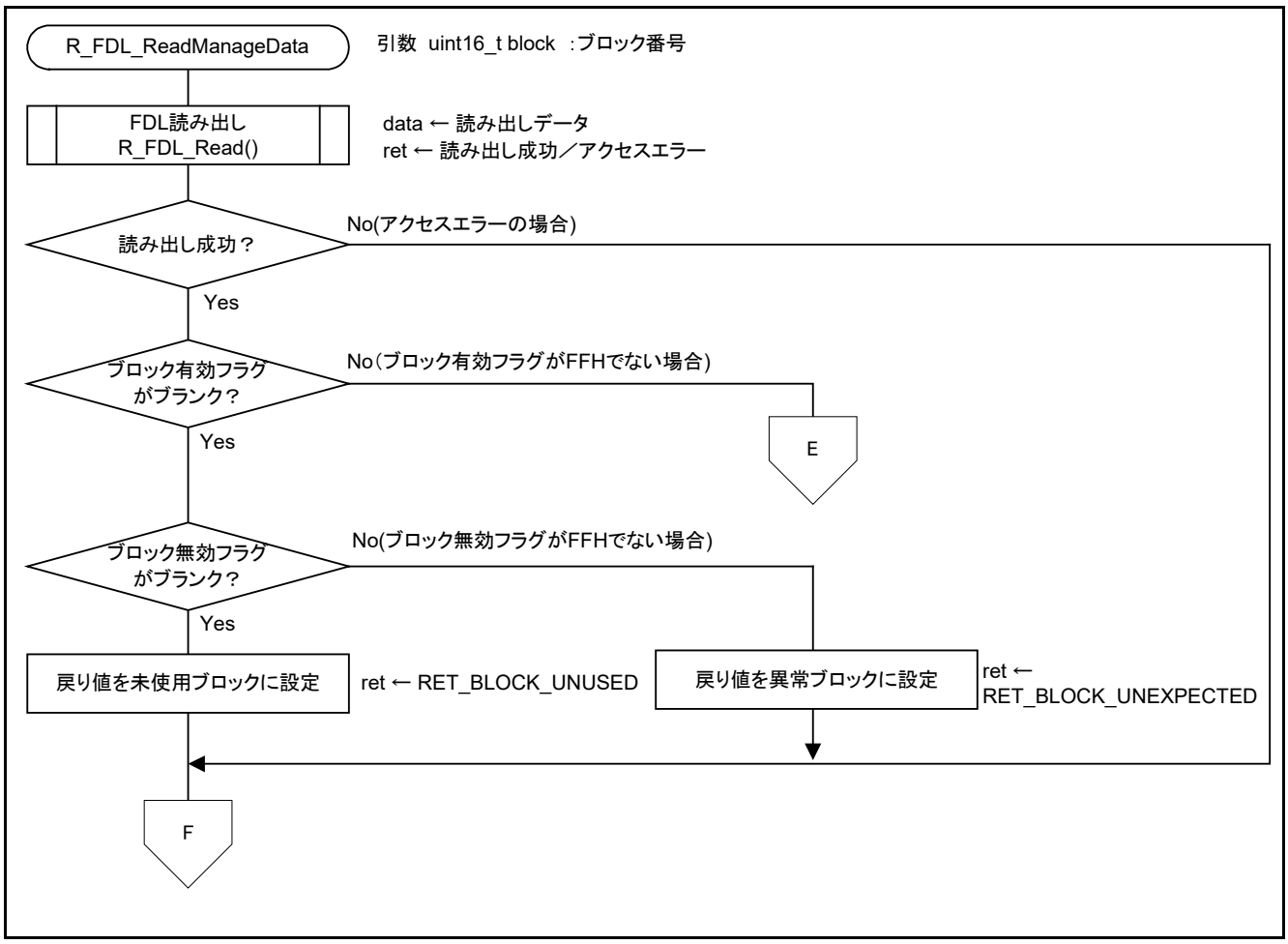


図 6.32 管理データ読み出し(1/2)

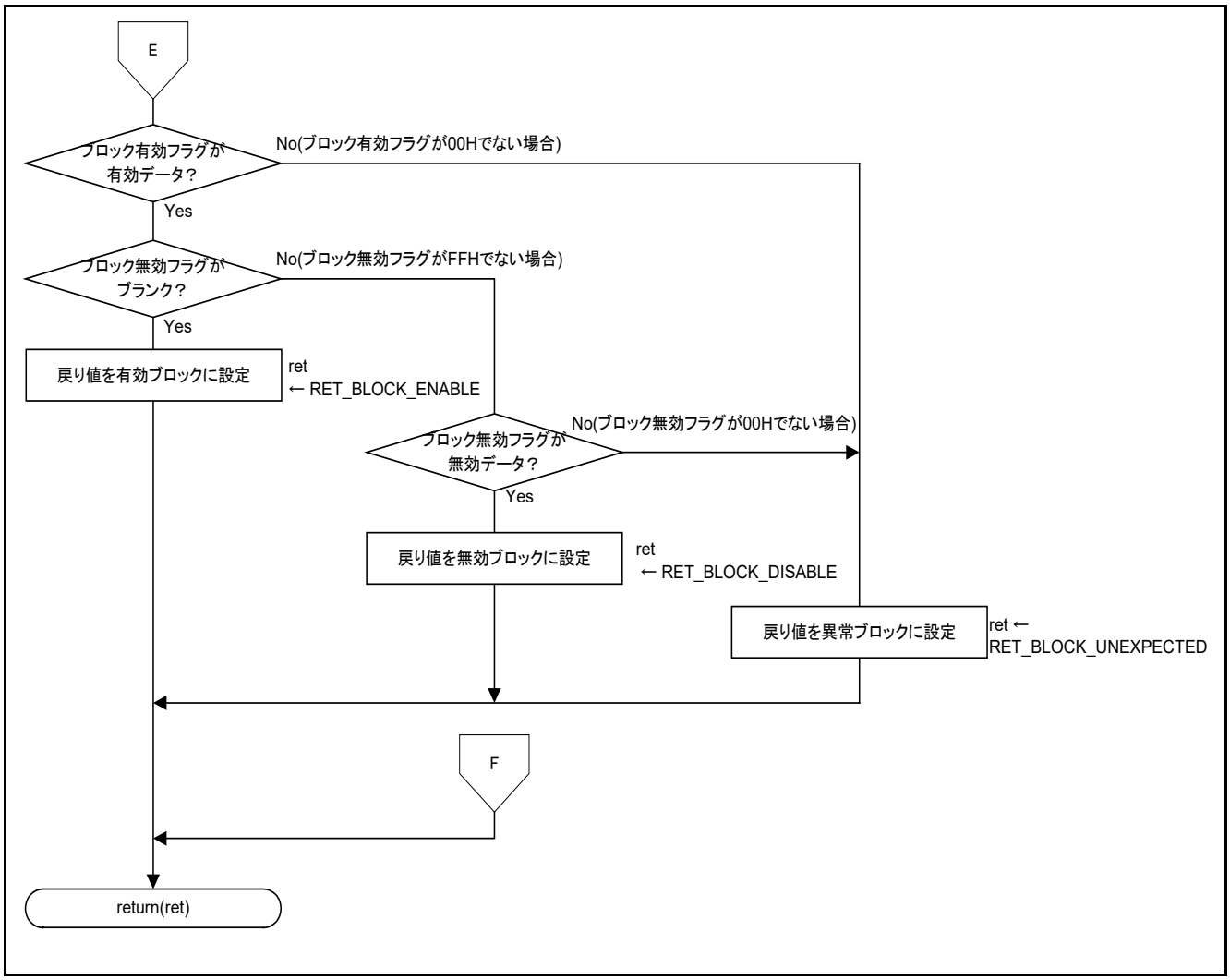


図 6.33 管理データ読み出し(2/2)

6.8.29　FDL 読み出し

図 6.34にFDL読み出しのフローチャートを示します。

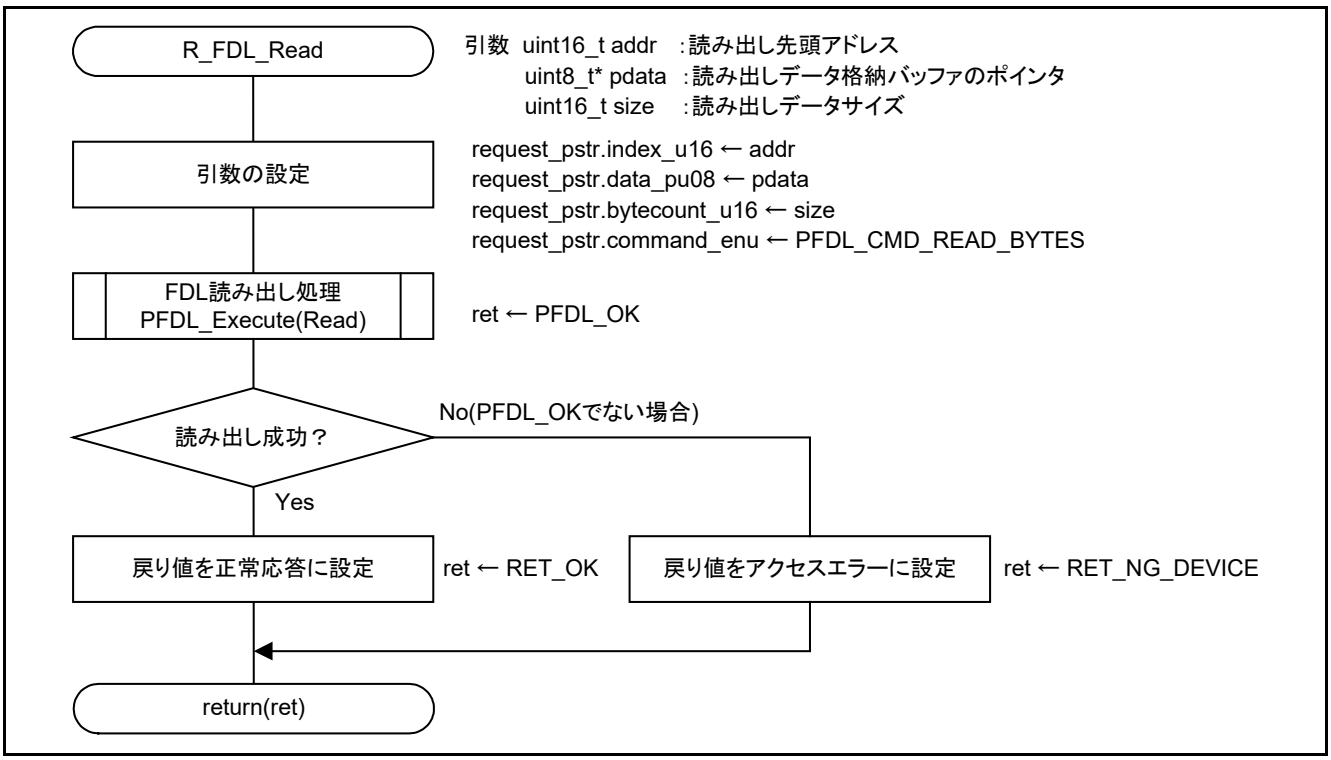


図 6.34 FDL 読み出し

6.8.30　FDL ブランクチェック

図 6.35にFDLブランクチェックのフローチャートを示します。

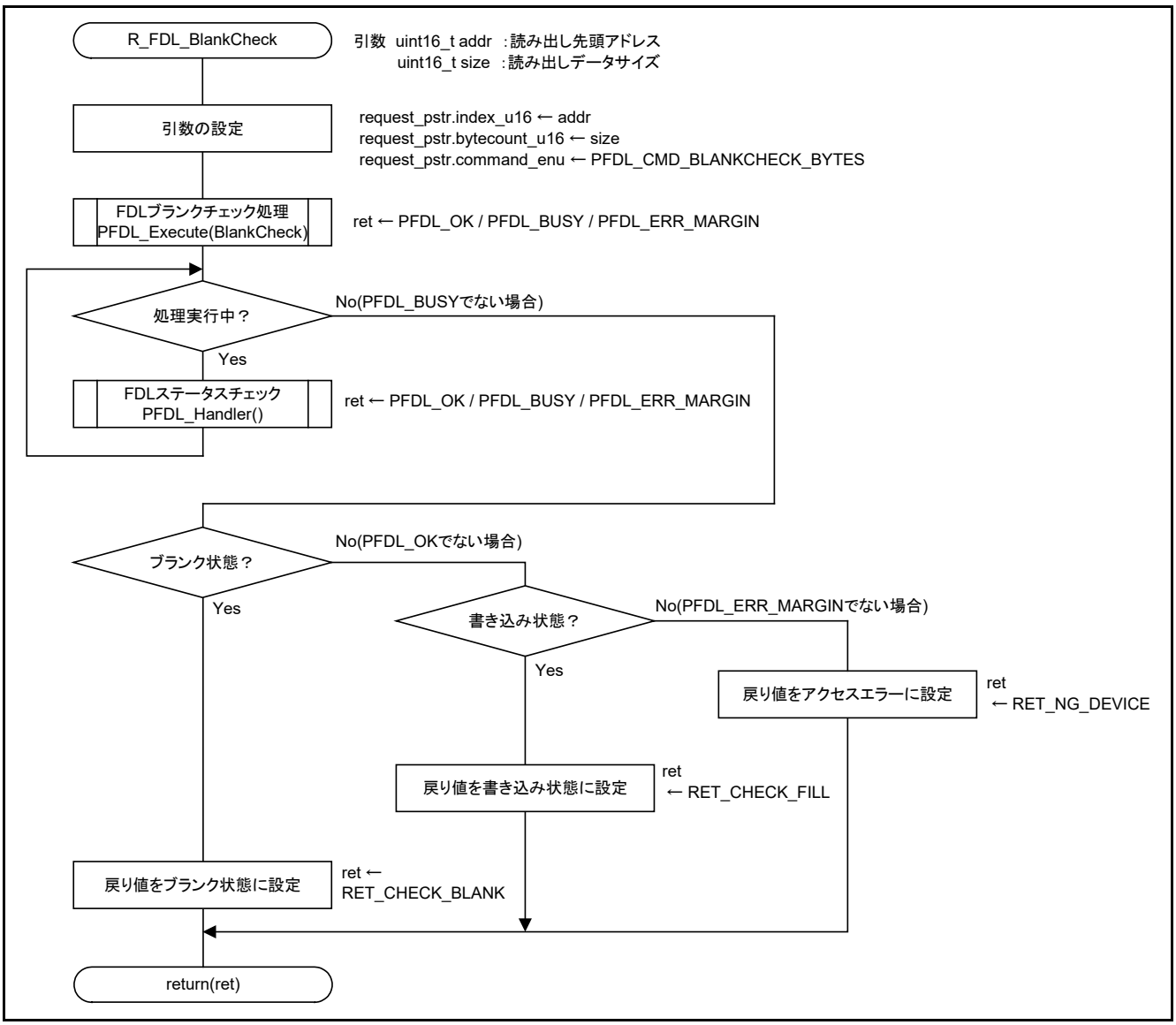


図 6.35 FDL ブランクチェック

6.8.31　FDL 書き込み

図 6.36にFDL書き込みのフローチャートを示します。

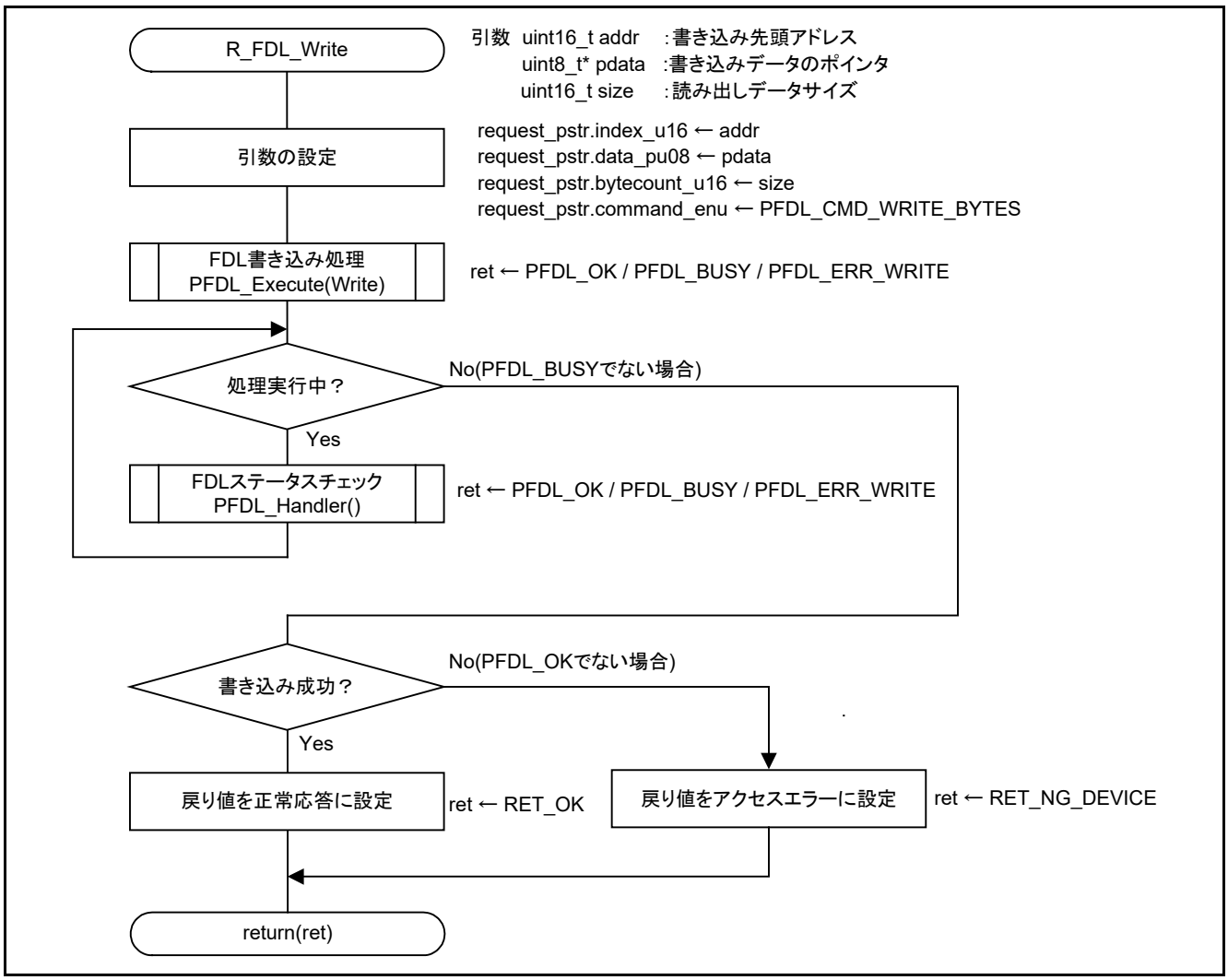


図 6.36 FDL 書き込み

6.8.32　ブロックチェンジ

図 6.37にブロックチェンジのフローチャートを示します。

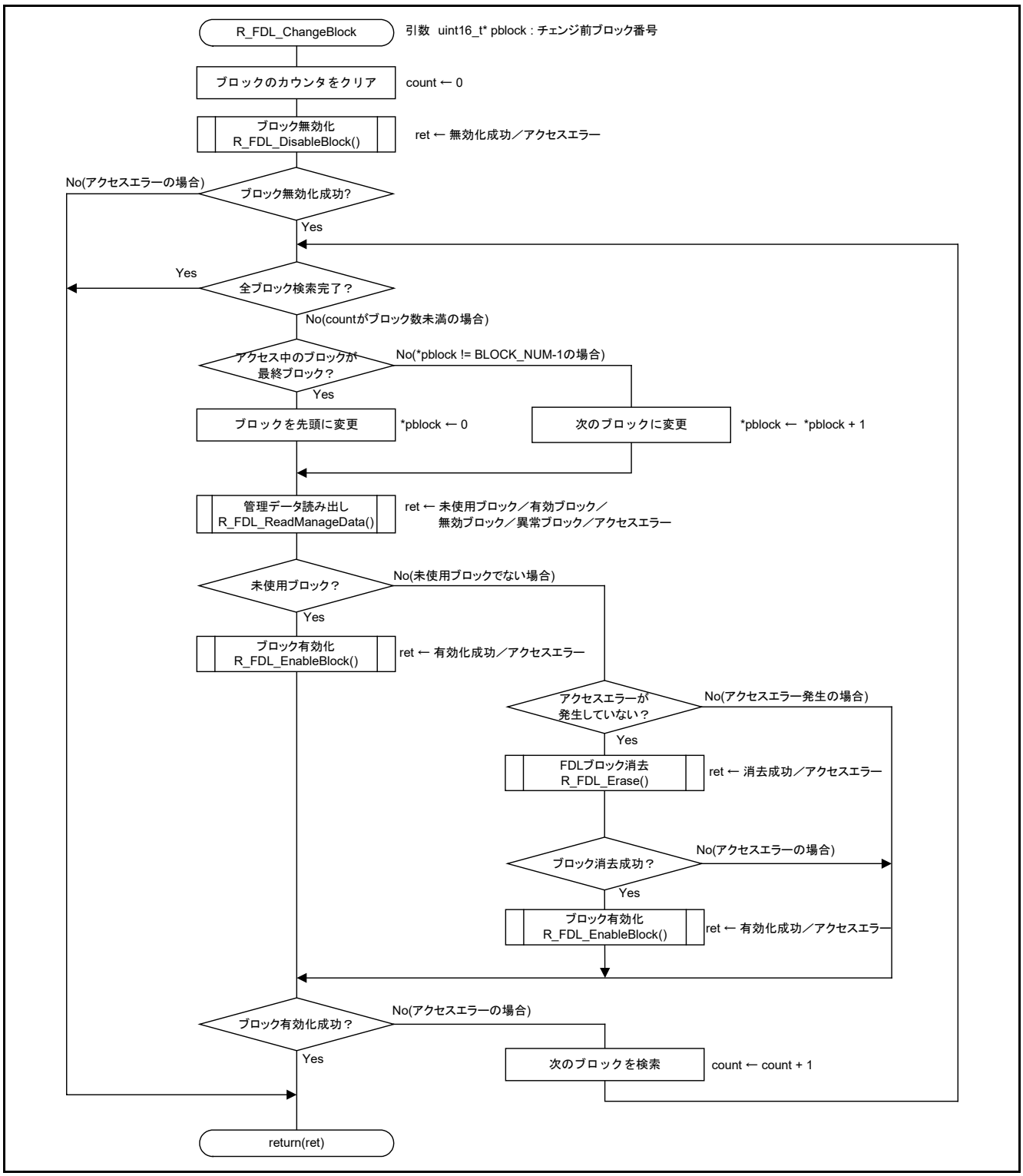


図 6.37 ブロックチェンジ

6.8.33　ブロック無効化

図 6.38にブロック無効化のフローチャートを示します。

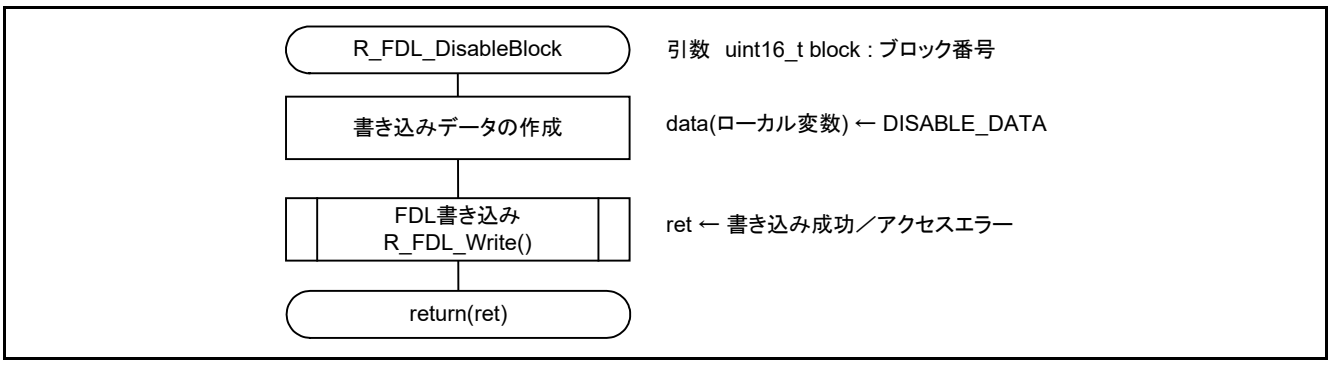


図 6.38 ブロック無効化

6.8.34　FDL ブロック消去

図 6.39にFDLブロック消去のフローチャートを示します。

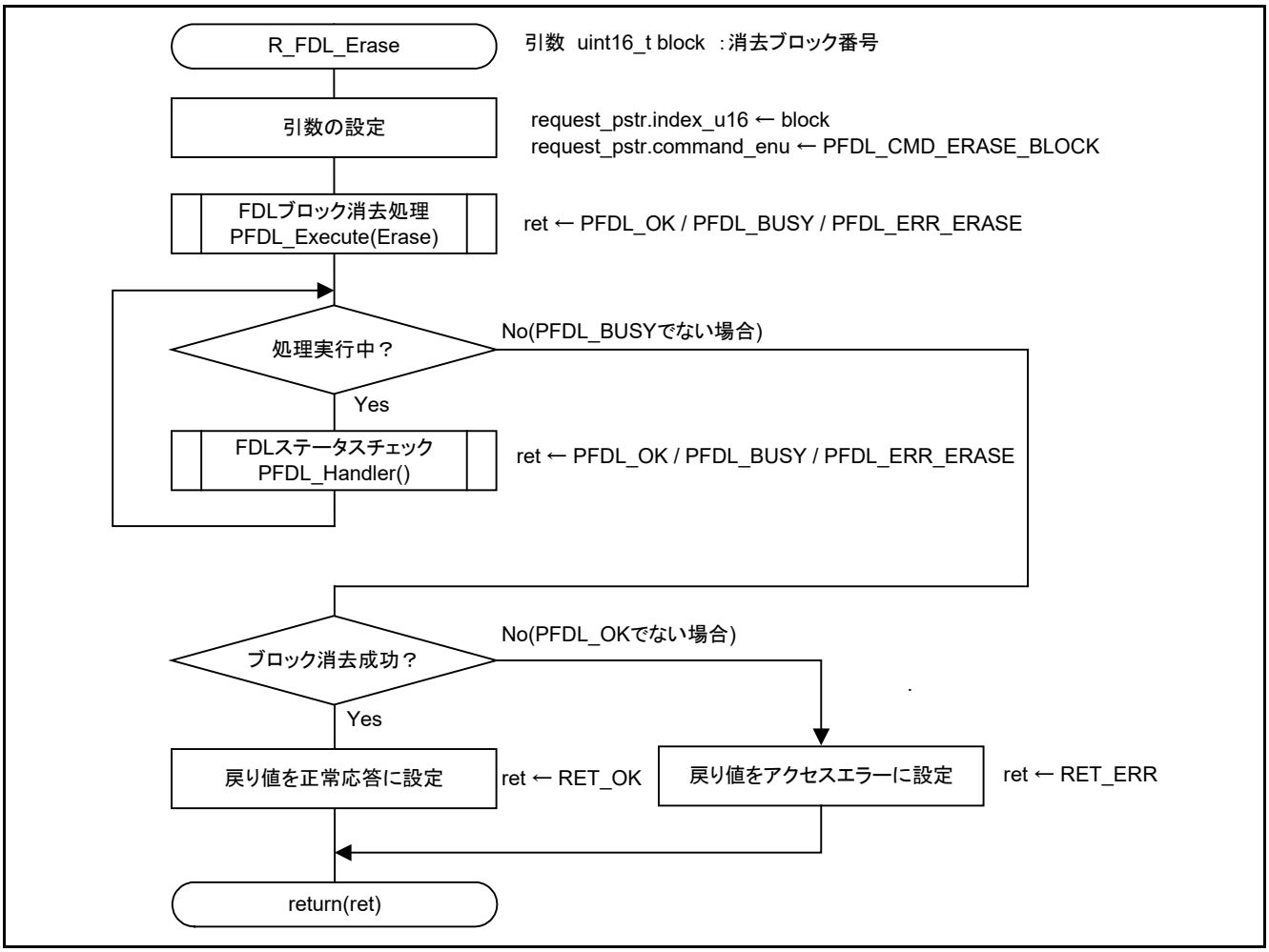


図 6.39 FDL ブロック消去

6.9　FDL の取り込み方

本アプリケーションで使用する FDL 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　CS+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。
- pfdl.h
- pfdl_types.h
- pfdl.lib

CS+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。（.h、.lib、.dr）

6.9.2　e2studio 版

(1) プロジェクト内、user_src ディレクトリに以下のファイルをコピーする。
- pfdl.h
- pfdl_types.h
- pfdl.lib

(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。

6.9.3　IAR 版

(1) インストールした FDL のフォルダをプロジェクトのルートディレクトリにコピーする。

(2) IAR のプロジェクトのオプションから「C/C++コンパイラ」→「プリプロセッサ」を選択する。

(3) 「追加インクルードディレクトリ」にルートディレクトリにコピーしたフォルダをサブフォルダまで指定する。

(例：$PROJ_DIR$¥FDL¥IAR_210¥lib)

(4) 「https://github.com/IARSystems/ewrl78-linker-config-flashlibs」

上の GitHub のページから IAR 用のリンカスクリプトをダウンロードする。

(5) ダウンロードした中にある以下のフォルダをユーザー関数のフォルダにコピーする。
- trio_lnkR5F100xE.icf
- common.icf
- self_ram.icf

(6) コピーした「trio_lnkR5F100xE.icf」の名前を「trio_lnkR5F10RLC.icf」に変更し、以下の通りにコードを修正する。

```
define symbol _FLASH_END = 0x07FFF;
define symbol _RAM_START = 0xFF900;
define symbol _DATAFRASH_SZ = 2K;
```

(7) IAR のプロジェクトのオプションから「リンカ」→「設定」→「デフォルトのオーバーライド」にチェックを付ける。

(8) 修正後の「trio_lnkR5F10RLC.icf」を指定する。

(9) 「設定ファイルシンボルの定義」に GitHub のページの下部にある「RAM reservation symbols」の表から対応する FDL のシンボルを記述する。

(例：__RESRVE_T04_FDL=1)

(10) IAR のプロジェクトのオプションから「リンカ」→「ライブラリ」→「追加ライブラリ」に FDL のライブラリを追加する

(例：$PROJ_DIR$¥FDL¥IAR_210¥lib¥pfdl.a)

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　：CS+版、e2studio 版

ファイル名：r_cg_main.c

修正箇所　：■R_MAIN_UserInit();の記載場所

main 関数内に生成される Start user code.の上の R_MAIN_Userinit();を削除する。

```
void main(void)
{
    R_MAIN_UserInit();
    /* Start user code. Do not edit comment generated here */
```

↓

```
void main(void)
{

    /* Start user code. Do not edit comment generated here */
```

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/L12 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。）

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2022.06.24	-	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. 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。

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。