お客様各位

カタログ等資料中の旧社名の扱いについて

2010年4月1日を以ってNECエレクトロニクス株式会社及び株式会社ルネサステクノロジが合併し、両社の全ての事業が当社に承継されております。従いまして、本資料中には旧社名での表記が残っておりますが、当社の資料として有効ですので、ご理解の程宜しくお願い申し上げます。

ルネサスエレクトロニクス ホームページ（http://www.renesas.com）

2010年4月1日
ルネサスエレクトロニクス株式会社

【発行】ルネサスエレクトロニクス株式会社（http://www.renesas.com）

【問い合わせ先】http://japan.renesas.com/inquiry

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。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。
3. 当社製品を改造、改変、複製等しないでください。
4. 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。
5. 輸出に際しては、「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。本資料に記載されている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の目的で使用しないでください。また、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に使用することができません。
6. 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。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されてい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生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   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

   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、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社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。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、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、社会的損害など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、機器またはシステムとしての出荷保証をお願いいたします。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、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、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。
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固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

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。

SH7216 グループ

データフラッシュ書き換え効率化例

要旨

本アプリケーションノートでは、データフラッシュ・ドライバソフトウェアサンプルを使用して、データを管理する際の使用方法、設定手順、使用例について紹介します。

なお、本アプリケーションノートに掲載されているタスク例およびアプリケーション例は確認済みですが、実際にご使用になる場合には、必ず動作環境を確認の上ご使用くださいますようにお願いします。

動作確認デバイス

SH7216

目次

1. データフラッシュ・ドライバソフトウェアサンプルの仕様について 2
2. データフラッシュ・ドライバソフトウェアサンプル使用時の設定方法について 57
3. データフラッシュ（FLD）使用上の注意事項 59
4. 参考ドキュメント 59

1. データフラッシュ・ドライバソフトウェアサンプルの仕様について

データフラッシュ・ドライバソフトウェアサンプル（以下、データフラッシュドライバ、または単にドライバとします。）は、SH7216 に内蔵されているデータフラッシュへのデータの格納および管理をするためのデバイスドライバサンプルプログラムです。

データフラッシュドライバのユーザ API 関数（以下、単にデータフラッシュドライバ関数、またはドライバ関数とします。）をコールするだけで、データの更新、データの読み出しなどが可能です。

1.1 機能概要

(1) データフラッシュドライバ関数をコールするだけで、データの更新、データの読み出しが可能です。データフラッシュドライバを使用すれば、データフラッシュを使用したデータ管理システムを容易に構築することが可能です。
(2) データフラッシュドライバは、各データに対して論理的なデータ番号を付与し、データの更新、データの読み出しを行うことが可能な論理ブロック型デバイスドライバです。
(3) データフラッシュドライバは、データフラッシュ上に、0～256 バイトのサイズの異なる複数のデータを管理することが可能です。
(4) データフラッシュドライバは、データを更新するたびにイレーズ⇒プログラムを行うのではなく、ブロック内に存在する書き込み可能領域に、領域をずらしながらプログラムを行います。データフラッシュドライバを使用すれば、デバイスの持つ書き換え（イレーズ）可能回数に対して、より多いデータ更新可能回数を実現することが可能です。
(5) データフラッシュドライバは、ブロック内に書き込み可能領域がなくなった場合に、ブロックグループ内の空いているもう一方のブロックにデータをコピーし、書き込み可能領域を生成します。これらの処理は、データフラッシュドライバ関数をコールすることにより、すべてデータフラッシュドライバ内で自動的に実行されます。
(6) データフラッシュドライバを使用して、データ更新を実行中にシステムがシャットダウンした場合、データフラッシュドライバは、次回のドライバ初期化時に、更新前のデータに復旧します。
(7) データフラッシュドライバを使用して、イレーズを実行中にシステムがシャットダウンした場合、データフラッシュドライバは、次回のドライバ初期化時に、イレーズが中断されたことを検出することができます。

1.2 ご使用時の注意事項

1.2.1 他のプログラムとの排他制御

データフラッシュドライバは、ユーザ API 関数内で、フラッシュメモリ／データフラッシュ専用のシーケンサ（FCU）制御レジスタの操作、コマンドを発行します。これらの処理は、データフラッシュドライバのみが行うことを前提にして設計しています。データフラッシュドライバが動作中に、他のプログラムにより以下の処理が発生しないようにしてください。

【データフラッシュドライバとの排他制御が必要な処理】

① FCU 制御レジスタへの Write アクセス

② FCU へのコマンド発行

③ データフラッシュアドレス領域への直接的なアクセス

1.2.2 ウェイト／タイムアウト検出用タイマモジュール

本ドライバは、ウェイト処理、およびタイムアウト検出処理用にマイコン内蔵のマルチファンクションタイマパルスユニット（MTU2）を使用しているため、他のプログラムにより本モジュールの該当チャネルはご使用になれません。本モジュール動作に関連するレジスタ、または共通レジスタ内の関連するビットを変更しないようにしてください。

1.3 ファイル構成

データフラッシュドライバは、以下のファイル構成になっています。

表 1.1 データフラッシュドライバファイル構成表

ファイル名	説明
DF_user.h	ユーザ API 関数のヘッダファイル。ドライバを使用する際にインクルードするファイルです。 データフラッシュドライバ使用時に必要なクロック設定値、データ数、API 関数内割り込みマスクレベルを設定します。システム仕様に基づき設定を行ってください。また、ユーザ API 関数のプロトタイプ、戻り値、使用する引数が定義されています。
DF_user.c	データフラッシュドライバのデータ番号ごとのデータサイズ、およびデータを格納する領域（ブロックグループ）を設定するファイル。システム仕様に基づき設定を行ってください。
DF_common.h	データフラッシュの情報設定ファイル。ドライバを使用する際にインクルードするファイルです。 データフラッシュドライバ内で使用する設定値定義ファイル。
DF_drv.h	データフラッシュドライバの内部ステータス設定値定義ファイル。ドライバを使用する際にインクルードするファイルです。
DF_drv.c	ユーザ API 関数の C ソースファイル。 • 初期化関数 • フォーマット関数 • 有効データ読み出し関数 • データ更新関数 • ブロック消去関数 等の、データフラッシュドライバ本体です。
DF_control.h	FCU 制御レジスタの設定値、FCU 制御コマンド、ウェイト／タイムアウト検出用タイマモジュール設定値定義ファイル。ドライバを使用する際にインクルードするファイルです。
DF_control.c	FCU 制御関数の C ソースファイル。
DF_typedef.h	データフラッシュドライバ用型宣言ファイル。

1.4 データフラッシュ内ブロック構成

データフラッシュドライバは、データフラッシュ内に、図 1.1 に示すブロックの組み合わせによるグループを構成し、グループごとに各ブロックの状態を管理します。

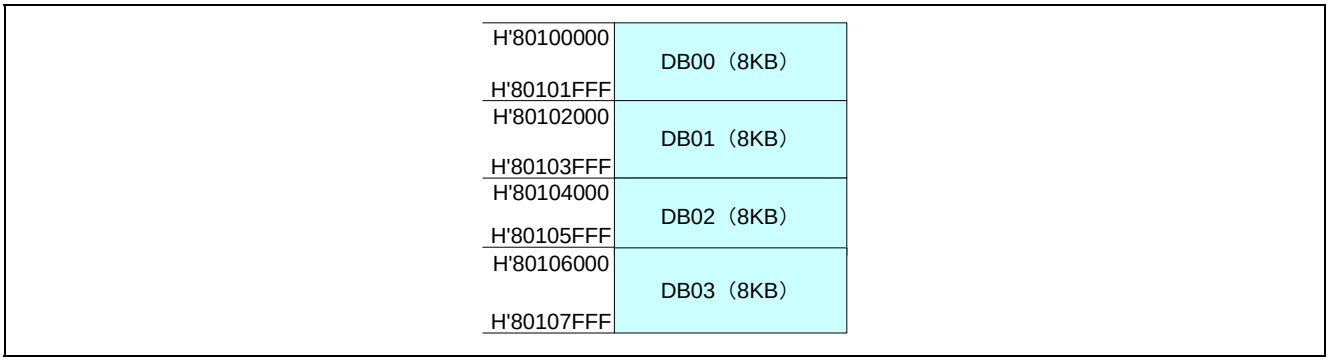

図 1.1　データフラッシュ・ブロック構成

1.5 ブロック内データ構造

データフラッシュドライバは、各ブロック内に、図 1.2 に示すデータ構造を構築し、データを管理します。ブロック内は、ブロック管理情報領域、データ管理情報領域とデータ領域に分けられ、データ管理情報領域とデータ領域は 1 対 1 で対応しています。

ブロック内オフセット	内容	サイズ	
0x000 0x3FF	データ領域 - 6	1024 Byte	データ領域
0x400 0x7FF	データ領域 - 5	1024 Byte	
0x800 0xBFF	データ領域 - 4	1024 Byte	
0xC00 0xFFF	データ領域 - 3	1024 Byte	
0x1000 0x13FF	データ領域 - 2	1024Byte	
0x1400 0x17FF	データ領域 - 1	1024 Byte	
0x1800 0x1BFF	データ領域 - 0	1024 Byte	
0x1C00 0x1C1F	データ管理情報領域 - 6	32 Byte	データ管理情報領域
0x1C20 0x1C3F	データ管理情報領域 - 5	32 Byte	
0x1C40 0x1C5F	データ管理情報領域 - 4	32 Byte	
0x1C60 0x1C7F	データ管理情報領域 - 3	32 Byte	
0x1C80 0x1C9F	データ管理情報領域 - 2	32 Byte	
0x1CA0 0x1CBF	データ管理情報領域 - 1	32 Byte	
0x1CC0 0x1CDF	データ管理情報領域 - 0	32 Byte	
0x1CE0 0x1CFF	ブロック管理情報領域	32 Byte	ブロック管理情報領域

図 1.2　データフラッシュ・ブロック内データ構造

1.6 ブロック管理情報

ブロック管理情報領域は、図 1.3 に示すように、ブロック ID、データリクレーム完了フラグ、ブロック消去開始フラグ、ブロック消去完了フラグに分けられます。

ブロック管理情報によるブロックの状態確認はドライバ初期化時に行われます。

ブロック内オフセット	内容	サイズ
0x1CE0 0x1CE7	ブロック消去完了フラグ	8 Byte
0x1CE8 0x1CEF	ブロック消去開始フラグ	8 Byte
0x1CF0 0x1CF7	データリクレーム完了フラグ	8 Byte
0x1CF8 0x1CFF	ブロックID	8 Byte

図 1.3　ブロック管理情報領域内データ構造

1.6.1 ブロック ID

ブロックが有効な状態（データの書き込みが可能な状態）であるかを示す情報です。

ブロック ID はブランクブロック（ブロック全体がブランク状態のブロック）に対して最初にプログラムされる情報ですので、ドライバは、ブロック ID 領域がブランク状態であり、かつそのブロック全体がブランク状態でないブロックを発見した場合、データフラッシュ全体が本ドライバによりフォーマットされていない状態と判断し、ユーザアプリケーションに対し、フォーマット要求を戻り値として返します。

また、ドライバは、ブランクブロックを発見した場合、同ブロックグループ内のペアとなるブロックのブロック消去開始フラグの状態により、ブロック消去処理中の異常なシステムシャットダウンによるブロック ID 未書き込み状態、もしくは未フォーマット状態と判断します（ブロック状態の判断は表 1.2 を参照ください）。

1.6.2 データリクレーム完了フラグ

ブロック内に有効なデータ（DF_user.c 内で設定された、同ブロックグループ内に格納される有効データ）がすべて格納された状態であるかを示す情報です。

1.6.3 ブロック消去開始フラグ／ブロック消去完了フラグ

同ブロックグループ内のペアとなるブロックの消去処理状態を示す情報です。「1.6.1　ブロック ID」、「1.6.2　データリクレーム完了フラグ」との組み合わせにより、表 1.2 のようにブロック状態を判断します。

1.6.4 ブロック消去の概略フロー

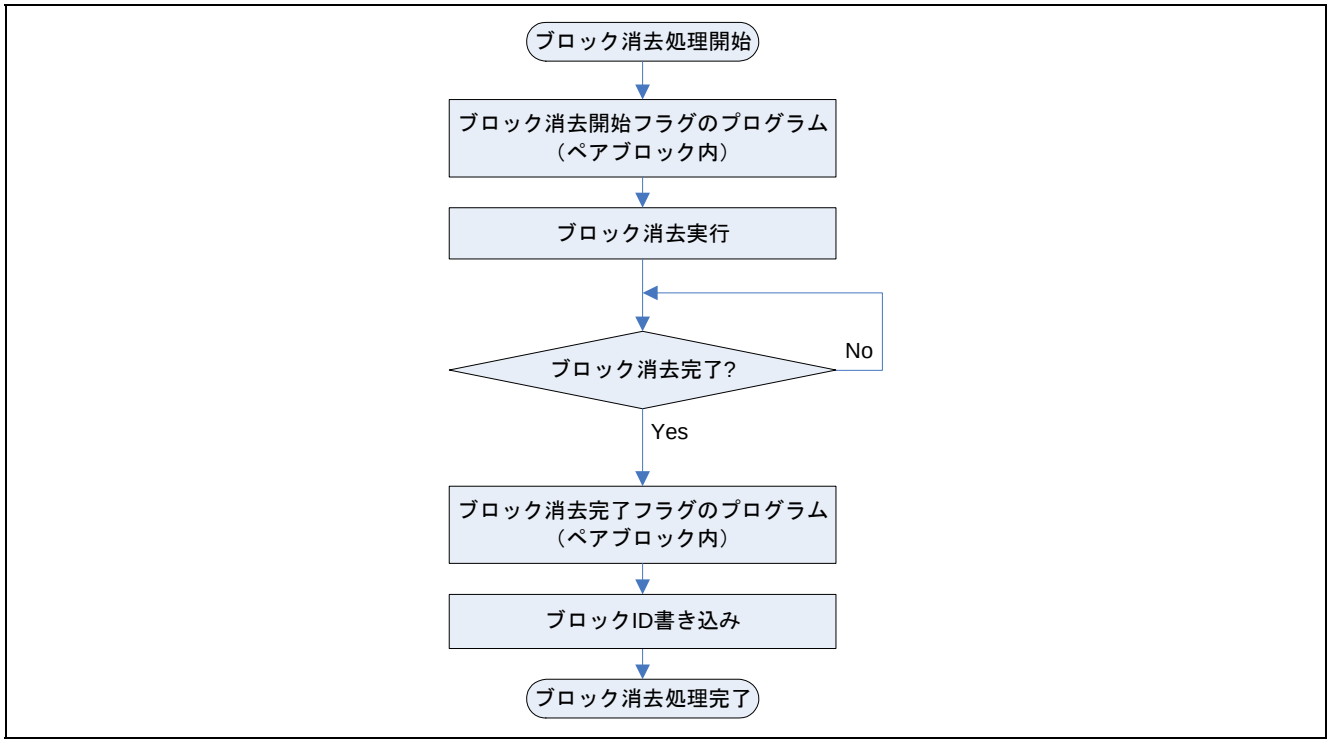

図 1.4 ブロック消去処理概略フロー

1.6.5 ブロックの状態判断

ドライバは、初期化時にすべてのブロック、および各ブロックの管理情報領域に対してブランクチェックを行い、表 1.2 のようにブロックの状態を判断します。

表 1.2 ブロックの状態

ブロック	blank	Non-blank						
ブロック ID	−	blank	Non-blank					
データリクレーム完了フラグ	−	−	blank		Non-blank			
ペアブロック用消去開始フラグ*1	−	−	Non-blank	blank	blank		Non-blank	
ペアブロック用消去完了フラグ*1	−	−			Non-blank	blank	blank	Non-blank
ブロックの状態	ブロック ID 未書き込み	未フォーマット状態		データ更新待機ブロック	未フォーマット状態	データ更新用ブロック		
ペアブロックの状態	*2	*3		データ更新用ブロック	*3	消去待機ブロック	消去中ブロック	*2

【注】 *1 各ブロックの消去開始／完了フラグは、ペアブロック内に格納されています。
*2 ペアブロック内のブロック管理情報からブロックの状態を判断します。
*3 ペアブロック内のブロック管理情報からブロックの状態を判断しますが、データフラッシュ内に未フォーマット状態のブロックが検出された場合には、データフラッシュ全体のフォーマット要求が戻り値として返されます。

1.7 データ管理情報

データの格納状態を示す情報です。本情報が格納される領域は、データが格納される領域に 1 対 1 で対応しており、有効なデータ管理情報に対応するデータ格納領域には有効なデータが格納されていることを示します（構成は「1.5　ブロック内データ構造」を、データの有効／無効に関しては「1.8　有効データの検索」を参照ください）。

データ管理情報は、データ更新開始フラグ、データ更新完了フラグで構成されており、各データは、必ず 1 つの有効なデータ管理情報を持っています。

1.7.1 データ更新開始フラグ

データ管理情報を更新する際に最初にプログラムされる情報です。ドライバは、このフラグの領域がノンブランク状態となっていた場合、その位置にデータ管理情報が存在すると判断します。また、後述のデータ更新完了フラグとの組み合わせにより、データ更新状態を判断します。

本情報内には、データ番号、データチェック用サムが含まれています。

(1) データ番号

ドライバによって付与される、データ管理用の論理的な番号です。

DF_user.c 内の配列（DF_DATA_SIZE_GROUP）でデータサイズ、およびデータの格納先ブロックグループを指定した配列メンバ順に"0"から昇順に割り振られます。データ番号は、1 バイトサイズのデータとして管理されます。

(2) データチェック用サム

ドライバによるデータ更新時に作成されるサム値です。更新するデータを 1 バイトごとに加算した値の下位 1 バイトとしており、ドライバ初期化中の有効データ検索時、データリクレーム中のエラー検出によるブロック内書き込み済みデータのチェック時に使用されます。

有効データ読み出し関数コールによるデータ読み出し時やデータリクレーム時には、本サム値によるデータのチェックは行われません。

1.7.2 データ更新完了フラグ

データの更新が完了した際にプログラムされる情報です。ドライバは、このフラグの領域がノンブランク状態となっていた場合、対応するデータ格納アドレスにデータのプログラムが正常に完了していると判断します。また、データ管理情報は、"アドレス大⇒小" の方向にアドレスをずらしながら更新されます。新しいデータ管理情報の更新が完了した時点で、古いデータ管理情報は論理的に無効になります。

1.7.3 データ更新の概略フロー

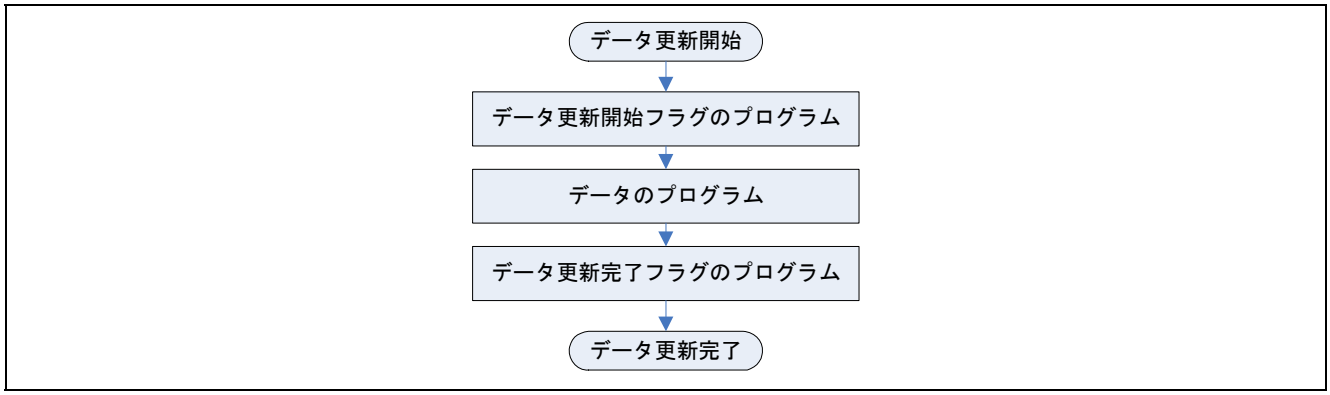

図 1.5　データ更新処理概略フロー

1.7.4 データ更新状態の判断

ドライバは、データ管理情報内の各フラグ状態から表 1.3 のようにデータ更新状態を判断します。

表 1.3　データ更新状態の判断

データ更新開始フラグ	blank		Non-blank	
データ更新完了フラグ	blank	Non-blank	blank	Non-blank
データの更新状態	未実行	無効	データ更新中	データ更新完了
状態の説明	本情報領域、および対応するデータ格納領域は未使用のため次回以降の更新時に使用可能。	データ管理情報領域が不正な状態のため本情報領域、および対応するデータ格納領域内データは無効。	データ管理情報書き込み中、またはデータ書き込み中のため本情報領域、および対応するデータ格納領域内データは無効。	データの更新が正常に完了し、本情報領域に対応するデータ領域内データは有効。

1.8 有効データの検索

ドライバは、初期化時にデータ番号ごとの有効なデータ管理情報を検索します。ドライバは、「表 1.3　データ更新状態の判断」により、データ更新完了の状態で最も小さいアドレスに格納されているデータ管理情報を有効なデータ管理情報と判断します。ドライバは、各データごとの有効なデータ管理情報に対応するデータ格納アドレスをドライバ内の RAM に保持します。ドライバはデータ読み出しを行う場合、RAM に保持されているデータ格納アドレスを基に、データフラッシュからデータを読み出します。

1.9 ブロック状態／データ格納イメージ

図 1.6 に示すドライバ設定例（DF_user.h, DF_user.c 内）で本ドライバを使用した際の、データフラッシュブロックグループ-A（ブロック-0、ブロック-1）を例にあげたブロック状態イメージ、およびデータ格納イメージを示します。

```
/*  データ数設定 */
#define DF_DATA_ID_NUM                  (3)             // データ数 = 3

/*  各データ番号ごとのデータサイズとデータを格納する領域（ブロックグループ）設定 */
const unsigned short DF_DATA_SIZE_GROUP[ DF_DATA_ID_NUM ][2] =
{
/* データ番号  0 */     1,          GROUP_A,        // データサイズ 1 バイト，ブロックグループ-A 内に格納
/* データ番号  1 */     129,        GROUP_A,        // データサイズ 129 バイト，ブロックグループ-A 内に格納
/* データ番号  2 */     256,        GROUP_A,        // データサイズ 256 バイト，ブロックグループ-A 内に格納
};
```

図 1.6　ドライバ設定例（DF_user.h, DF_user.c 内）

1.9.1 フォーマット後

前項図 1.6 に示すドライバ設定例で、データフラッシュのフォーマット処理を実行した直後のブロックグループ-A 内のブロック状態イメージ、およびデータ格納イメージを図 1.7 に示します。図中のデータ管理情報領域の各フラグの番号と同番号のデータ領域がそれぞれ対応します。

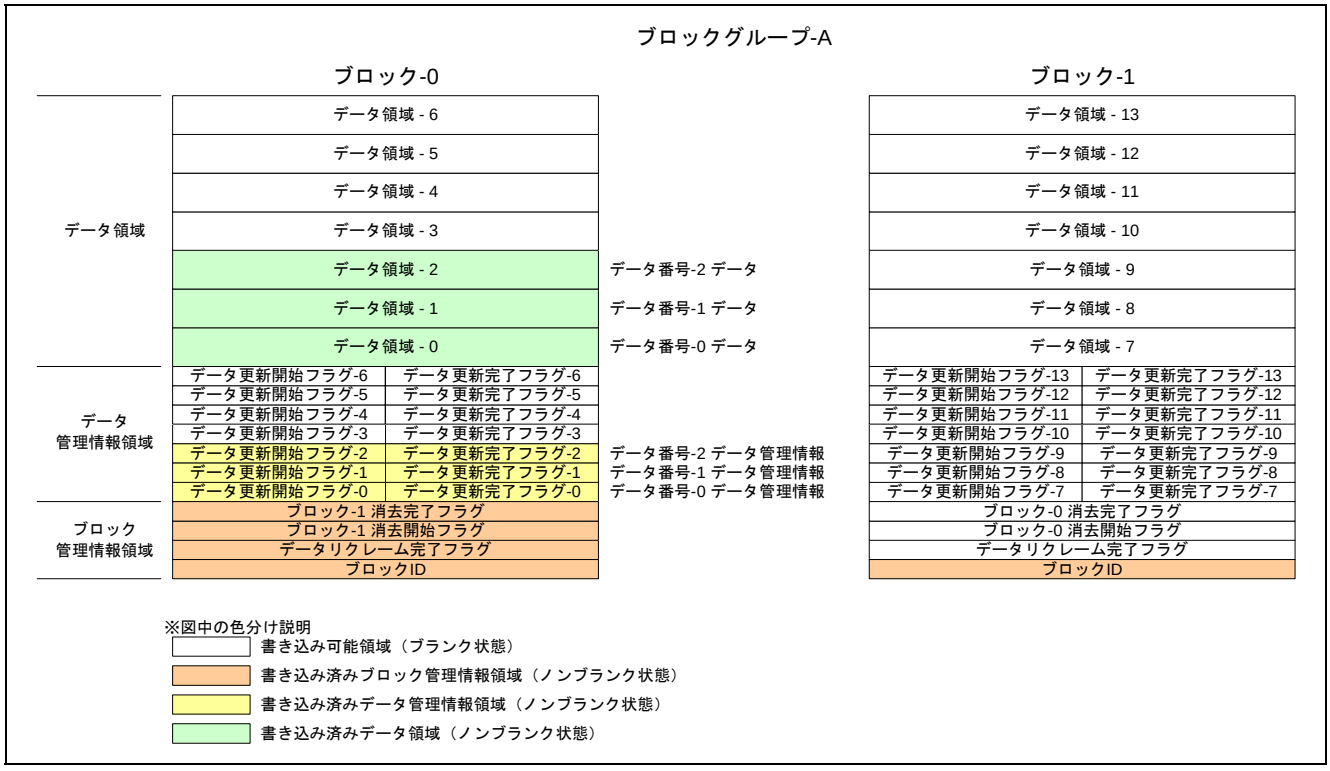

図 1.7　データフラッシュフォーマット直後のブロック状態およびデータ格納イメージ

ドライバは、データフラッシュフォーマット処理関数実行時、RAM 上のデータバッファ領域に格納されている値を全データ番号に対応するデータとして書き込みを行います。したがって、データフラッシュフォーマット処理関数実行後、データ更新処理関数によりデータを更新される前の各データ番号の有効データは不定値（直前に RAM 上のデータバッファ領域に格納されていた値）となります。

1.9.2 データ更新 － その 1（通常時）

前項図 1.7 の状態からデータ更新関数を使用してデータ番号 0 のデータを更新した場合の、ブロック状態イメージ、およびデータ格納イメージを図 1.8～図 1.11 に示します。

(1) ドライバは、ブロック-0 の書き込み可能なデータ管理情報領域（図 1.8 B）にデータ番号 0 のデータ更新開始フラグを書き込みます。

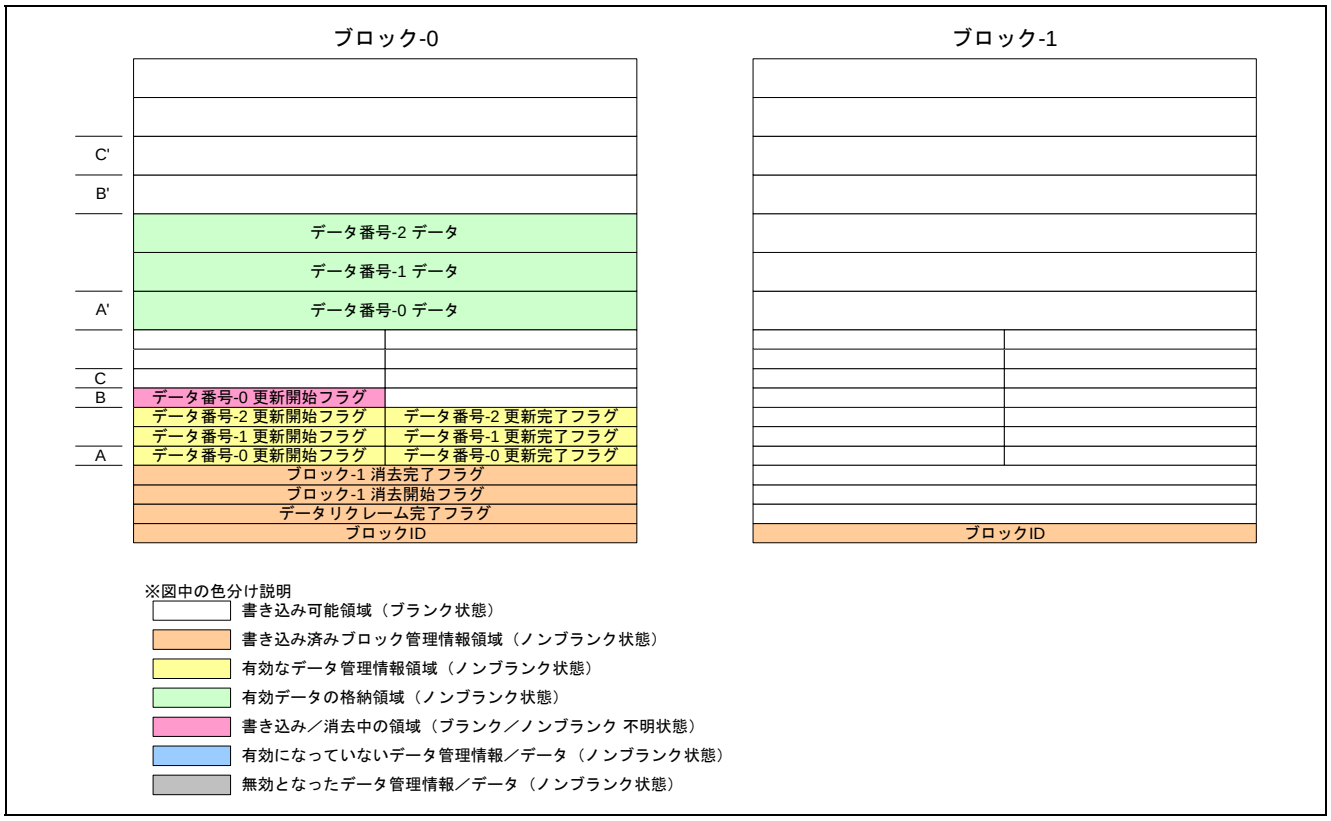

図 1.8 データ更新開始フラグのプログラム

図 1.8 の状態でシステムシャットダウンが発生した場合、ドライバは、次回ドライバ初期化時に A をデータ番号 0 の有効データ管理情報、A'を有効データとしたうえで、B のデータ更新開始フラグ領域のブランクチェック結果により、以下に示す(a)または(b)とします。

(a) B のデータ更新開始フラグ領域がまだブランクの状態であった場合、次回データ更新時に B のデータ管理情報領域、および B'のデータ領域を使用します。

(b) B のデータ更新開始フラグ領域がノンブランクの状態と判断された場合、B のデータ管理情報領域、および B'のデータ領域を無効な領域とし、次回データ更新時に C のデータ管理情報領域、および C'のデータ領域を使用します。

(2) データ更新開始フラグの書き込み完了後、対応するデータ領域（図 1.9 B'）にデータを書き込みます。

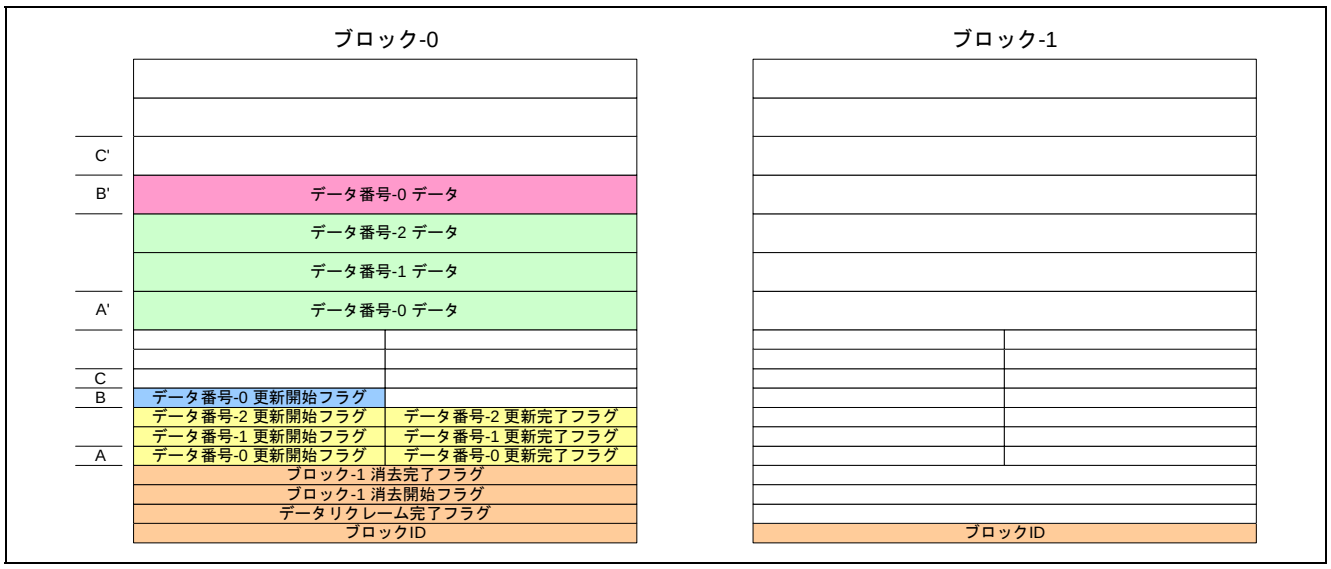

図 1.9　データのプログラム

図 1.9 の状態でシステムシャットダウンが発生した場合、ドライバは、次回ドライバ初期化時に A をデータ番号 0 の有効データ管理情報、A'を有効データとしたうえで、B のデータ管理情報領域、および B'のデータ領域を無効な領域とし、次回データ更新時に C のデータ管理情報領域、および C'のデータ領域を使用します。

(3) データの書き込み完了後、対応するデータ管理情報領域（図 1.10 B）にデータ更新完了フラグを書き込みます。

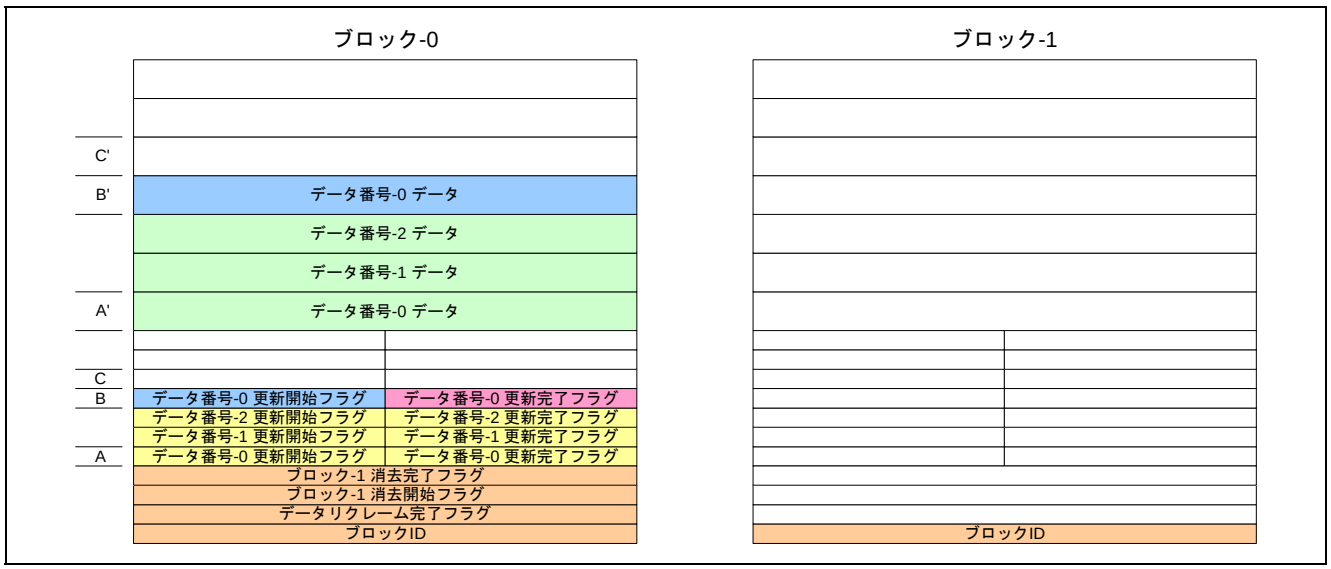

図 1.10　データ更新完了フラグのプログラム

図 1.10 の状態でシステムシャットダウンが発生した場合、ドライバは、次回ドライバ初期化時に B のデータ管理情報領域のブランクチェック結果により、以下に示す(a)または(b)とします。

(a) B のデータ管理情報領域内の少なくとも一方のフラグ領域がまだブランクの状態であった場合、A をデータ番号 0 の有効データ管理情報、A'を有効データとしたうえで、B のデータ管理情報領域、および B'のデータ領域を無効な領域とし、次回データ更新時に C のデータ管理情報領域、および C'のデータ領域を使用します。

(b) B のデータ管理情報領域がノンブランクの状態と判断された場合、B をデータ番号 0 の有効データ管理情報、B'をデータ番号 0 の有効データとしたうえで、A のデータ管理情報領域、および A'のデータ領域を無効な領域とし、次回データ更新時に C のデータ管理情報領域、および C'のデータ領域を使用します。本データ更新処理が完了した際のデータ格納イメージを図 1.11 に示します。

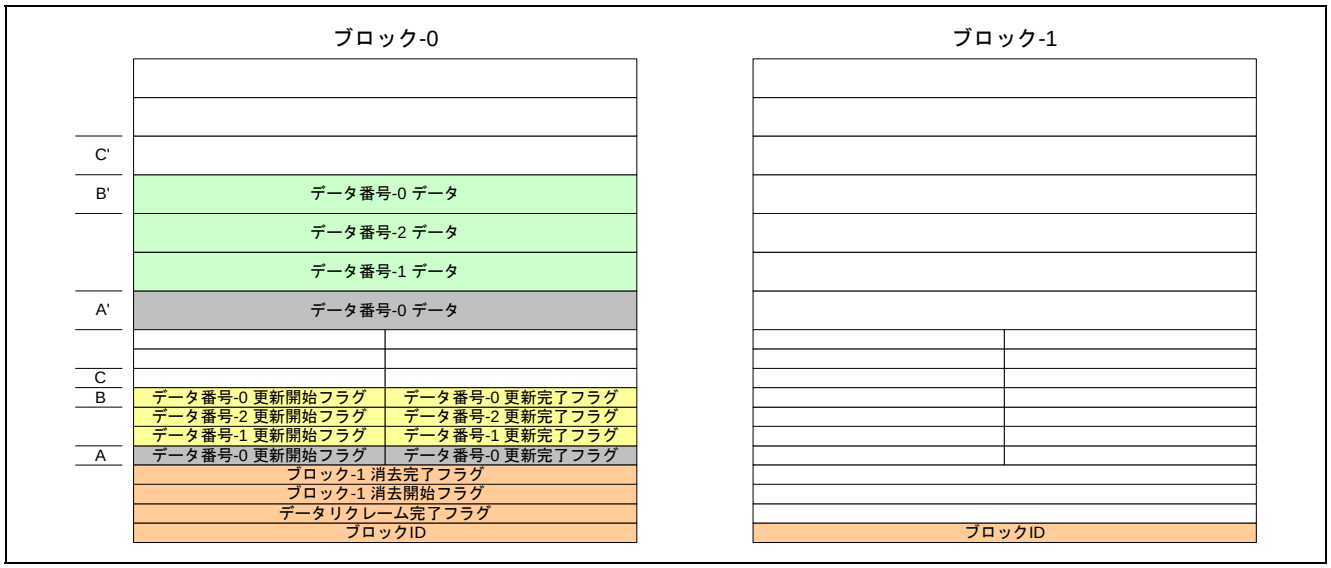

図 1.11　データ更新完了後のデータ格納イメージ

1.9.3 データ更新 － その 2（ブロック内に書き込み可能領域がない場合）

データ更新を継続していくと、データ更新に使用しているブロック（ブロック-0）内に書き込み可能な領域がない状態（図 1.12）となります。

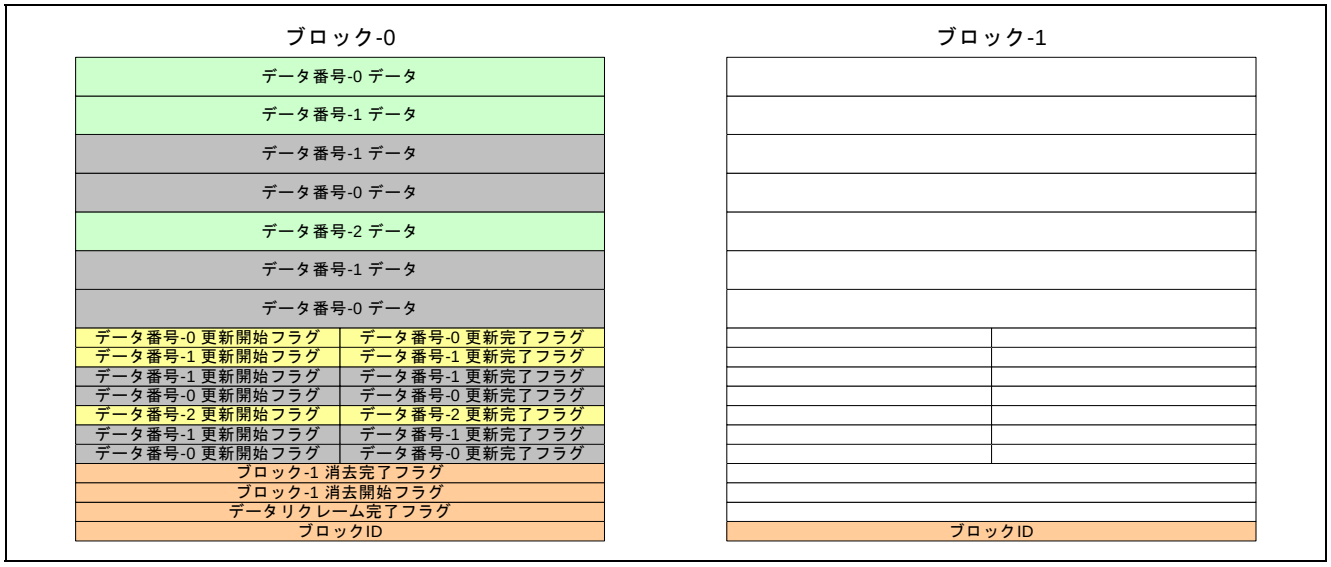

図 1.12 ブロック-0 内に書き込み可能領域がなくなった場合のブロック状態およびデータ格納イメージ

図 1.12 の状態からデータ番号 1 のデータを更新した場合の、ブロック状態イメージ、およびデータ格納イメージを図 1.13～図 1.16 に示します。

(1) ドライバは、ブロック-1 の書き込み可能なデータ管理情報領域（図 1.13 D）にデータ番号 1 のデータ更新開始フラグを書き込みます。

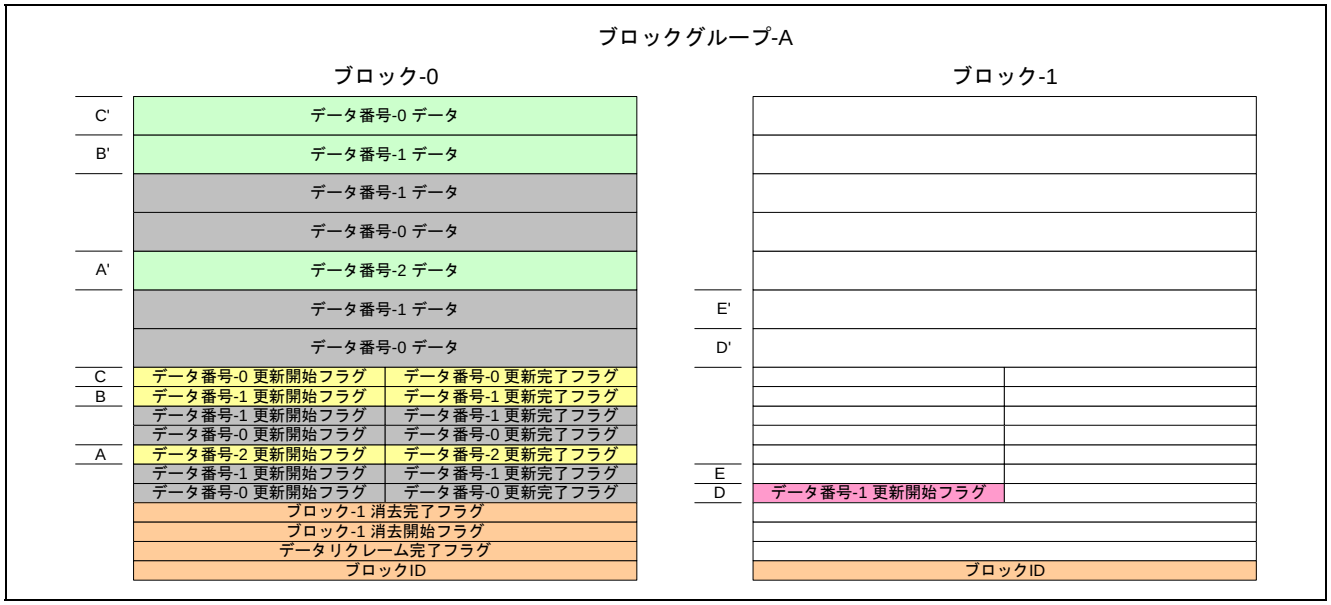

図 1.13　ブロック-0 内に書き込み可能領域がなくなった状態でのデータ更新開始フラグのプログラム

図 1.13 の状態でシステムシャットダウンが発生した場合、ドライバは、次回ドライバ初期化時にブロック-0 内の B をデータ番号 1 の有効データ管理情報、B'を有効データとしたうえで、ブロック-1 内 D のデータ更新開始フラグ領域のブランクチェック結果により、以下に示す(a)または(b)とします。

(a) D のデータ更新開始フラグ領域がまだブランクの状態であった場合、次回データ更新時に D のデータ管理情報領域、および D'のデータ領域を使用します。

(b) D のデータ更新開始フラグ領域がノンブランクの状態と判断された場合、D のデータ管理情報領域、および D'のデータ領域を無効な領域とし、次回データ更新時に E のデータ管理情報領域、および E'のデータ領域を使用します。

(2) データ更新開始フラグの書き込み完了後、対応するデータ領域（図 1.14 D'）にデータを書き込みます。

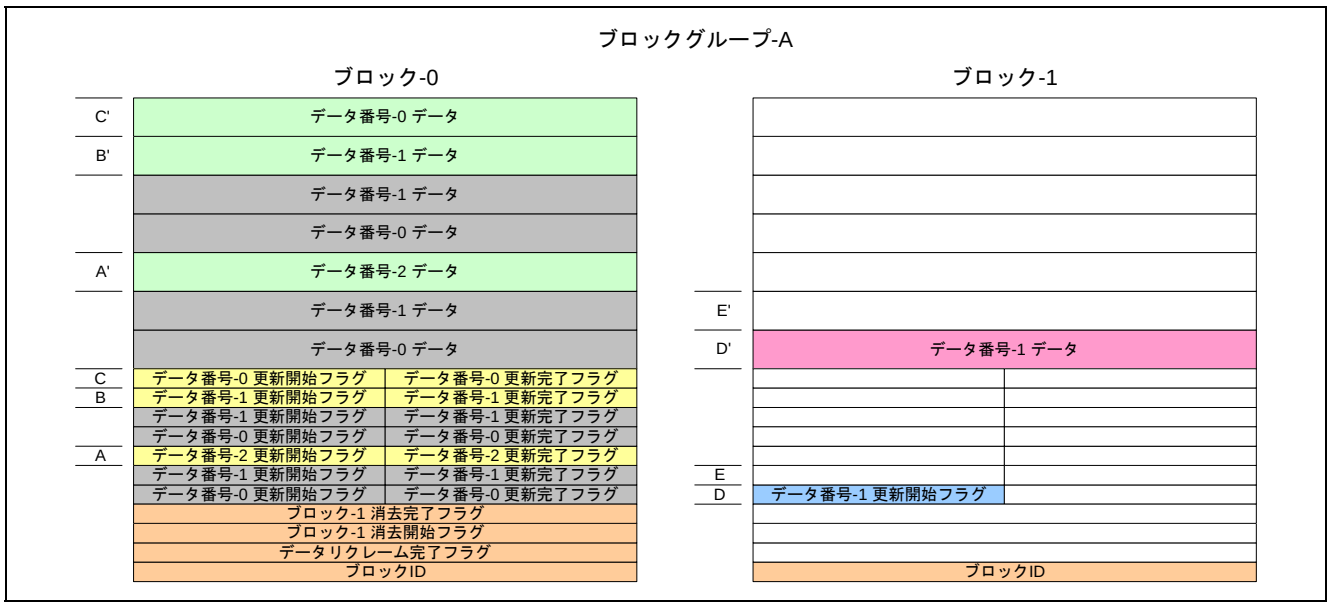

図 1.14　ブロック-0 内に書き込み可能領域がなくなった状態でのデータのプログラム

図 1.14 の状態でシステムシャットダウンが発生した場合、ドライバは、次回ドライバ初期化時に B をデータ番号 1 の有効データ管理情報、B'を有効データとしたうえで、D のデータ管理情報領域、および D'のデータ領域を無効な領域とし、次回データ更新時に E のデータ管理情報領域、および E'のデータ領域を使用します。

(3) データの書き込み完了後、対応するデータ管理情報領域（図 1.15 D）にデータ更新完了フラグを書き込みます。

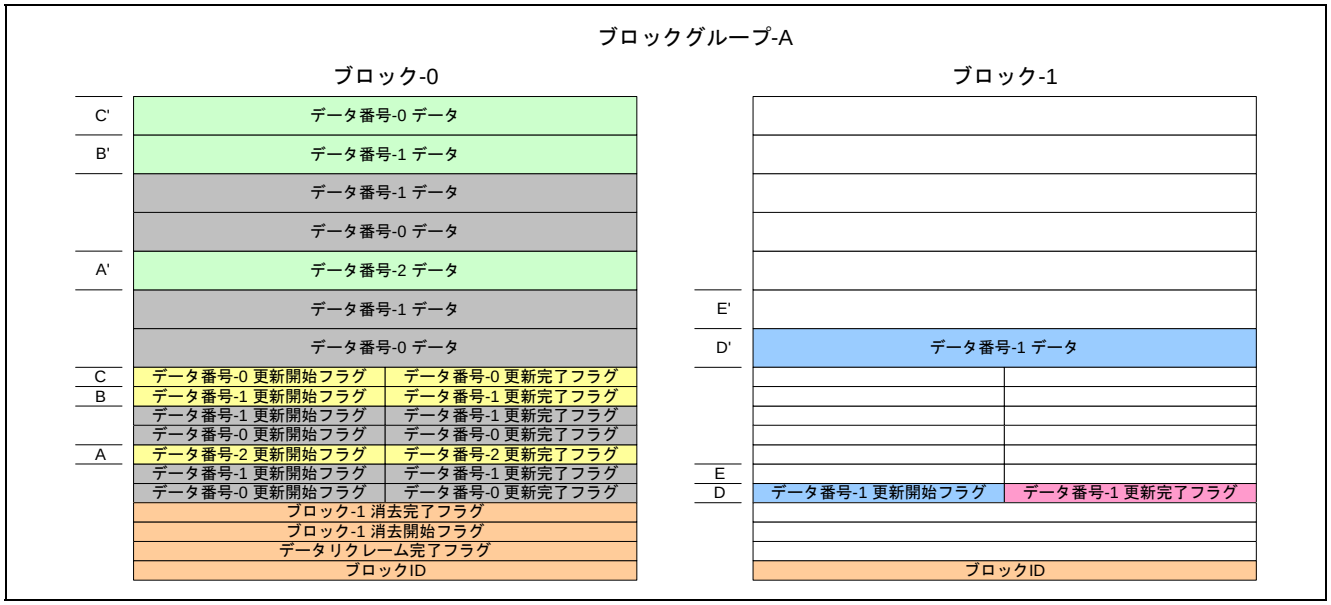

図 1.15　ブロック-0 内に書き込み可能領域がなくなった状態でのデータ更新完了フラグのプログラム

図 1.15 の状態でシステムシャットダウンが発生した場合、ドライバは、次回ドライバ初期化時に D のデータ管理情報領域のブランクチェック結果により、以下に示す(a)または(b)とします。

(a) D のデータ管理情報領域内の少なくとも一方のフラグ領域がまだブランクの状態であった場合、B をデータ番号 1 の有効データ管理情報、B'を有効データとしたうえで、D のデータ管理情報領域、および D'のデータ領域を無効な領域とし、次回データ更新時に E のデータ管理情報領域、および E'のデータ領域を使用します。

(b) D のデータ管理情報領域がノンブランクの状態と判断された場合、D をデータ番号 1 の有効データ管理情報、D'を有効データとしたうえで、B のデータ管理情報領域、および B'のデータ領域を無効な領域とします。本データ更新処理が完了した際のデータ格納イメージを図 1.16 に示します。

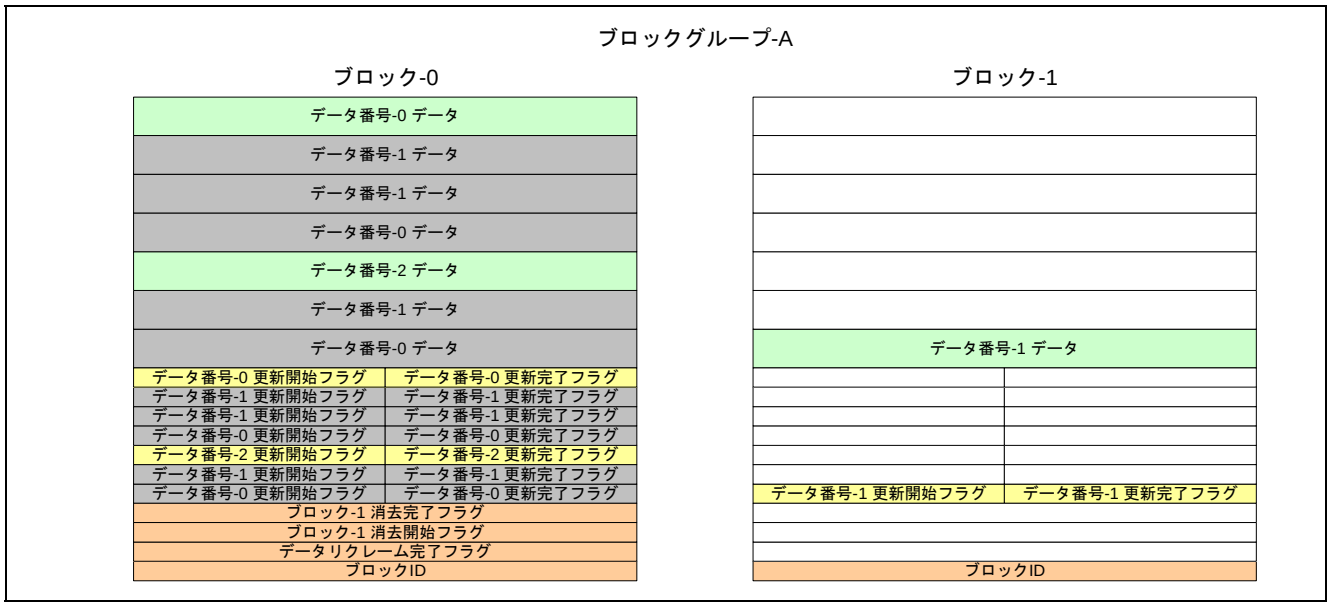

図 1.16　ブロック-0 内に書き込み可能領域がなくなった状態でのデータ更新完了後の格納イメージ

1.9.4　データリクレーム　－　その 1（通常時）

ドライバは、前項の（ブロック内に書き込み可能領域がない状態でのデータ更新）処理に引き続き、ブロック-0 内に存在する有効データをブロック-1 にコピーします。この処理をデータリクレーム処理と呼びます。データリクレーム処理は、データ更新関数内で自動的に実行され、データ更新処理と同等の手順により書き込みを行いますが、データ更新開始フラグ内のサム値の再演算は行わず、各領域のコピー処理となります。

データリクレーム時のブロック状態イメージ、およびデータ格納イメージを図 1.17～図 1.19 に示します。

(1) 図 1.17 の状態からブロック-0 内のデータ番号 0 の有効データ管理情報（C）をブロック-1 内 E に、有効データ（C'）を E'にコピーします。

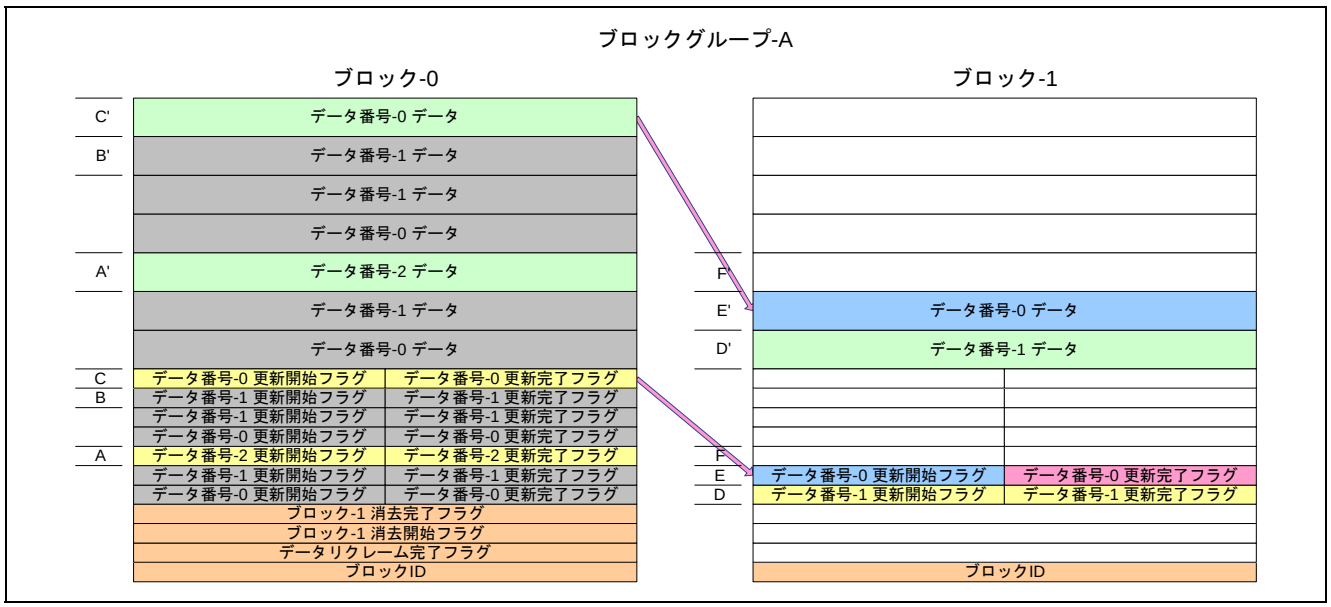

図 1.17　データリクレーム時のデータ格納イメージその 1

図 1.17 の状態でシステムシャットダウンが発生した場合、ドライバは、次回ドライバ初期化時に E のデータ管理情報領域のブランクチェック結果により、以下に示す(a)または(b)とします。

(a) E のデータ管理情報領域内の少なくとも一方のフラグ領域がまだブランクの状態であった場合、C をデータ番号 0 の有効データ管理情報、C'を有効データとしたうえで、E のデータ管理情報領域、および E'のデータ領域を無効な領域とし、次回データ更新時に F のデータ管理情報領域、および F'のデータ領域を使用します。F および F'へのデータ更新処理後、再度データリクレーム処理が実行されます。

(b) E のデータ管理情報領域がノンブランクの状態と判断された場合、E をデータ番号 0 の有効データ管理情報、E'を有効データとしたうえで、C のデータ管理情報領域、および C'のデータ領域を無効な領域とします。F および F'へのデータ更新処理後、再度データリクレーム処理が実行されます。

(2) 続けて同様にブロック-0 内のデータ番号 2 の有効データ管理情報（A）をブロック-1 内 F に、有効データ（A'）を F'にコピーします。

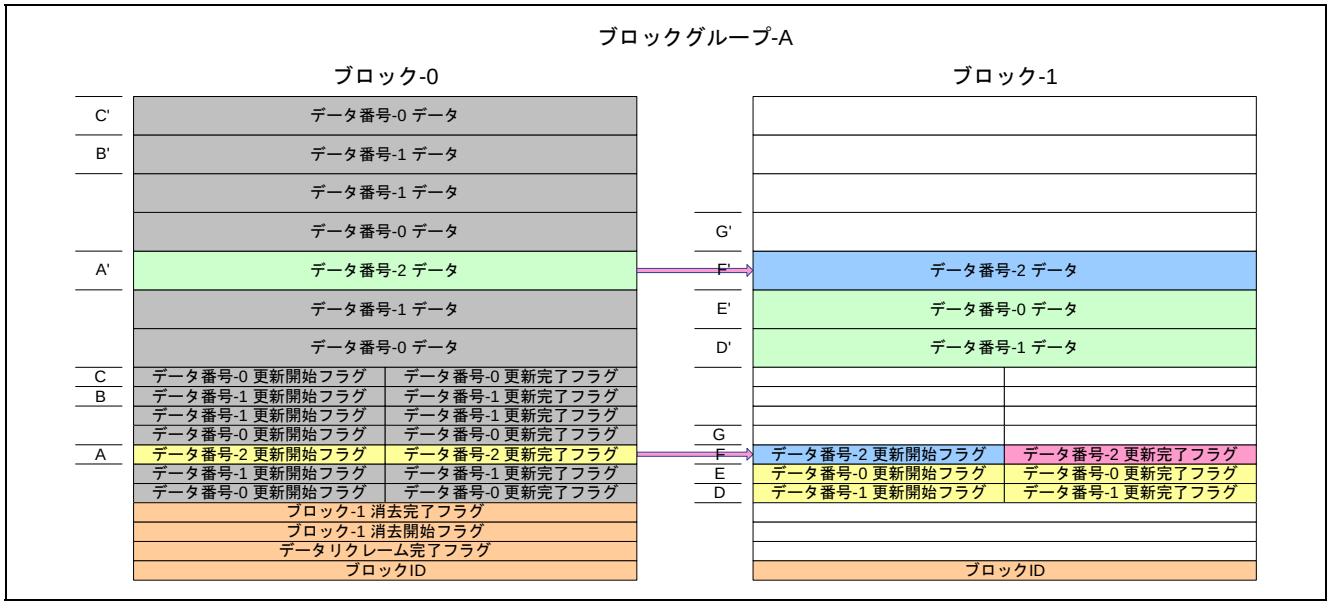

図 1.18　データリクレーム時のデータ格納イメージその 2

図 1.18 の状態でシステムシャットダウンが発生した場合、ドライバは、次回ドライバ初期化時に F のデータ管理情報領域のブランクチェック結果により、以下に示す(a)または(b)とします。

(a) F のデータ管理情報領域内の少なくとも一方のフラグ領域がまだブランクの状態であった場合、A をデータ番号 2 の有効データ管理情報、A'を有効データとしたうえで、F のデータ管理情報領域、および F'のデータ領域を無効な領域とし、次回データ更新時に G のデータ管理情報領域、および G'のデータ領域を使用します。G および G'へのデータ更新処理後、再度データリクレーム処理が実行されます。

(b) F のデータ管理情報領域がノンブランクの状態と判断された場合、F をデータ番号 2 の有効データ管理情報、F'を有効データとしたうえで、A のデータ管理情報領域、および A'のデータ領域を無効な領域とします。このとき、ブロックグループ-A 内のすべての有効データがブロック-1 内に集約されたことになり、ドライバは、ブロック-1 のデータリクレーム完了フラグを書き込み、ブロック-0 を消去対象ブロックとして登録します。（図 1.19 と同様）

(3) データリクレーム処理が完了（ブロックグループ-A 内のすべての有効データがブロック-1 内に集約）すると、ドライバは、ブロック-1 のデータリクレーム完了フラグを書き込み、ブロック-0 を消去対象ブロックとして登録します。

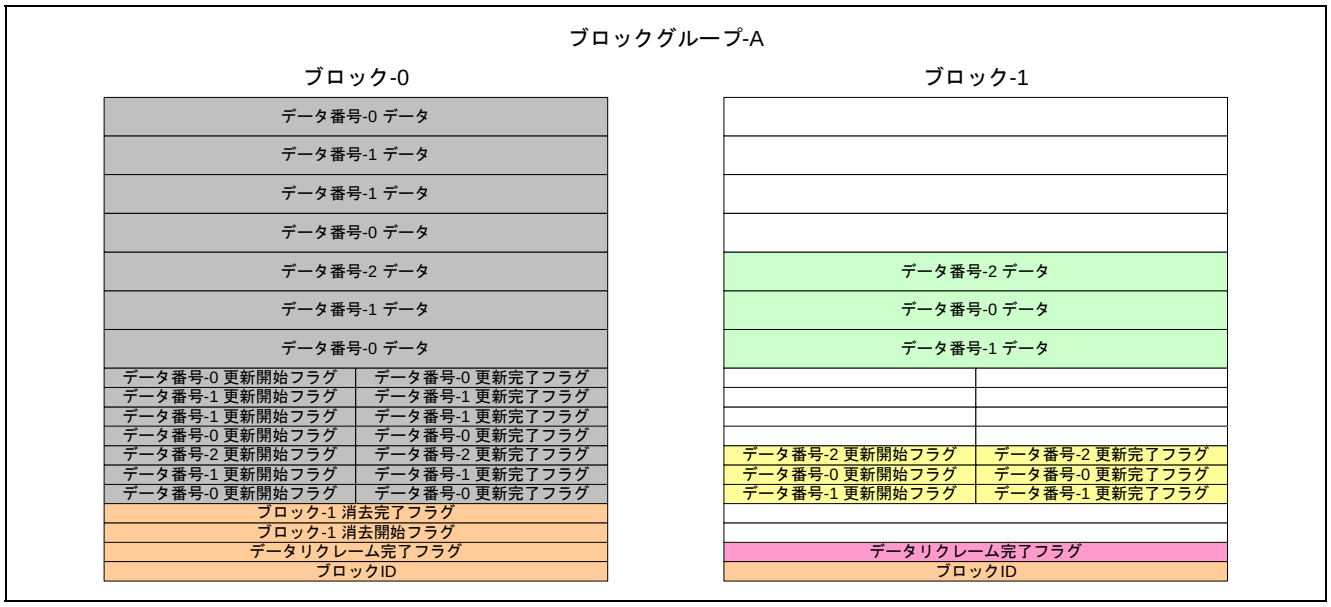

図 1.19　データリクレーム完了フラグのプログラム

図 1.19 の状態でシステムシャットダウンが発生した場合、ドライバは、次回ドライバ初期化時にブロック-1 のデータリクレーム完了フラグ領域がブランクであった場合は、データリクレーム完了フラグの書き込みを行ったうえで、ブロック-0 を消去対象ブロックとして登録します。

1.9.5　　ブロック消去　－　その 1（通常時）

前項 図 1.19 の状態からブロック消去関数を使用してブロック-0 を消去した場合の、ブロック状態イメージ、を図 1.20～図 1.22 に示します。

(1) ドライバは、ブロック-1 内の消去開始フラグを書き込みます。

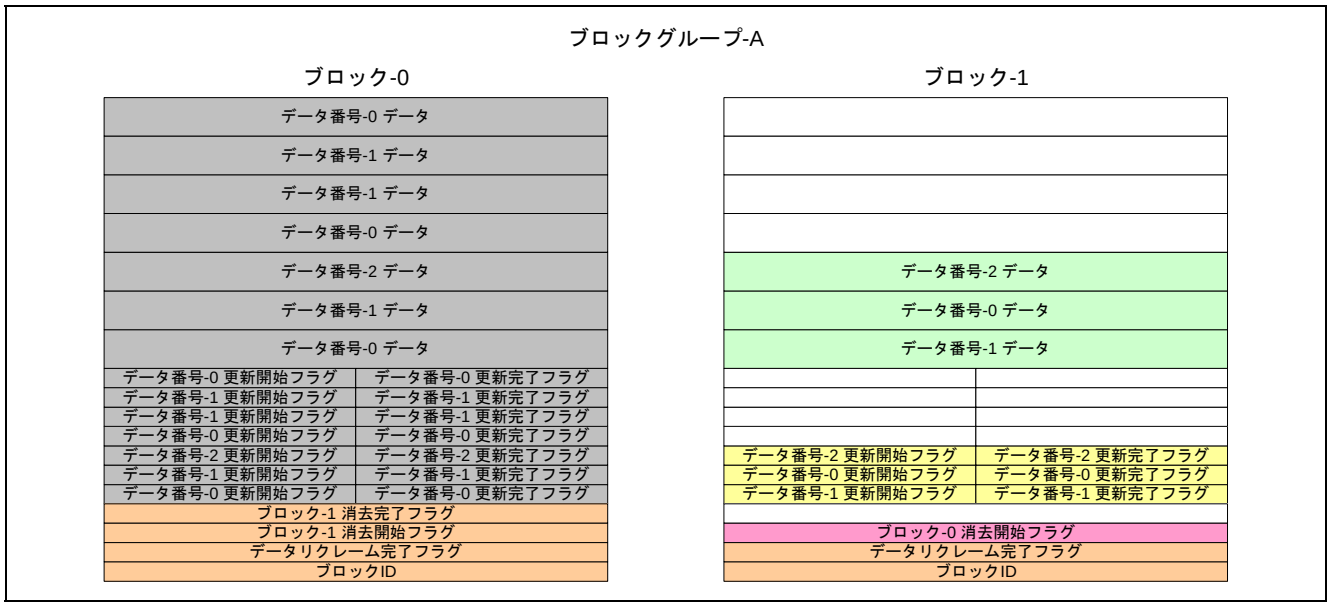

図 1.20　消去開始フラグのプログラム

図 1.20 の状態でシステムシャットダウンが発生した場合、ドライバは、次回ドライバ初期化時にブロック-0 を消去対象ブロックとして登録します。ブロック消去関数のコールによりブロック消去処理が再度実行されます。その際、ブロック-1 内の消去開始フラグ領域がノンブランクであった場合、消去開始フラグの書き込み処理は省略されます。

(2) 続けてブロック-0 を消去します。

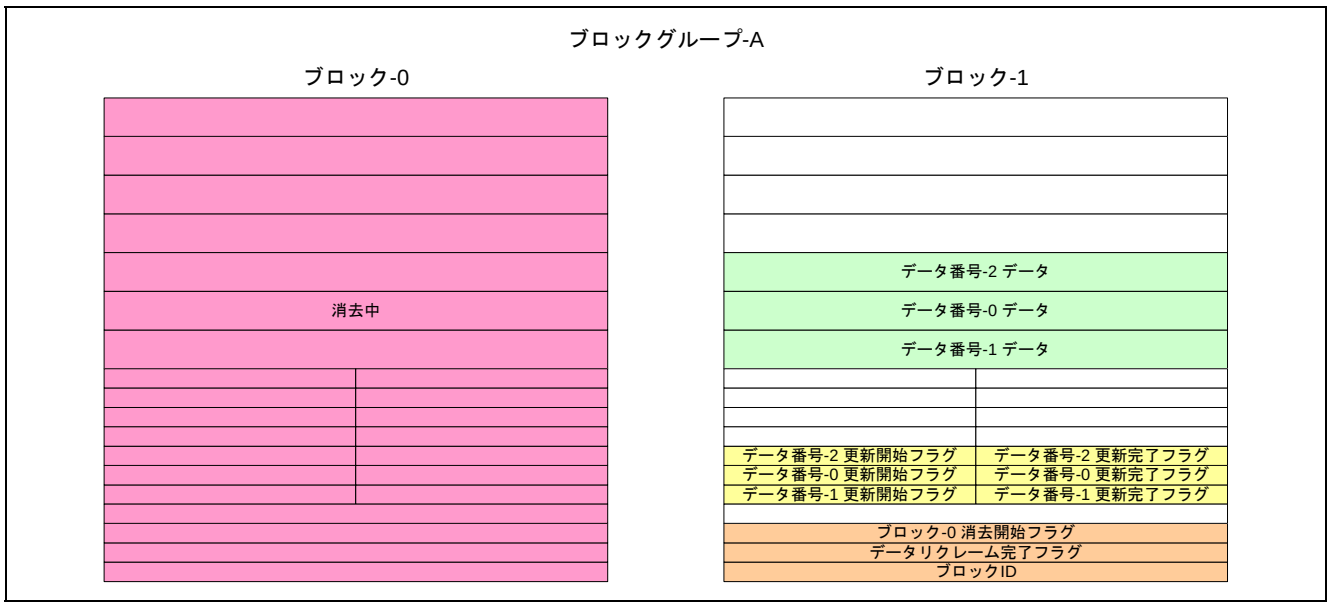

図 1.21　ブロック消去

図 1.21 の状態でシステムシャットダウンが発生した場合、ドライバは、次回ドライバ初期化時にブロック-0 のブランクチェック結果により、以下に示す(a)または(b)とします。

(a) ブロック-0 がブランクの状態であった場合、ドライバは、消去処理が完了したものと判断し、ブロック-1 内の消去完了フラグの書き込み（図 1.22 と同様）、およびブロック-0 のブロック ID の書き込み（図 1.23 と同様）を行い、ブロック-0 のブロック消去処理を完了します。
(b) ブロック-0 がまだノンブランクの状態であった場合、ドライバは、消去処理が中断したものと判断し、ブロック-0 を消去対象ブロックとして登録します。ブロック消去関数のコールによりブロック消去処理が再度実行されます。その際、ブロック-1 内の消去開始フラグの書き込み処理は省略されます。

(3) ブロック-0 の消去完了後、ブロック-1 内の消去完了フラグを書き込みます。

図 1.22　消去完了フラグのプログラム

図 1.22 の状態でシステムシャットダウンが発生した場合、ドライバは、次回ドライバ初期化時にブロック-1 内の消去完了フラグのブランクチェック結果により、以下に示す(a)または(b)とします。

(a) ブロック-1 内の消去完了フラグがブランクの状態であった場合、ドライバは、消去完了フラグの書き込み処理中に中断したものと判断し、ブロック-1 内の消去完了フラグの書き込みを行います。続いてブロック-0 のブロック ID の書き込み（図 1.23 と同様）を行い、ブロック-0 のブロック消去処理を完了します。

(b) ブロック-1 内の消去完了フラグがまだノンブランクの状態であった場合、ドライバは、消去完了フラグの書き込み後、ブロック-0 のブロック ID 書き込み処理中に中断したものと判断し、ブロック-0 のブロック ID の書き込み（図 1.23 と同様）を行い、ブロック-0 のブロック消去処理を完了します。

(4) 続けてブロック-0 のブロック ID を書き込みます。

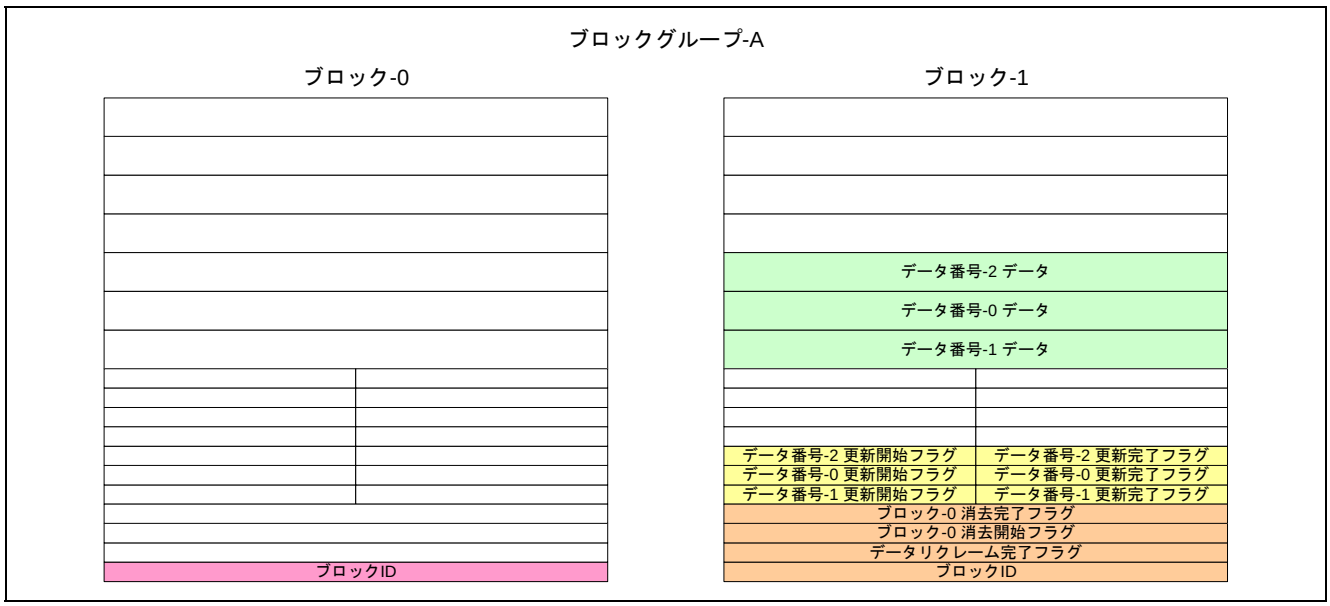

図 1.23　ブロック ID のプログラム

図 1.23 の状態でシステムシャットダウンが発生した場合、ドライバは、次回ドライバ初期化時にブロック-0 のブランクチェック結果により、以下に示す(a)または(b)とします。

(a) ブロック-0 がブランクの状態であった場合、ドライバは、ブロック ID の書き込み処理中に中断したものと判断し、ブロック-0 のブロック ID の書き込みを行い、ブロック-0 のブロック消去処理を完了します。
(b) ブロック-0 がノンブランクの状態であった場合（ブロック-0 のブロック ID 領域がノンブランクの状態であった場合）、ドライバは、ブロック-0 のブロック消去処理を完了します。

1.9.6 データリクレーム － その 2（コピー領域が不足状態となる場合）

ドライバは、前述のとおり、データ更新関数処理中、およびデータリクレーム処理中にシステムシャットダウンが発生した場合等には、各処理が完了していない場合ですでに書き込みがされている（ノンブランク）データ管理情報領域、およびそれに対応するデータ領域を無効な領域とし、次回のデータ更新に使用する書き込み領域を "アドレス大⇒小" の方向にずらします。

ブロック-0 内にデータ書き込み可能領域がない状態でデータ番号 1 のデータ更新を実行し、その更新処理中にシステムシャットダウンが発生した場合の、次回ドライバ初期化後のブロック状態イメージ、およびデータ格納イメージを図 1.24 に示します。

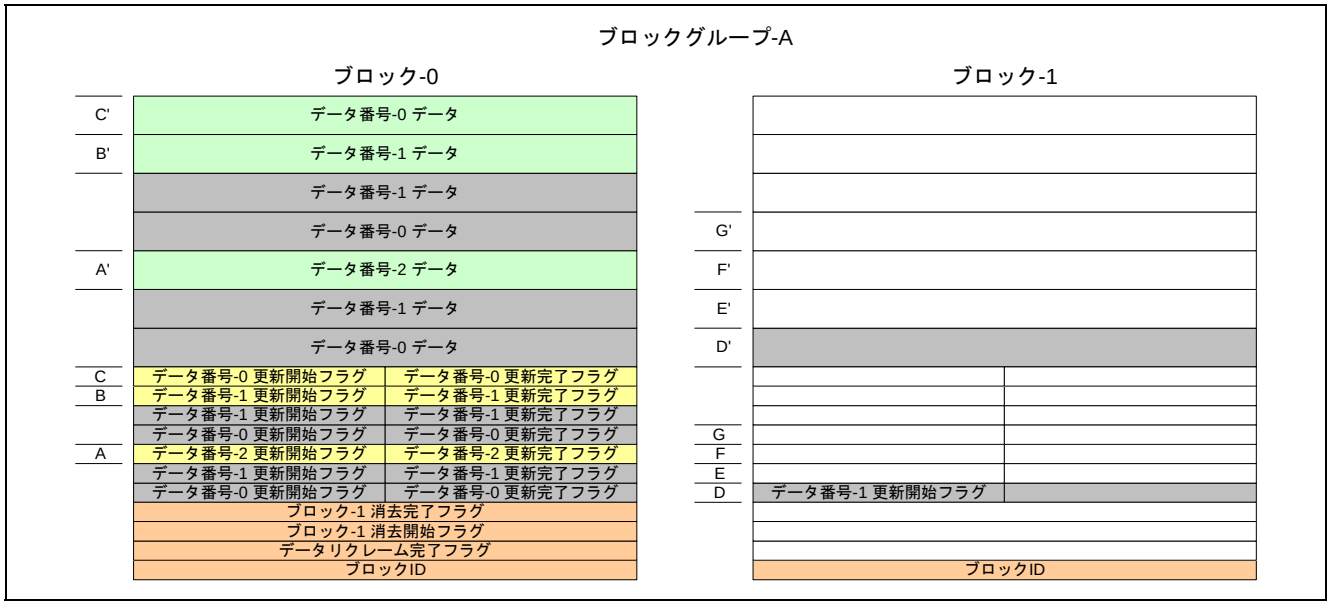

図 1.24 データ更新中断時のデータ格納イメージその 1（コピー領域が足りている状態）

図 1.24 の状態で再度データ番号 1 のデータ更新を実行する場合、データ番号 1 の更新データは E'に書き込まれ、その後ドライバにより自動的に実行されるデータリクレーム処理によってデータ番号 0 のリクレームデータは F'に、データ番号 2 のリクレームデータは G'に書き込まれることになります（データ管理情報もそれぞれに対応した領域（E、F、G）に書き込まれます）。

図 1.24 の状態から以下(a)～(d)に示すような前述と同様の処理中断条件が重なった場合の、次回ドライバ初期化時のブロック状態イメージ、およびデータ格納イメージを図 1.25 に示します。

(a) データ番号 1 のデータ更新完了（図 1.25 E、E'）
(b) (a)に継続して行われるデータ番号 0 のデータリクレーム処理中（データ書き込み中）にシステムシャットダウン（図 1.25 F、F'）
(c) データ番号 0 のデータ更新処理中（更新開始フラグ書き込み中）にシステムシャットダウン（図 1.25 G）
(d) データ番号 0 のデータ更新処理中（データ書き込み中）にシステムシャットダウン（図 1.25 H'）

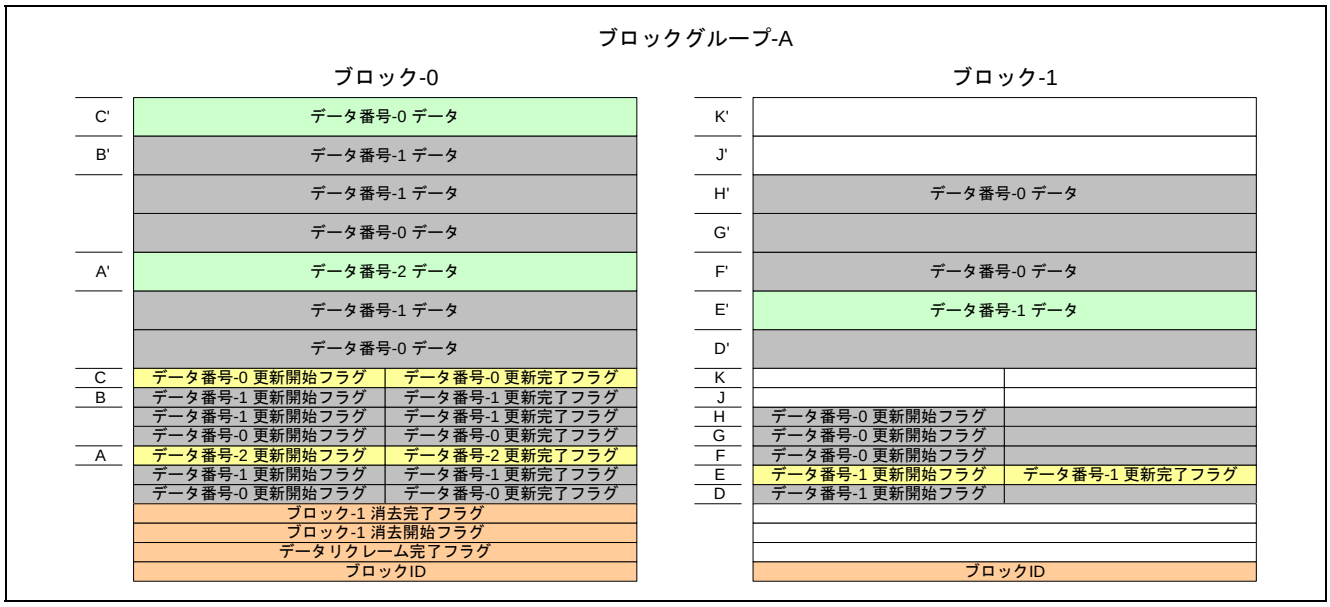

図 1.25　データ更新中断時のデータ格納イメージその 2（コピー領域が不足の状態）

図 1.25 の状態となった場合、ブロック-0 内に残されている有効データは、データ番号 0、2 の 2 つ、それに対しブロック-1 内の書き込み可能領域も J（J'）、K（K'）の 2 つとなり、リクレーム処理を必要とするデータ数＝書き込み可能領域数の関係となります。また、データリクレーム処理は、データ更新処理実行後に継続して行われるため、仮に、この状態でデータ番号 1 のデータ更新要求があった場合、J（J'）にデータ番号 1 のデータを更新すると、ブロック-1 内の残りの書き込み可能領域が K（K'）のみとなり、ブロック-0 内に残されている有効データ（データ番号 0、2）のすべてをブロック-1 内にコピーすることができなくなってしまいます。結果として、ブロックグループ-A 内の有効データがブロック-0、1 の双方に残ったまま書き込み領域がなくなり、同ブロックグループ内で有効データを保持したままでの、以後のデータ更新が不可能となってしまいます。

そこで、ドライバは、ブロックグループ-A 内のすべての有効データがブロック-1 内に集約できない可能性がある（リクレーム処理を必要とするデータ数＝書き込み可能領域数の状態）と判断した場合、ブロック-1 内にすでに更新、またはコピーされた有効データ（図 1.25 E'）をすべて無効なデータとし、更新、またはコピー前のブロック-0 内のデータ（図 1.25 A'、B'、C'）を有効データとしたうえで、ブロック-1 を消去対象ブロックとして登録します。

【注】　上述の例では、データ番号 1 は、1 度有効となったブロック-1 内の最新のデータが無効となり、すでに無効となっていたブロック-0 内の古いデータが有効データに戻ります。

その後、ブロック-1 のブロック消去をすることにより「1.9.3　データ更新－その 2（ブロック内に書き込み可能領域がない場合）」内 図 1.12 の状態に戻したうえで、以後のデータ更新、データリクレーム処理が可能となります。

【注】　このときの、ブロック-1 消去時のブロック消去処理関数内では、ブロック-0 内の消去開始フラグのプログラム処理、および消去完了フラグのプログラム処理は省略されます。

1.9.7　データ更新　－　その 3（ブロック消去処理中のデータ更新）

ブロック消去中に割り込みによりデータの更新処理関数がコールされた場合、基本的にはデータの更新処理が優先され、その間ブロック消去処理はサスペンド状態となります。

【注】 消去開始フラグ／消去完了フラグのプログラム中、およびブロック ID のプログラム中にデータの更新処理関数がコールされた場合、ドライバは、ビジー状態であること（DF_ST_WARNING_DRIVER_BUSY）を戻り値として返し、消去処理を継続します。タイミングをずらし、再度データ更新処理関数をコールしてください。

「1.9.5　ブロック消去－その 1（通常時）」内 図 1.21 に示されるブロック消去処理状態中に、データ番号 2 のデータ更新のため、割り込みによりデータの更新処理関数がコールされた場合のブロック状態イメージ、およびデータ格納イメージを図 1.26 に示します。

(1) ドライバは、ブロック-0 の消去処理をサスペンド状態にします。

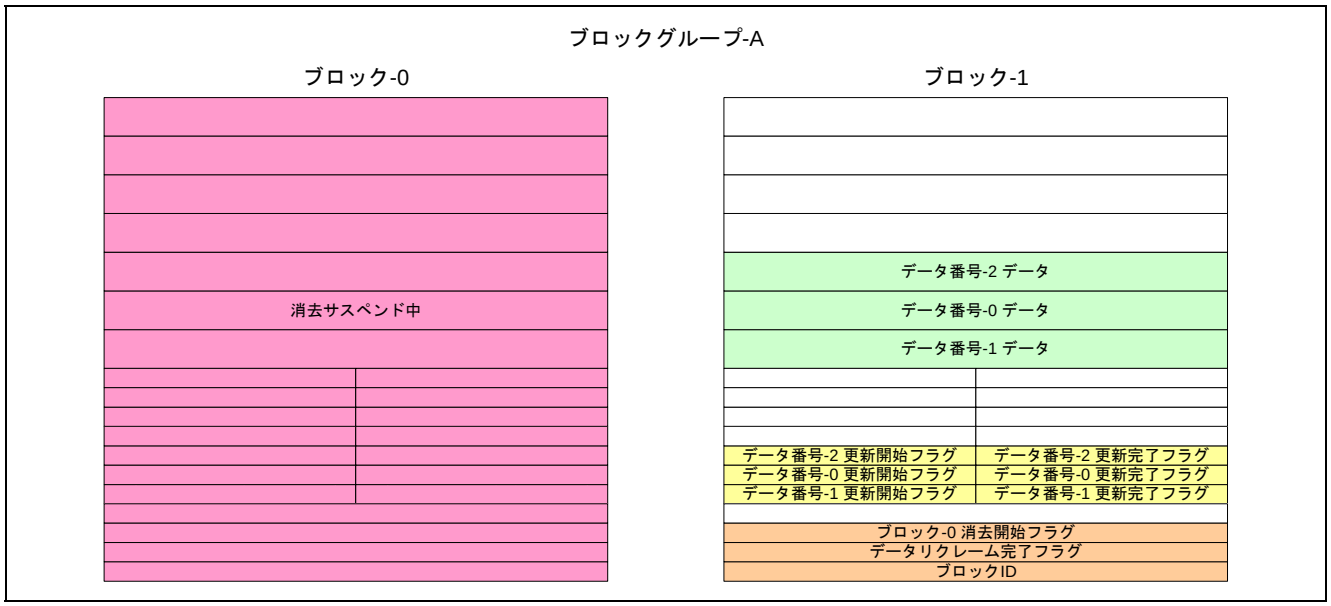

図 1.26　ブロック消去中のデータ更新イメージその 1（消去サスペンド）

図 1.26 の状態でシステムシャットダウンが発生した場合、データ番号 2 のデータ更新処理は中止され、ドライバは、次回ドライバ初期化時にブロック-0 を消去対象ブロックとして登録します。ブロック消去関数のコールによりブロック消去処理が再度実行されます。その際、ブロック-1 内の消去開始フラグの書き込み処理は省略されます。

(2) 続けて、ブロック-1 の書き込み可能領域（図 1.27 B、B'）にデータ番号 2 のデータ更新開始フラグ、データ、データ更新完了フラグを順に書き込みます。（通常のデータ更新時と同様）

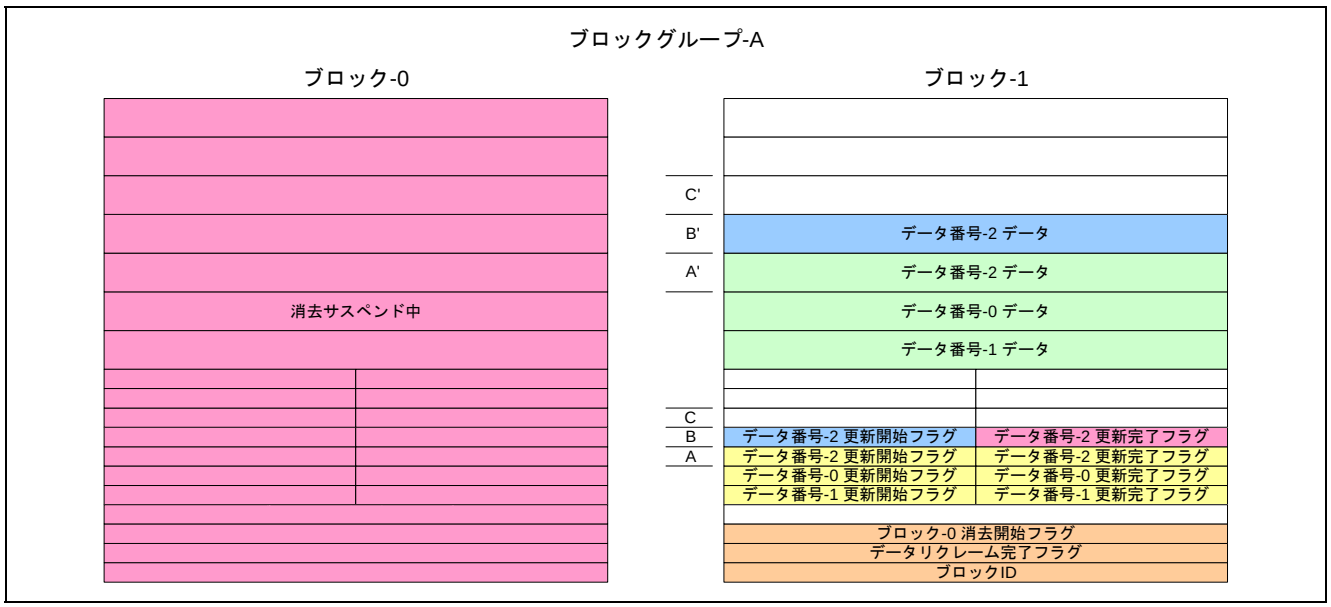

図 1.27　ブロック消去中のデータ更新イメージその 2（更新データのプログラム）

図 1.27 の各プログラム処理中にシステムシャットダウンが発生した場合の次回ドライバ初期化時の判断、および処理は、「1.9.2　データ更新－その 1（通常時）」の判断、および処理と同様になります。
また、ブロック-0 を消去対象ブロックとして登録し、次回ブロック消去関数コールによるブロック消去処理再実行時には、ブロック-1 内の消去開始フラグの書き込み処理は省略されます。

(3) ドライバは、データ更新の完了後、レジューム処理によりブロック-0 の消去サスペンド状態を解除し、消去処理を自動的に再開します。

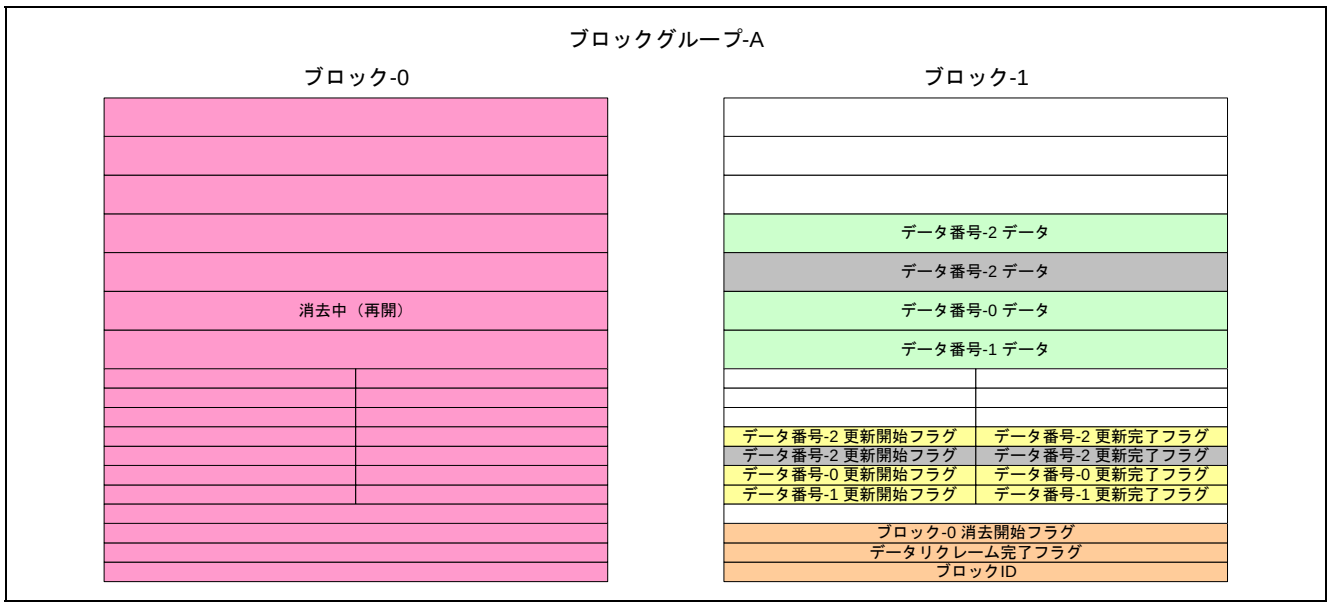

図 1.28　ブロック消去中のデータ更新イメージその 3（消去再開）

図 1.28 の状態以後は、「1.9.5　ブロック消去－その 1（通常時）」と同様の処理となります。

1.9.8　　データ更新　－　その 4（ブロックグループ内に書き込み可能領域がない場合）

ブロックグループ内に書き込み可能領域がない状態でデータの更新関数がコールされると、ドライバは、書き込み領域がないこと（DF_ST_WARNING_NO_BLANK_AREA）を戻り値として返します。

書き込み領域がない状態となる主な要因を以下に示します。

(1) 「1.9.4　データリクレームーその 1（通常時）」の図 1.19 の状態となってから、ブロック消去処理関数コールによるブロック消去を行わず、データ更新をし続けた場合。（図 1.29）

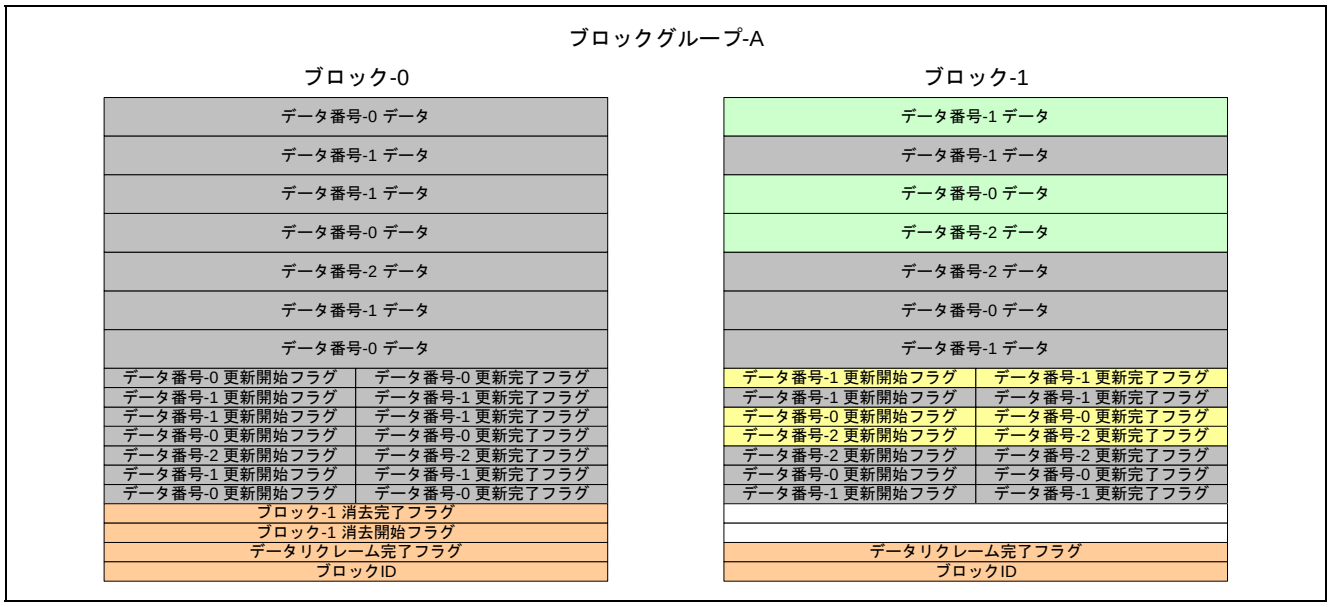

図 1.29　無効データ（ブロック）未消去により書き込み可能領域がない状態のイメージ

(2) 「1.9.6　データリクレームーその 2（コピー領域が不足状態となる場合）」の図 1.25 の状態と判断された場合。

(3) (2)の状態からその後のブロック消去完了までの状態。

(4) 「1.9.7　データ更新－その 3（ブロック消去処理中のデータ更新）」に示すデータ更新処理が重複し、ブロック-0 の消去完了までにブロック-1 内の書き込み可能領域がなくなった場合。（図 1.30）

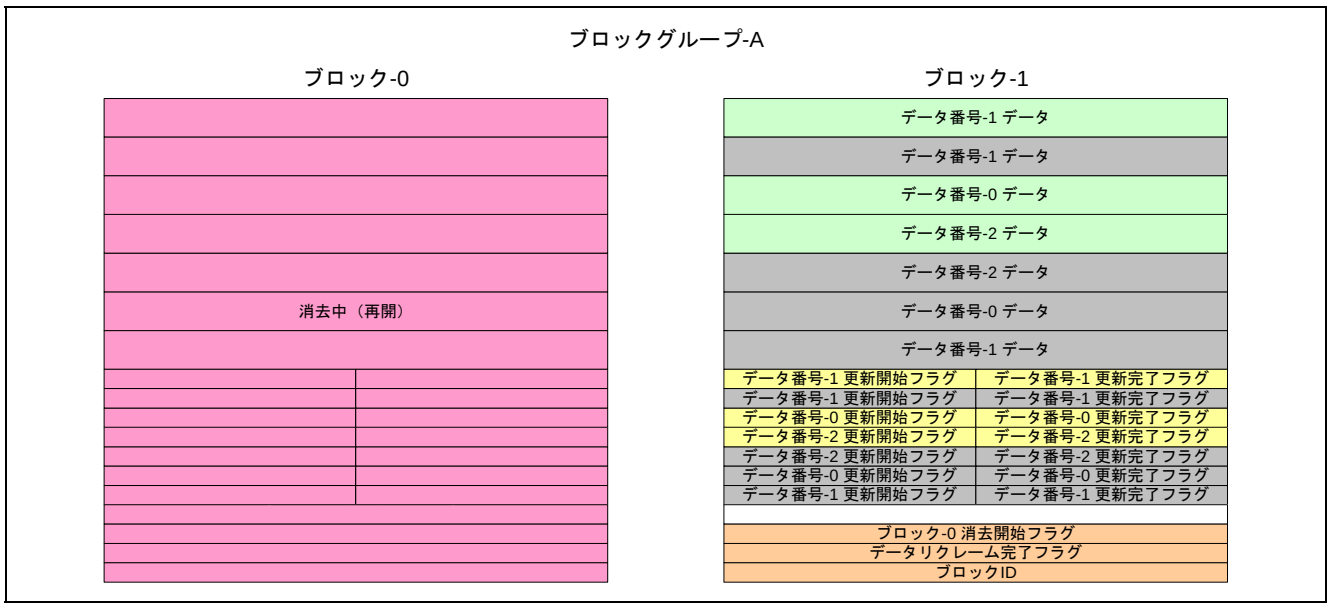

図 1.30　消去中に書き込み領域がなくなった状態のイメージ

(5) その他エラーやシステムシャットダウン等により、一方のブロックの消去が完了していない状態でもう一方のブロック内の書き込み可能領域がなくなった場合。

1.10 ユーザ API 関数

データフラッシュ・ドライバソフトウェアサンプルのユーザ API 関数一覧を表 1.4 に、関数機能表で使用されている型定義一覧を表 1.5 に示します。

表 1.4 ユーザ API 関数一覧

関数名	ラベル	機能概略説明
ドライバ初期化処理関数	DF_Drv_Init	データフラッシュドライバの初期化を行います。データフラッシュドライバを使用する場合、システム起動時に必ずコールしてください。
データフラッシュフォーマット処理関数	DF_Format	データフラッシュ上に、本ドライバにより制御するためのデータ構造を構築します。
データ更新処理関数	DF_Write_Data	引数で指定されたデータ番号のデータを更新します。データリクレーム処理が必要な場合、データリクレーム処理を行います。
有効データ読み出し処理関数	DF_Read_Data	引数で指定されたデータ番号のデータフラッシュ上に格納されている有効データを読み出し、引数で指定された格納ポインタに格納します。
ブロック消去処理関数	DF_Erase_block	消去対象ブロックとして登録されているブロックの消去を行います。
エラー処理関数	DF_Error_Management	エラーの状態に応じた処理を行います。
データサイズ取得関数	DF_Data_Size	引数で指定されたデータ番号のデータサイズを戻り値として返します。
書き込み可能領域数取得関数	DF_Blank_Number	引数で指定されたブロックグループの、データ更新に使用しているブロック内の書き込み可能領域数を戻り値として返します。
データフラッシュ読み出しプロテクト処理関数	DF_Read_Protect	データフラッシュの読み出しをプロテクトします。
データフラッシュ読み出しプロテクト解除処理関数	DF_Read_Protect_Cancel	データフラッシュの読み出しプロテクトを解除します。
データフラッシュ書き込み／消去プロテクト処理関数	DF_WriteErase_Protect	データフラッシュの書き込み／消去をプロテクトします。
データフラッシュ書き込み／消去プロテクト解除処理関数	DF_WriteErase_Protect_Cancel	データフラッシュの書き込み／消去プロテクトを解除します。

表 1.5 関数機能表で使用されている型定義一覧

型定義	型
_u1	unsigned char
_s2	signed short

1.10.1 ドライバ初期化処理関数

データフラッシュドライバ初期化処理関数の機能表を表 1.6 に、概略フローを図 1.31 に示します。

表 1.6 データフラッシュドライバ初期化処理関数機能表

関数名			
データフラッシュドライバ初期化処理関数 _s2 DF_Drv_Init(void)			
引数			
-			
戻り値			
_s2	DF_ST_OK	0 ：初期化正常終了	
	DF_ST_OK_REQUEST_ERASE	-1 ：初期化正常終了	（ブロック消去処理関数のコールを要求）
	DF_ST_ERR_REQUEST_FORMAT	-7 ：未フォーマット or データ紛失	（フォーマット処理関数のコールを要求）
	DF_ST_ERR_DEVICE	-8 ：デバイスエラー	（読み出しのみ可）
	DF_ST_ERR_INIT	-12：デバイス初期化エラー	（データフラッシュアクセス不可）
	DF_ST_ERR_PARAMETER_INVALID	-14：パラメータエラー	（DF_user.c 内の設定値が有効範囲外）
機能			
データフラッシュドライバの初期化を行います。データフラッシュドライバを使用する場合、システム起動時に必ずコールしてください。			
使用上の注意事項			

- データフラッシュドライバを使用する場合、システム起動時に必ずコールしてください。
- 関数リターン時、初期化処理が完了します。関数戻り値により、初期化成否を判断してください。

- 本関数の戻り値が "DF_ST_OK_REQUEST_ERASE" の場合、ドライバは、本関数処理内で "イレーズすべきブロック" を発見したことを示します。ブロック消去処理関数をコールし、イレーズを行ってください。イレーズを行わずにデータ更新処理関数をコールした場合、ブロックグループ内にデータ更新するための書き込み可能領域があればデータ更新が行われますが、書き込み可能領域がない場合、データ更新は行われずに、戻り値 "DF_ST_WARNING_NO_BLANK_AREA" をリターンします。
- 本関数の戻り値が "DF_ST_ERR_REQUEST_FORMAT" の場合、ドライバは、データへのアクセスが不可能な状態です。このような状態となった場合、データフラッシュフォーマット処理関数をコールしてデータ領域を再構築してください。この場合、すべてのデータは初期データ（不定値）になります。
- 本関数の戻り値が "DF_ST_ERR_DEVICE" の場合、ドライバは、本関数処理内でプログラムエラーを検出したため、有効データ読み出し処理関数によるデータ読み出しのみが可能な状態です。データ更新処理関数によるデータ更新やブロック消去処理関数によるイレーズはできません。
- 本関数の戻り値が " DF_ST_ERR_INIT" の場合、ドライバは、FCU ファームベリファイエラーもしくはデータフラッシュのブランクチェック時のエラー検出によりデータ領域のアクセスに必要な情報を取得できなかったため、データフラッシュ上の有効データの読み出し、データの更新、ブロック消去ができない状態です。データフラッシュエラー処理関数コールによるエラー処理後、再度本関数をコールしてください。（デバイスの状態やエラーの状況により正常復帰しない場合もあります。）
- 本関数の戻り値が "DF_ST_ERR_PARAMETER_INVALID" の場合、DF_user.h, DF_user.c 内の設定値が規定値の範囲外のため、初期化できない状態です。DF_user.h, DF_user.c 内の設定値を規定範囲内とし、コンパイルし直してください。

- 本関数実行中に割り込みによりユーザアプリケーションからコールできるドライバ関数は以下のとおりです。その他のドライバ関数をコールした場合、以後のドライバ動作が正常に行われない可能性があります。
  - データフラッシュ読み出しプロテクト解除関数
  - データフラッシュ書き込み／消去プロテクト解除関数
  - データサイズ取得関数

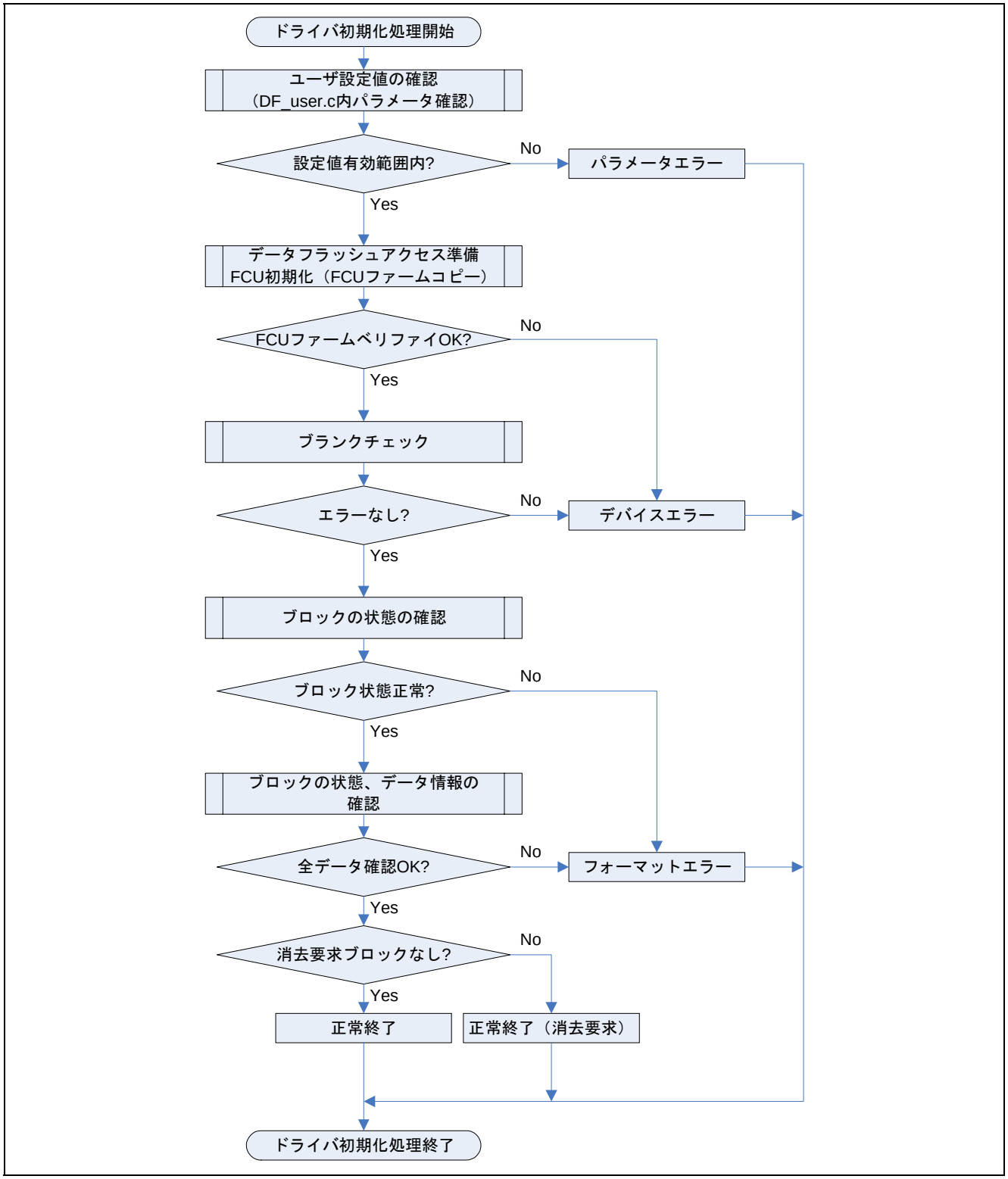

図 1.31　データフラッシュドライバ初期化処理関数概略フロー

1.10.2 フォーマット処理関数

データフラッシュフォーマット処理関数の機能表を表 1.7 に、概略フローを図 1.32、図 1.33 に示します。

表 1.7 データフラッシュフォーマット処理関数機能表

関数名			
データフラッシュフォーマット処理関数 _s2 DF_Format(_u1 code)			
引数			
_u1	Format_start	170：フォーマット実行許可コード	
戻り値			
_s2	DF_ST_OK	0 ：フォーマット正常終了	
	DF_ST_WARNING_WRITE_ERASE_PROTECT	-5 ：書き込み禁止状態	（プロテクト解除要求）
	DF_ST_WARNING_INVALID_ARGUMENT	-6 ：実行許可コードエラー	（フォーマット実行許可コードが不正）
	DF_ST_ERR_FORMAT	-13 ：フォーマット失敗	（データフラッシュアクセス不可）
機能			
データフラッシュ上に、本ドライバにより制御するためのデータ構造を構築します。			
使用上の注意事項			
● 本関数は、データフラッシュドライバ初期化処理関数によるドライバの初期化が行われた状態でコールしてください。ただし、データフラッシュドライバ初期化処理関数の戻り値が "DF_ST_ERR_DEVICE"、"DF_ST_ERR_INIT"、"DF_ST_ERR_PARAMETER_INVALID" であった場合、本関数はコールしないでください。 ● 関数リターン時、フォーマット処理が完了します。関数戻り値により、フォーマット成否を判断してください。 ● 本関数実行後は、実行前のデータフラッシュの状態にかかわらず新たにデータ領域を構築し、データ領域は不定値となります。したがって、本関数実行後、データ更新前のデータ（ID）の読み出し値も不定値となります。 ● 本関数の戻り値が "DF_ST_WARNING_WRITE_ERASE_PROTECT" の場合、データフラッシュが書き込み／消去プロテクトされており、フォーマット処理が実行できなかったことを示します。データフラッシュ書き込み／消去プロテクト解除関数をコールしプロテクト解除してください。 ● 本関数の戻り値が "DF_ST_WARNING_INVALID_ARGUMENT" の場合、引数（フォーマット実行許可コード：170）が不一致であったため、フォーマット処理を実行しなかったことを示します。 ● 本関数の戻り値が "DF_ST_ERR_FORMAT" の場合、フォーマット（データ構造の構築）に失敗し、ドライバによる有効データの読み出し、データの更新、ブロック消去処理ができない状態を示します。 ● 本関数実行中に割り込みによりユーザアプリケーションからコールできるドライバ関数は以下のとおりです。その他のドライバ関数をコールした場合、以後のドライバ動作が正常に行われない可能性があります。 — データフラッシュ読み出しプロテクト解除関数 — データフラッシュ書き込み／消去プロテクト解除関数 — データサイズ取得関数			

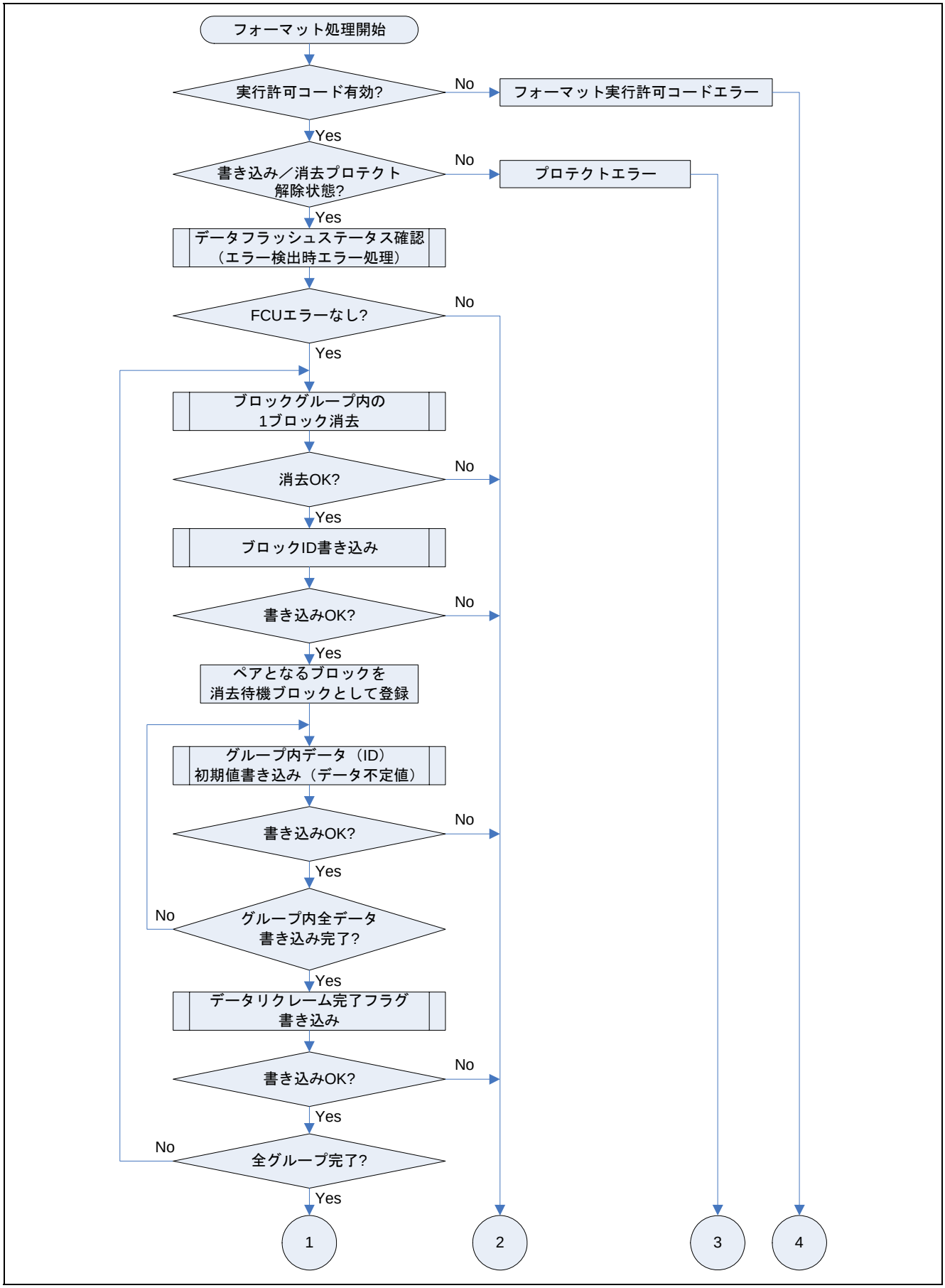

図 1.32　データフラッシュフォーマット処理関数概略フロー1

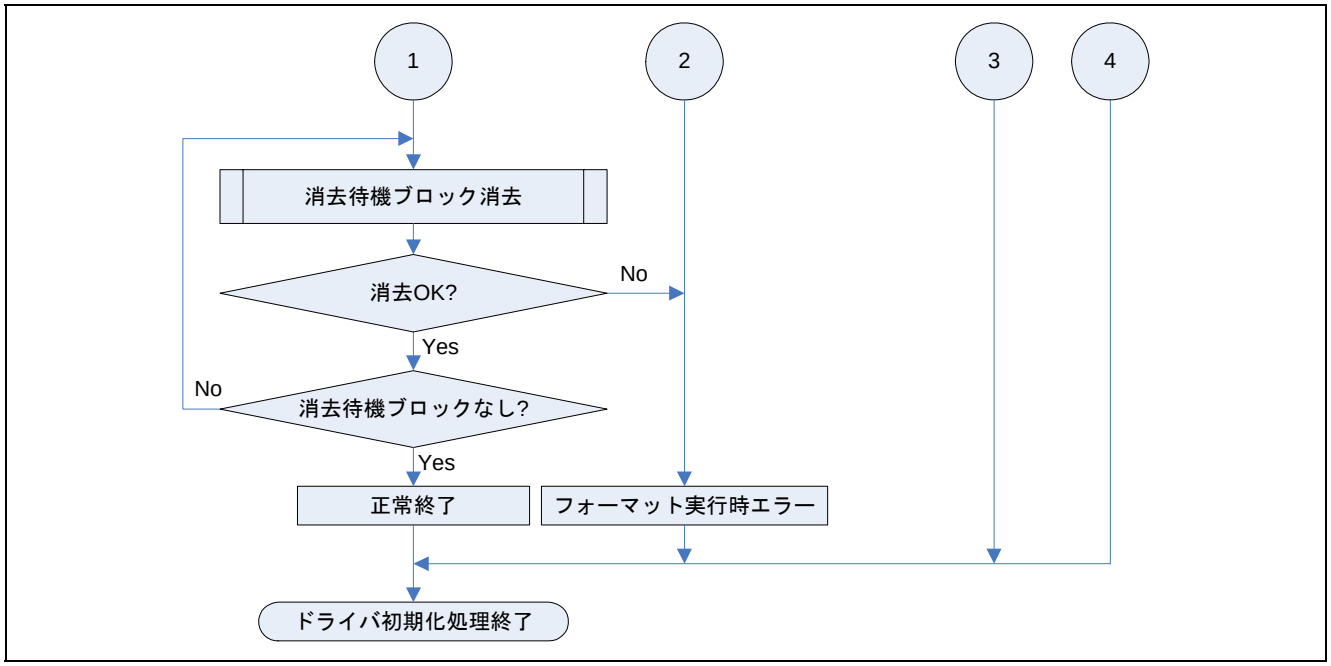

図 1.33　データフラッシュフォーマット処理関数概略フロー2

1.10.3 データの更新処理関数

データ更新処理関数の機能表を表 1.8 に、概略フローを図 1.34、図 1.35 に示します。

表 1.8 データ更新処理関数機能表

関数名				
データ更新処理関数 _s2 DF_Write_Data(_u1 data_id, _u1 *user_data_address)				
引数				
_u1	data_id		：更新するデータ番号	
_u1	*user_data_address		：更新データの格納元ポインタ	
戻り値				
_s2	DF_ST_OK	0	：データ更新正常終了	
	DF_ST_OK_REQUEST_ERASE	-1	：データ更新正常終了	（ブロック消去処理関数のコールを要求）
	DF_ST_WARNING_DRIVER_BUSY	-2	：ドライバビジー	（待ち要求）
	DF_ST_WARNING_NO_BLANK_AREA	-3	：書き込み可能領域なし	（ブロック消去完了待ち） （ブロック消去処理関数のコールを要求）
	DF_ST_WARNING_READ_PROTECT	-4	：読み出し禁止状態	（プロテクト解除要求）
	DF_ST_WARNING_WRITE_ERASE_PROTECT	-5	：書き込み禁止状態	（プロテクト解除要求）
	DF_ST_WARNING_INVALID_ARGUMENT	-6	：データ番号エラー	（データ番号が定義範囲外）
	DF_ST_ERR_DEVICE	-8	：異常検出	（読み出しのみ可）
	DF_ST_ERR_DEVICE_TIMEOUT	-9	：異常検出	（読み出しのみ可）
	DF_ST_ERR_FCU	-10	：異常検出	（読み出しのみ可）
	DF_ST_ERR_LOST_ID	-11	：異常検出	（データ紛失）
機能				
引数で指定されたデータ番号のデータを更新します。データリクレーム処理が必要な場合、データリクレーム処理を行います。				
使用上の注意事項				

- 本関数は、データフラッシュドライバ初期化処理関数によるドライバの初期化が正常に行われた状態でコールしてください。
- 関数リターン時、データ更新処理が完了します。関数戻り値により、データ更新成否を判断してください。
- 本関数の戻り値が "DF_ST_OK_REQUEST_ERASE" の場合、ドライバは、データの更新処理を正常に完了したうえで、本関数処理内において "イレーズすべきブロック" を発見したことを示します。ブロック消去処理関数をコールし、イレーズを行ってください。イレーズを行わずにデータ更新処理関数を再度コールした場合、ブロックグループ内にデータ更新するための書き込み可能領域があればデータ更新が行われますが、書き込み可能領域がない場合、データ更新は行われずに、戻り値 "DF_ST_WARNING_NO_BLANK_AREA" をリターンします。
- 本関数の戻り値が "DF_ST_WARNING_DRIVER_BUSY" の場合、ドライバは、以下の状態であるためデータの更新ができないことを示します。
  - 本関数を多重にコール
  - 有効データ読み出し処理関数を実行中
  - データフラッシュドライバ初期化処理関数、データフラッシュフォーマット関数実行中
  - 本関数コール時にすでにエラー状態
- 本関数の戻り値が "DF_ST_WARNING_NO_BLANK_AREA" の場合、ブロックグループ内にデータ更新するための書き込み可能領域がないため、データの更新ができないことを示します。ブロック消去処理完了後にデータの更新を行ってください。
- 本関数の戻り値が "DF_ST_WARNING_READ_PROTECT" の場合、データ更新処理正常終了後のデータリクレーム処理内でデータフラッシュが読み出しプロテクトされており、リクレーム処理ができないことを示します。データフラッシュ読み出しプロテクト解除関数をコールしプロテクト解除してください。（リクレーム処理は次回以降のデータ更新時に行われます。）
- 本関数の戻り値が "DF_ST_WARNING_WRITE_ERASE_PROTECT" の場合、データフラッシュが書き込み／消去プロテクトされていることを示します。データフラッシュ書き込み／消去プロテクト解除関数をコールしプロテクト解除してください。
- 本関数の戻り値が "DF_ST_WARNING_INVALID_ARGUMENT" の場合、引数で指定したデータ番号が定義範囲外であることを示します。データ番号の有効範囲は DF_user.h 内で定義されたデータ数（DF_DATA_ID_NUM）に従い、0～（DF_DATA_ID_NUM-1）となります。
- 本関数の戻り値が "DF_ST_ERR_DEVICE"、"DF_ST_ERR_DEVICE_TIMEOUT"、"DF_ST_ERR_FCU" の場合、本関数処理内でエラーを検出し、データの更新が正常に行われなかったことを示します。データフラッシュエラー処理関数コールによるエラー処理を行ってください。（デバイスの状態やエラーの状況により正常復帰しない場合もあります。）
- 本関数の戻り値が "DF_ST_ERR_LOST_ID" の場合、データ更新処理正常終了後のデータリクレーム処理内でリクレームデータが同ブロックグループ内に見つからずデータ紛失したことを示します。
- 本関数実行中に割り込みによりユーザアプリケーションからコールできるドライバ関数は以下のとおりです。その他のドライバ関数をコールした場合、以後のドライバ動作が正常に行われない可能性があります。
  - 有効データ読み出し処理関数
  - データフラッシュ読み出しプロテクト解除関数
  - データフラッシュ書き込み／消去プロテクト解除関数
  - データサイズ取得関数
  - 書き込み可能領域数取得関数（ただし、データの更新処理状態により正しい値が返されない場合があります。）

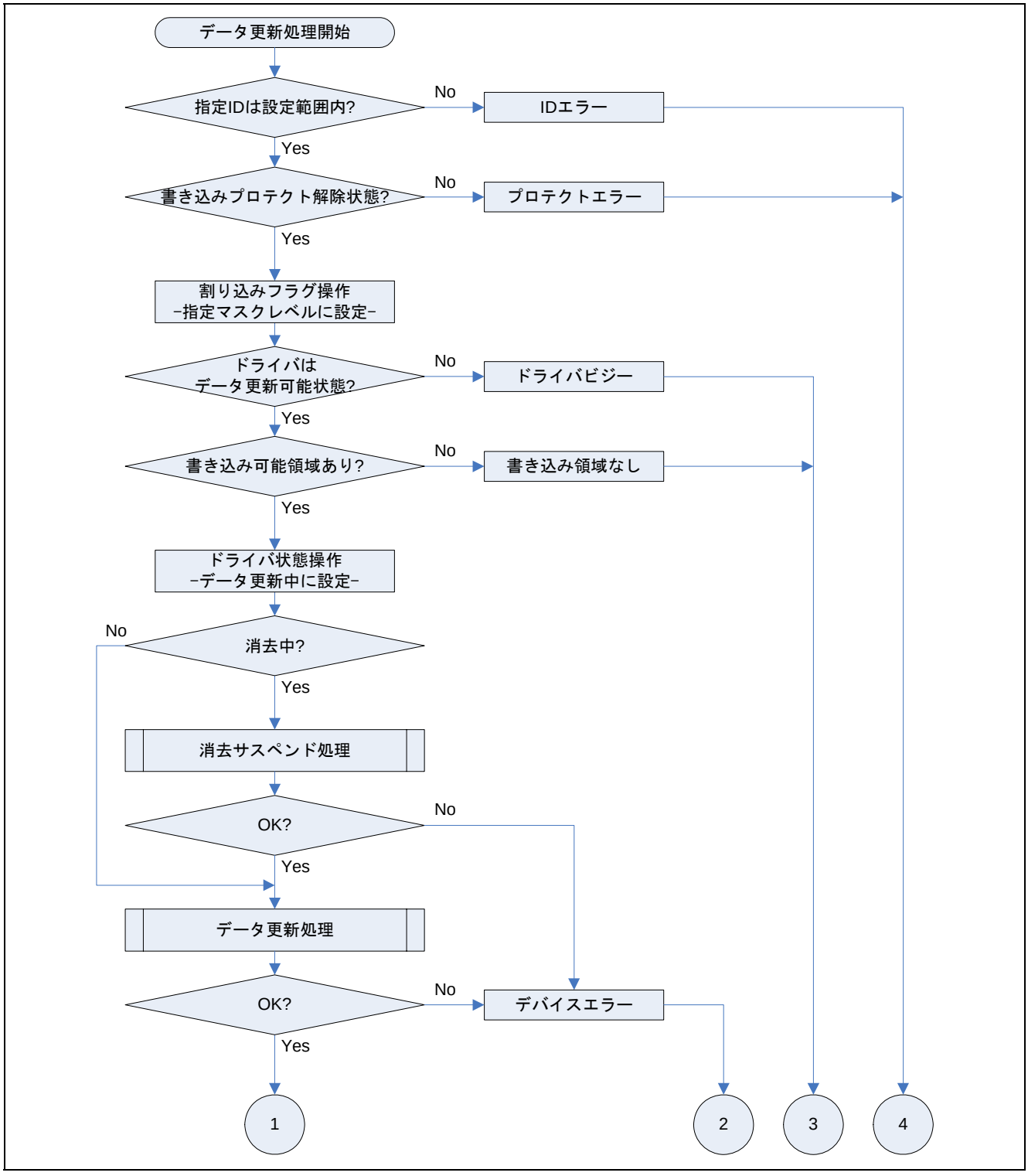

図 1.34　データ更新処理関数概略フロー1

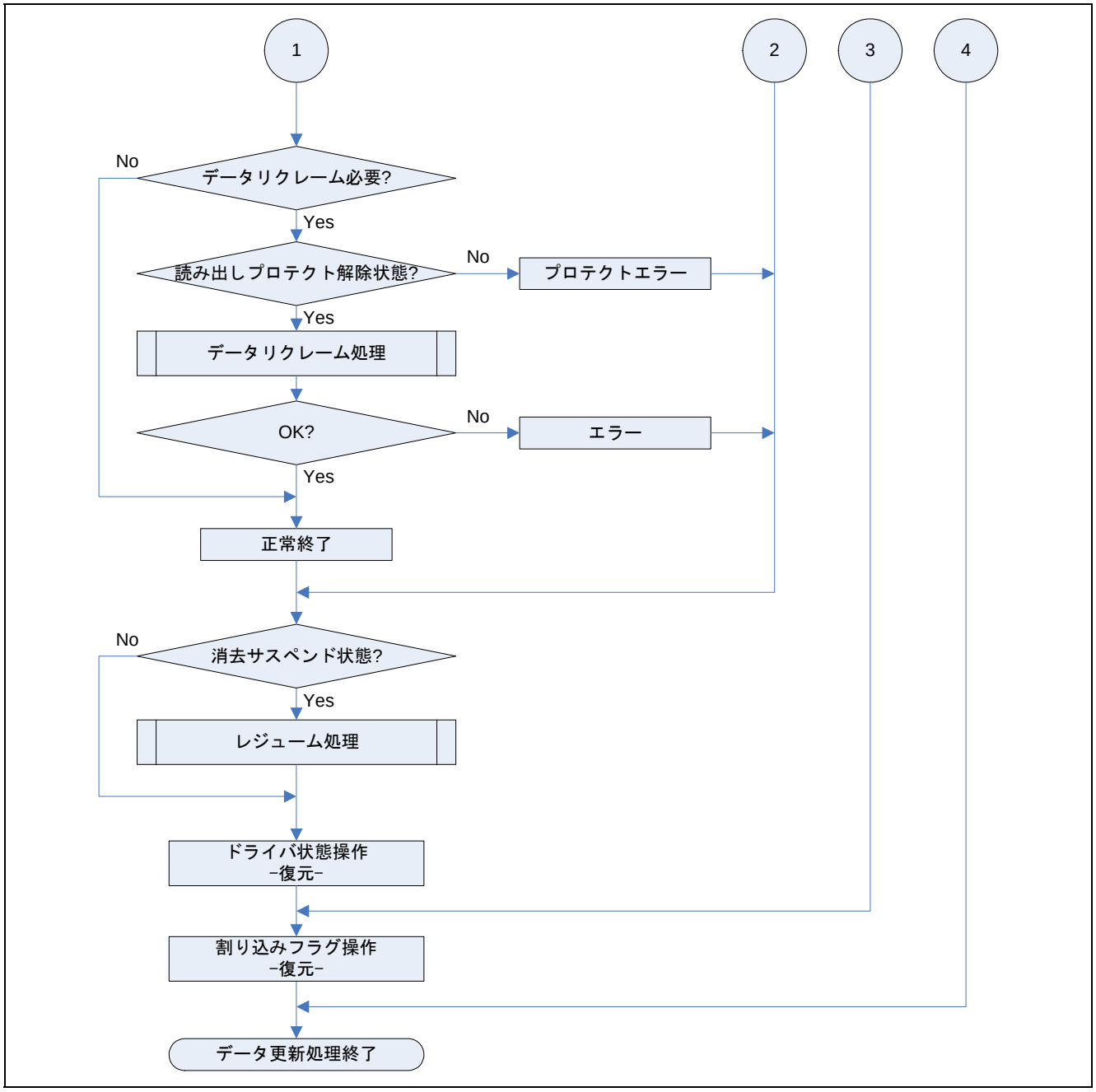

図 1.35　データ更新処理関数概略フロー2

1.10.4 有効データの読み出し処理関数

有効データ読み出し処理関数の機能表を表 1.9 に、概略フローを図 1.36 に示します。

表 1.9 有効データ読み出し処理関数機能表

関数名				
有効データ読み出し処理関数 _s2 DF_Read_Data(_u1 data_id, _u1 *user_storage_address)				
引数				
_u1	data_id		：読み出すデータ番号	
_u1	*user_storage_address		：読み出したデータの格納先ポインタ	
戻り値				
_s2	DF_ST_OK	0	：読み出し正常終了	
	DF_ST_OK_REQUEST_ERASE	-1	：読み出し正常終了	（ブロック消去処理関数のコールを要求）
	DF_ST_WARNING_DRIVER_BUSY	-2	：ドライバビジー	（待ち要求）
	DF_ST_WARNING_READ_PROTECT	-4	：読み出し禁止状態	（プロテクト解除要求）
	DF_ST_WARNING_INVALID_ARGUMENT	-6	：データ番号エラー	（データ番号が定義範囲外）
	DF_ST_ERR_DEVICE	-8	：異常検出	
	DF_ST_ERR_DEVICE_TIMEOUT	-9	：異常検出	
	DF_ST_ERR_FCU	-10	：異常検出	
	DF_ST_ERR_LOST_ID	-11	：異常検出	（データ紛失）
機能				
引数で指定されたデータ番号のデータフラッシュ上に格納されている有効データを読み出し、引数で指定された格納ポインタに格納します。				
使用上の注意事項				

- 本関数は、データフラッシュドライバ初期化処理関数によるドライバの初期化が正常に行われた状態でコールしてください。
- 関数リターン時、読み出し処理が完了します。関数戻り値により、読み出し成否を判断してください。

- 読み出すデータ番号のデータが更新中（データ更新処理完了前）であった場合、読み出すデータは更新前の有効データとなります。

- 本関数の戻り値が "DF_ST_OK_REQUEST_ERASE" の場合、ドライバは、データの読み出し処理を正常に完了したうえで、本関数処理内で "イレーズすべきブロック" を発見したことを示します。
- 本関数の戻り値が "DF_ST_WARNING_DRIVER_BUSY" の場合、ドライバは、以下の状態であるためデータの更新ができないことを示します。
  - 本関数を多重にコール
  - ブロック消去サスペンド状態でデータ更新処理関数実行中
  - データフラッシュドライバ初期化処理関数、データフラッシュフォーマット関数実行中
  - ドライバ初期化エラーまたはデータフラッシュフォーマットエラー
- 本関数の戻り値が "DF_ST_WARNING_READ_PROTECT" の場合、データフラッシュが読み出しプロテクトされていることを示します。データフラッシュ読み出しプロテクト解除関数をコールしプロテクト解除してください。
- 本関数の戻り値が "DF_ST_WARNING_INVALID_ARGUMENT" の場合、引数で指定したデータ番号が定義範囲外であることを示します。データ番号の有効範囲は DF_user.h 内で定義されたデータ数（DF_DATA_ID_NUM）に従い、0～（DF_DATA_ID_NUM -1）となります。
- 本関数の戻り値が "DF_ST_ERR_DEVICE"、"DF_ST_ERR_DEVICE_TIMEOUT"、"DF_ST_ERR_FCU" の場合、消去サスペンド処理もしくは書き込みサスペンド処理中にエラーを検出したことを示しますが、有効データの読み出し処理は実行します。
- 本関数の戻り値が "DF_ST_ERR_LOST_ID" の場合、引数で指定されたデータ番号の有効データがデータフラッシュ内に見つからずデータ紛失したことを示します。

- 本関数実行中に割り込みによりユーザアプリケーションからコールできるドライバ関数は以下のとおりです。その他のドライバ関数をコールした場合、以後のドライバ動作が正常に行われない可能性があります。
  - データフラッシュ読み出しプロテクト解除関数
  - データフラッシュ書き込み／消去プロテクト関数（ただし、その他ドライバ関数実行中で割り込みにより本関数を実行している場合は、実行中のドライバ関数により制限されている場合があります。）
  - データフラッシュ書き込み／消去プロテクト解除関数
  - ユーザデータサイズ取得関数
  - 書き込み可能領域数取得関数（ただし、データ更新中で書き込みサスペンド状態となった場合、正しい値が返されない場合があります。）

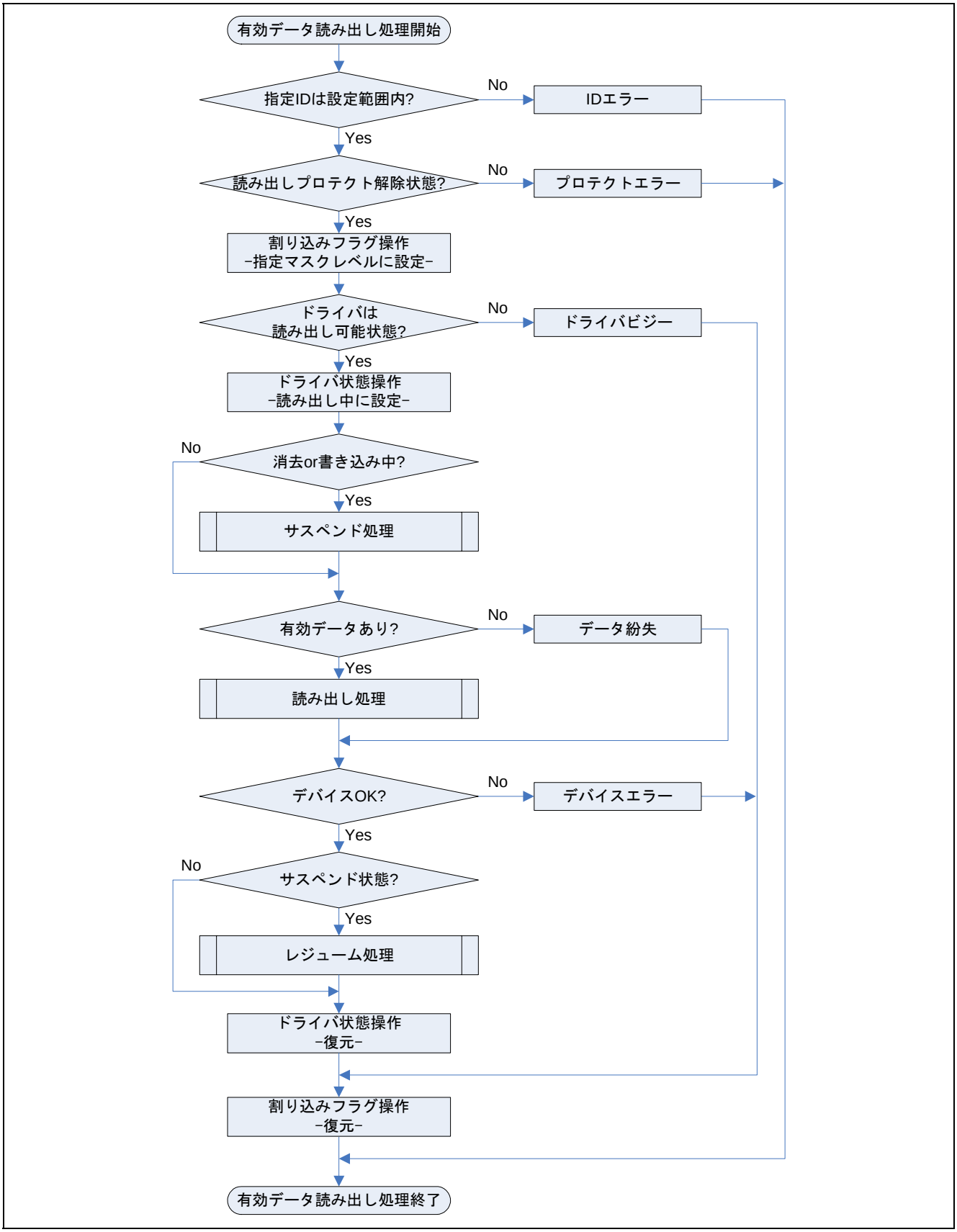

図 1.36　有効データ読み出し処理関数概略フロー

1.10.5 ブロック消去処理関数

ブロック消去処理関数の機能表を表 1.10 に、概略フローを図 1.37、図 1.38 に示します。

表 1.10 ブロック消去処理関数機能表

関数名			
ブロック消去処理関数 _s2 DF_Erase_block(void)			
引数			
–			
戻り値			
_s2	DF_ST_OK_ERASE_BLOCK_NONE	1 ：消去対象ブロックなし	
	DF_ST_OK	0 ：消去正常終了	
	DF_ST_OK_REQUEST_ERASE	-1 ：消去正常終了	（ブロック消去処理関数のコールを再要求）
	DF_ST_WARNING_DRIVER_BUSY	-2 ：ドライバビジー	（待ち要求）
	DF_ST_WARNING_WRITE_ERASE_PROTECT	-5 ：消去禁止状態	（プロテクト解除要求）
	DF_ST_ERR_DEVICE	-8 ：異常検出	（データフラッシュアクセス不可）
	DF_ST_ERR_DEVICE_TIMEOUT	-9 ：異常検出	（DF_user.c 内の設定値が指定範囲外）
機能			
消去対象ブロックとして登録されているブロックの消去を行います。			
使用上の注意事項			

- 本関数は、データフラッシュドライバ初期化処理関数によるドライバの初期化が正常に行われた状態でコールしてください。
- 関数リターン時、消去対象ブロックとして登録されているブロックの内 1 ブロックのみの消去処理が完了します。関数戻り値により、消去成否を判断してください。

- 本関数の戻り値が "DF_ST_OK_ERASE_BLOCK_NONE" の場合、消去対象ブロックとして登録されているブロックがないことを示します。
- 本関数の戻り値が "DF_ST_OK_REQUEST_ERASE" の場合、ドライバは、本関数処理後 "イレーズすべきブロック" が残っていることを示します。本関数を再度コールし、イレーズを行ってください。イレーズを行わずにデータ更新処理関数をコールした場合、ブロックグループ内にデータ更新するための書き込み可能領域があればデータ更新が行われますが、書き込み可能領域がない場合、データ更新は行われずに、エラーを示す戻り値 "DF_ST_WARNING_NO_BLANK_AREA" をリターンします。
- 本関数の戻り値が "DF_ST_WARNING_DRIVER_BUSY" の場合、ドライバは、以下の状態であるため本関数によるブロック消去処理ができないことを示します。
  - 本関数を多重にコール
  - 有効データ読み出し処理関数を実行中
  - データ更新処理関数を実行中
  - データフラッシュドライバ初期化処理関数、データフラッシュフォーマット関数実行中
  - 本関数コール時にすでにエラー状態
- 本関数の戻り値が "DF_ST_WARNING_WRITE_ERASE_PROTECT" の場合、データフラッシュが書き込み／消去プロテクトされていることを示します。データフラッシュ書き込み／消去プロテクト解除関数をコールしプロテクト解除してください。
- 本関数の戻り値が "DF_ST_ERR_REQUEST_FORMAT" の場合、ドライバは、データへのアクセスが不可能な状態です。このような状態となった場合、データフラッシュフォーマット処理関数を使用してデータ領域を再構築してください。この場合、すべてのデータは初期データ（不定値）になります。
- 本関数の戻り値が "DF_ST_ERR_DEVICE"、"DF_ST_ERR_DEVICE_TIMEOUT" の場合、消去処理時、もしくは消去処理に関連するブロック管理情報の書き込み時にエラーを検出したことを示し、ブロック消去処理が正常に行われなかったことを示します。データフラッシュエラー処理関数コールによるエラー処理を行ってください。（デバイスの状態やエラーの状況により正常復帰しない場合もあります。）

- 本関数実行中に割り込みによりユーザアプリケーションからコールできるドライバ関数は以下のとおりです。その他のドライバ関数をコールした場合、以後のドライバ動作が正常に行われない可能性があります。
  - 有効データ読み出し処理関数
  - データ更新処理関数
  - データフラッシュ読み出しプロテクト関数
  - データフラッシュ読み出しプロテクト解除関数
  - データフラッシュ書き込み／消去プロテクト解除関数
  - データサイズ取得関数
  - 書き込み可能領域数取得関数（ただし、ブロック消去処理状態により正しい値が返されない場合があります。）

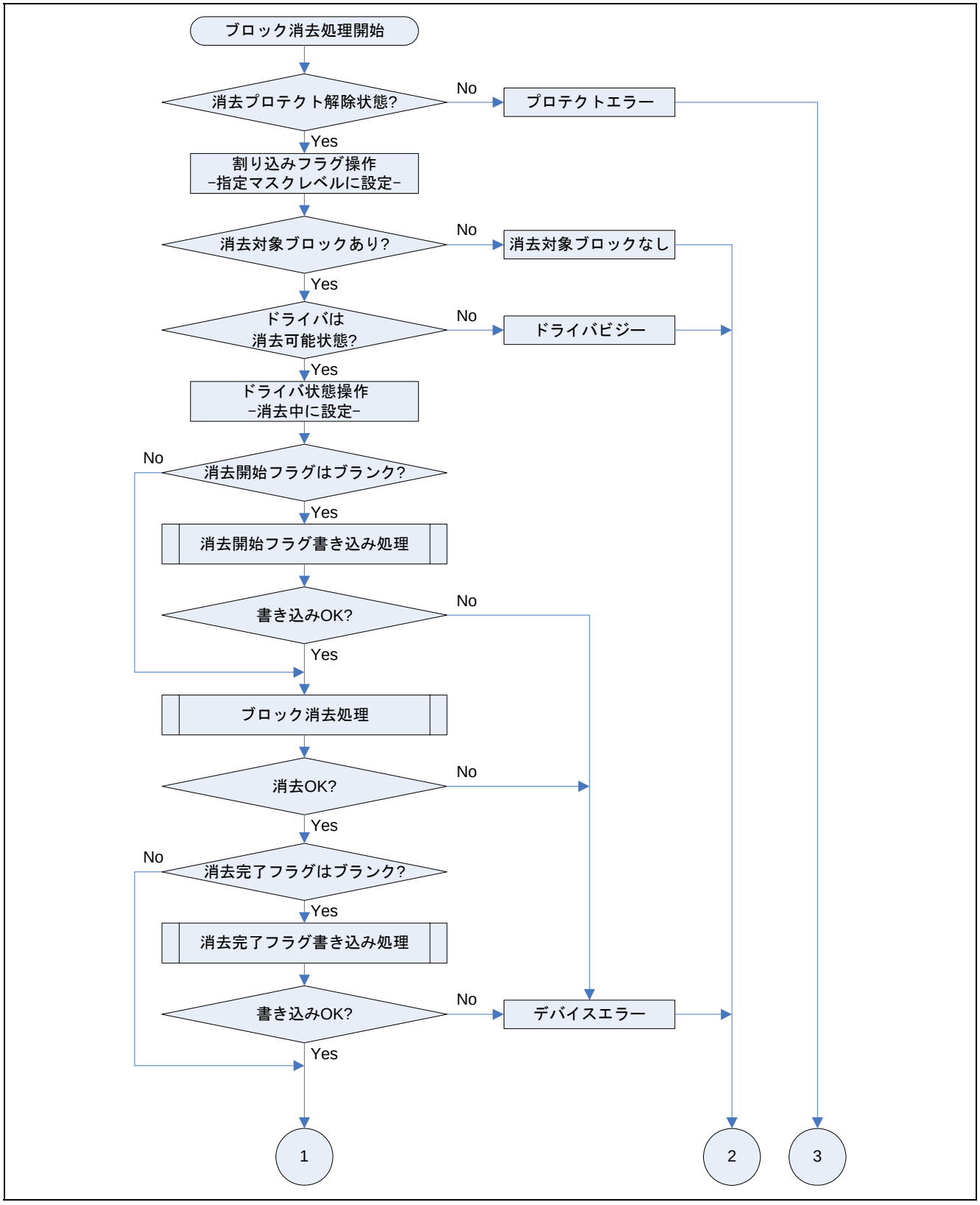

図 1.38　ブロック消去処理関数概略フロー1

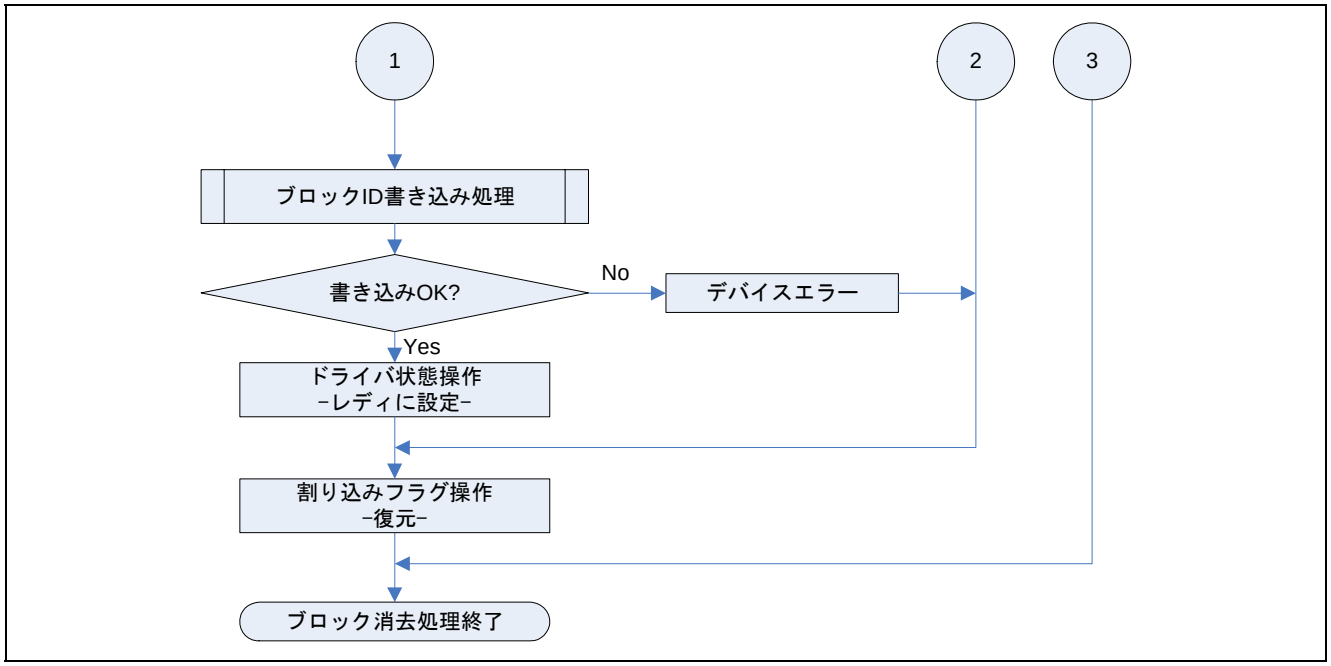

図 1.38　ブロック消去処理関数概略フロー2

1.10.6　エラー処理関数

エラー処理関数の機能表を表 1.11 に、概略フローを図 1.39 に示します。

表 1.11　エラー処理関数機能表

関数名		
エラー処理関数 _s2　DF_Error_Management (void)		
引数		
-		
戻り値		
_s2	DF_ST_OK	0　：エラー処理正常終了
	DF_ST_OK_REQUEST_ERASE	-1　：エラー処理正常終了　（ブロック消去処理関数のコールを要求）
	DF_ST_ERR_DEVICE	-8　：デバイスエラー
	DF_ST_ERR_DEVICE_TIMEOUT	-9　：デバイスエラー
	DF_ST_ERR_FCU	-10：デバイスエラー
機能		
デバイス（データフラッシュ）やデータフラッシュのシーケンサ（FCU）のエラー状態に応じた処理を行います。 また、書き込み／消去サスペンド状態で本関数をコールした場合、FCU をリセットし、サスペンド状態を解除します。		
使用上の注意事項		
● ドライバ初期化処理が完了していない場合は、ドライバ初期化処理関数をコールしてください。 ● 関数リターン時、エラー処理が完了します。関数戻り値により、エラー処理成否を判断してください。（デバイスの状態やエラーの状況により正常復帰しない場合もあります。） ● 本関数の戻り値が "DF_ST_OK_REQUEST_ERASE" の場合、ドライバは、エラー処理を正常終了したうえで、本関数処理内で "イレーズすべきブロック" を発見したことを示します。ブロック消去処理関数をコールし、イレーズを行ってください。 ● 本関数の戻り値が "DF_ST_ERR_DEVICE"、"DF_ST_ERR_DEVICE_TIMEOUT"、"DF_ST_ERR_FCU" の場合、エラー処理後、正常復帰しなかったことを示します。		

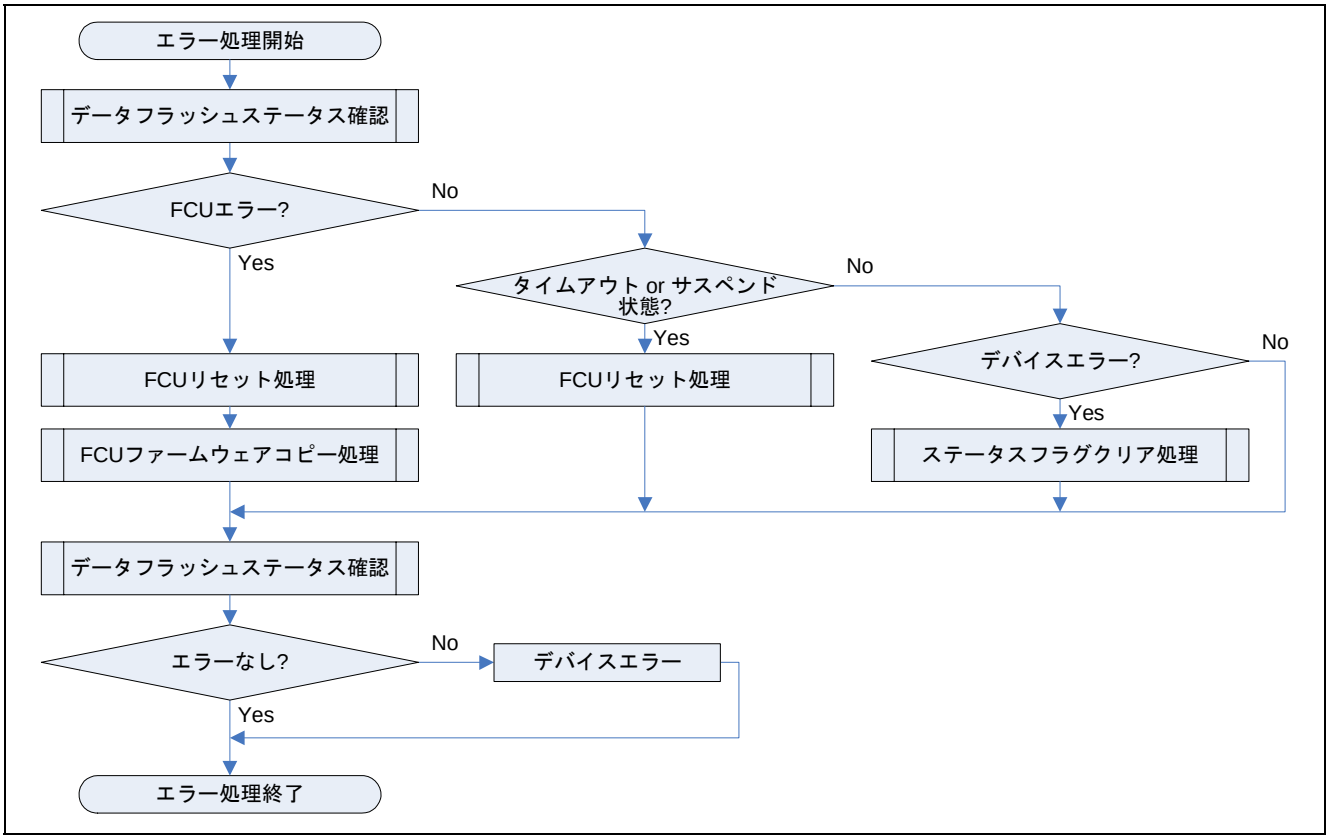

図 1.39　エラー処理関数概略フロー

1.10.7 データサイズ取得関数

データサイズ取得関数の機能表を表 1.12 に、概略フローを図 1.40 に示します。

表 1.12 データサイズ取得関数機能表

関数名			
データサイズ取得関数 _s2 DF_Data_Size(_u1 data_id)			
引数			
_u1	data_id		：サイズを取得するデータ番号
戻り値			
_s2		0～256	：データサイズ
	DF_ST_WARNING_INVALID_ARGUMENT	-6	：データ番号エラー （データ番号が定義範囲外）
機能			
引数で指定されたデータ番号のデータサイズを戻り値として返します。			
使用上の注意事項			
● 本関数の戻り値が "DF_ST_WARNING_INVALID_ARGUMENT" の場合、引数で指定したデータ番号が定義範囲外であることを示します。データ番号の有効範囲は DF_user.h 内で定義されたデータ数（DF_DATA_ID_NUM）に従い、0～（DF_DATA_ID_NUM-1）となります。			

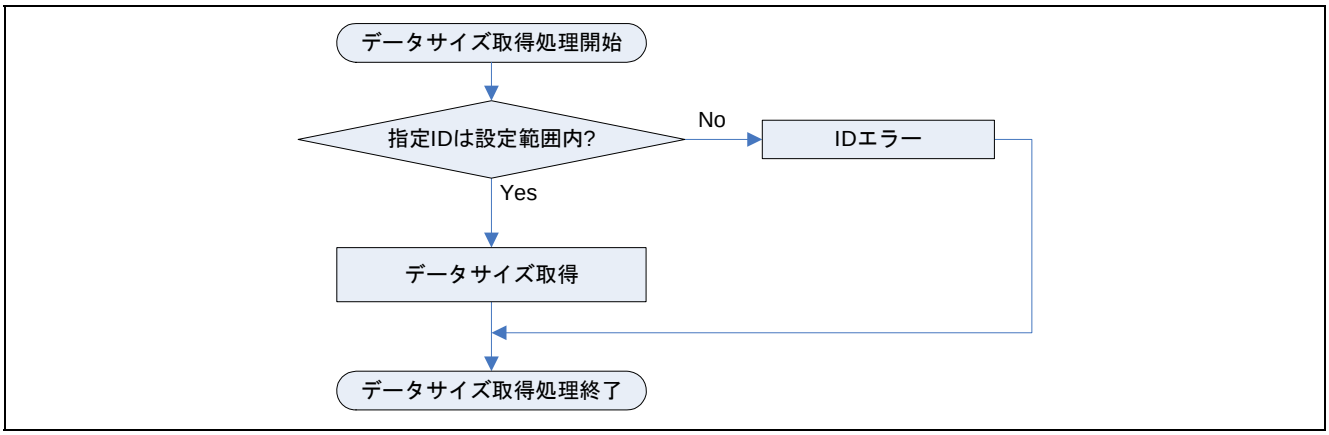

図 1.40 データサイズ取得関数概略フロー

1.10.8 書き込み可能領域数取得関数

書き込み可能領域数取得関数の機能表を表 1.13 に、概略フローを図 1.41 に示します。

表 1.13 書き込み可能領域数取得関数機能表

関数名	
書き込み可能領域数取得関数 _s2 DF_Blank_Number(_u1 grp)	
引数	
_u1 grp	：書き込み可能領域数を取得するブロックグループ番号
戻り値	
_s2	0～ ：書き込み可能領域数
DF_ST_WARNING_INVALID_ARGUMENT	-6 ：ブロックグループ番号エラー （ブロックグループ番号が定義範囲外）
機能	
引数で指定されたブロックグループの、データ更新に使用しているブロック内の書き込み可能領域数を戻り値として返します。	
使用上の注意事項	
● 本関数は、データフラッシュドライバ初期化処理関数によるドライバの初期化が正常に行われた状態でコールしてください。 ● 関数リターン時、書き込み可能領域数取得処理が完了します。関数戻り値により、書き込み可能領域数取得成否を判断してください。 ● 他のドライバ関数実行中に、割り込みにより本ドライバ関数をコールした場合、または、本関数実行中に割り込みにより他のドライバ関数をコールした場合には、ドライバの実行状態により正しい値が返されない場合があります。（書き込み可能領域数の増減にかかわるドライバ関数実行の場合、増減前の値が返される場合があります。） ● 本関数の戻り値が "DF_ST_WARNING_INVALID_ARGUMENT" の場合、引数で指定したブロックグループ番号が定義範囲外であることを示します。ブロックグループ番号の有効範囲は DF_user.h 内で定義されたブロックグループ数（DF_GROUP_NUM）に従い、0～（DF_GROUP_NUM-1）となります。 ● 本関数実行中に割り込みによりユーザアプリケーションからコールできるドライバ関数は以下のとおりです。その他のドライバ関数をコールした場合、以後のドライバ動作が正常に行われない可能性があります。 — 有効データ読み出し処理関数 — データ更新処理関数 — ブロック消去処理関数 — データフラッシュ読み出しプロテクト／プロテクト解除関数 — データフラッシュ書き込み／消去プロテクト／プロテクト解除関数 — データサイズ取得関数	

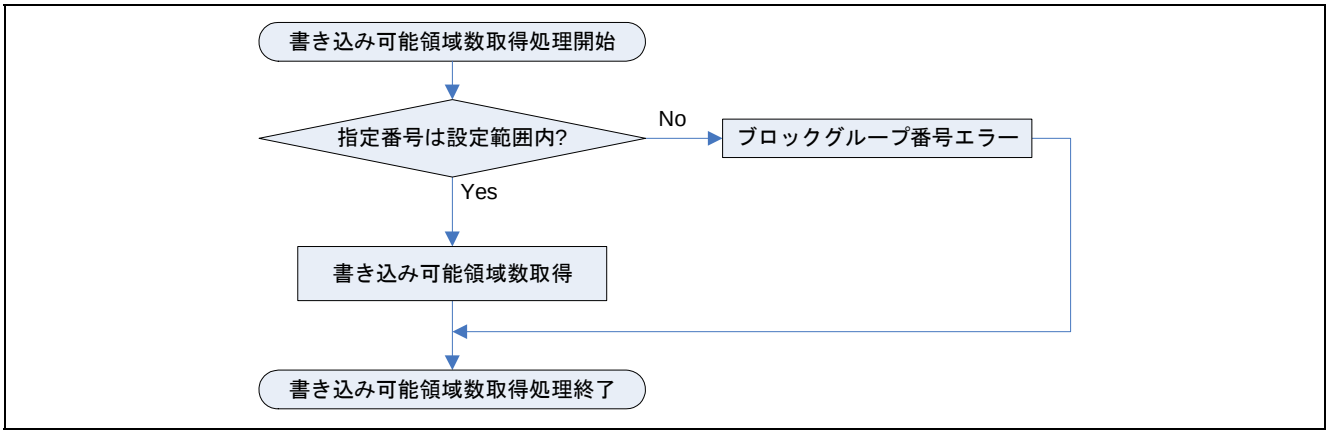

図 1.41 書き込み可能領域数取得関数概略フロー

1.10.9 データフラッシュ読み出しプロテクト処理関数

データフラッシュ読み出しプロテクト処理関数の機能表を表 1.14 に、概略フローを図 1.42 に示します。

表 1.14 データフラッシュ読み出しプロテクト処理関数機能表

関数名
データフラッシュ読み出しプロテクト処理関数 void DF_Read_Protect(void)
引数
-
戻り値
-
機能
データフラッシュの読み出しをプロテクトします。
使用上の注意事項
● 本関数実行後（読み出しプロテクト）の状態で以下に示すドライバ関数をコールした場合、コールされた各関数処理が正常に終了できない場合があります。 — 有効データ読み出し処理関数 — データ更新処理関数（データリクレーム処理を伴う場合） 上記ドライバ関数の戻り値が "DF_ST_WARNING_READ_PROTECT" であった場合、データフラッシュ読み出しプロテクト解除関数をコールし、データフラッシュの読み出しプロテクトを解除してください。 また、以下に示すドライバ関数実行中に割り込みにより本関数をコールした場合、実行中のドライバ関数処理が正常に終了できない場合があります。 — データフラッシュドライバ初期化処理関数 — 有効データ読み出し処理関数 — データ更新処理関数（データリクレーム処理を伴う場合）

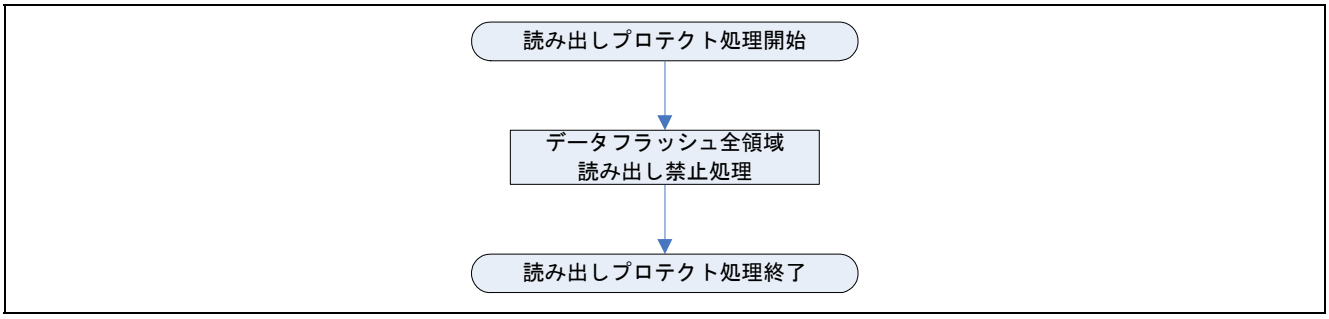

図 1.42 データフラッシュ読み出しプロテクト処理関数概略フロー

1.10.10　データフラッシュ読み出しプロテクト解除処理関数

データフラッシュ読み出しプロテクト解除処理関数の機能表を表 1.15 に、概略フローを図 1.43 に示します。

表 1.15　データフラッシュ読み出しプロテクト解除処理関数機能表

関数名
データフラッシュ読み出しプロテクト解除処理関数 void　DF_Read_Protect_Cancel(void)
引数
-
戻り値
-
機能
データフラッシュの読み出しプロテクトを解除します。
使用上の注意事項
-

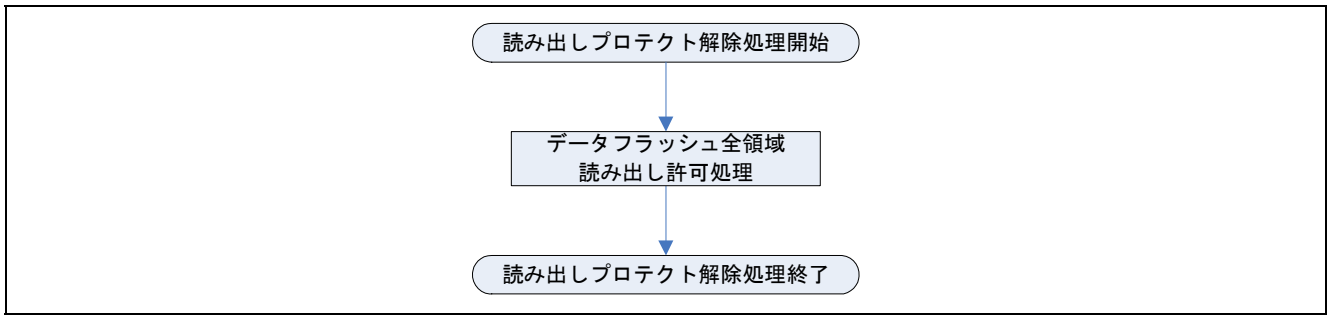

図 1.43　データフラッシュ読み出しプロテクト解除処理関数概略フロー

1.10.11　データフラッシュ書き込み／消去プロテクト処理関数

データフラッシュ書き込み／消去プロテクト処理関数の機能表を表 1.16 に、概略フローを図 1.44 に示します。

表 1.16　データフラッシュ書き込み／消去プロテクト処理関数機能表

関数名
データフラッシュ書き込み／消去プロテクト処理関数 void　DF_WriteErase_Protect(void)
引数
-
戻り値
-
機能
データフラッシュの書き込み／消去をプロテクトします。
使用上の注意事項
● 本関数実行後（書き込み／消去プロテクト）の状態で以下に示すドライバ関数をコールした場合、コールされた各関数処理が正常に終了できない場合があります。 — データフラッシュフォーマット処理関数 — データ更新処理関数 — ブロック消去処理関数 上記ドライバ関数の戻り値が "DF_ST_WARNING_WRITE_ERASE_PROTECT" であった場合、データフラッシュ書き込み／消去プロテクト解除関数をコールし、データフラッシュの書き込み／消去プロテクトを解除してください。 また、以下に示すドライバ関数実行中に割り込みにより本関数をコールした場合、実行中のドライバ関数処理が正常に終了できない場合があります。 — データフラッシュドライバ初期化処理関数 — データフラッシュフォーマット処理関数 — データ更新処理関数 — ブロック消去処理関数

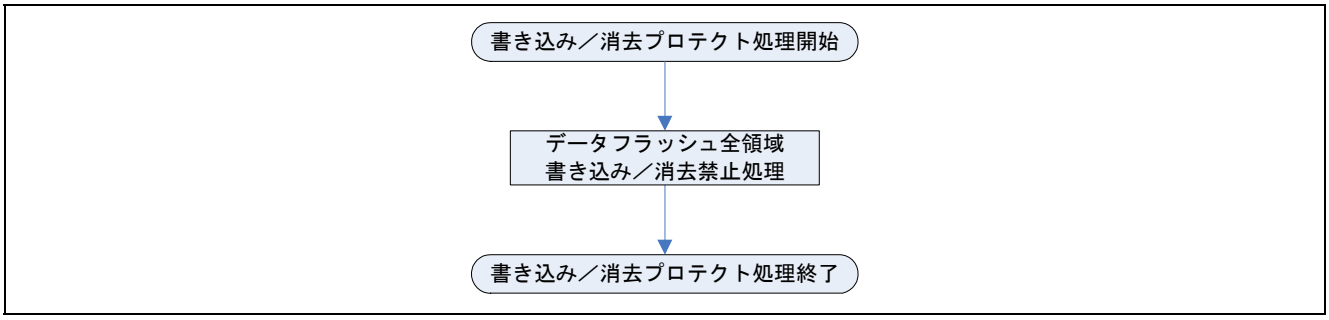

図 1.44　データフラッシュ書き込み／消去プロテクト処理関数概略フロー

1.10.12　データフラッシュ書き込み／消去プロテクト解除処理関数

データフラッシュ書き込み／消去プロテクト解除処理関数の機能表を表 1.17 に、概略フローを図 1.45 に示します。

表 1.17　データフラッシュ書き込み／消去プロテクト解除処理関数機能表

関数名
データフラッシュ書き込み／消去プロテクト解除処理関数 void　DF_WriteErase_Protect_Cancel(void)
引数
-
戻り値
-
機能
データフラッシュの書き込み／消去プロテクトを解除します
使用上の注意事項
-

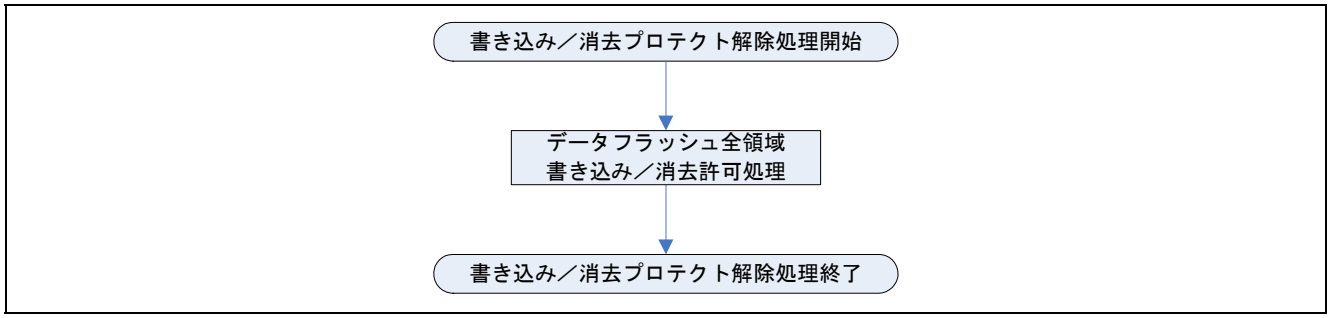

図 1.45　データフラッシュ書き込み／消去プロテクト解除処理関数概略フロー

1.11 ドライバの状態とコールできるドライバ関数

ドライバは、各ドライバ関数処理の実行に伴う状態の変化を特定の値（変数）として RAM に保存します。

1.11.1 ドライバの状態

ドライバの状態を示す RAM の値を表 1.18 に示します。

表中の各変数の値は、DF_drv.h 内で定義されています。

表 1.18 ドライバ状態-RAM 変数対応表

ドライバ状態				ドライバの状態を示す RAM の値		
				drv_status	drv_write_status	drv_erase_status
ドライバ初期化前				不定		
待機	正常状態			DF_DRV_ST_READY	DF_WRITE_ST_READY	DF_ERASE_ST_READY
	警告状態	未フォーマット		DF_DRV_ST_WARNING_REQ_FORMAT	DF_WRITE_ST_BUSY	DF_ERASE_ST_BUSY
	エラー状態	初期化実行時エラー	読み出しのみ可	DF_DRV_ST_READY	DF_WRITE_ST_ERROR	DF_ERASE_ST_ERROR
			アクセス不可	DF_DRV_ST_ERROR_INIT		
		フォーマット実行時エラー		DF_DRV_ST_ERROR_FORMAT		
		書き込みエラー		DF_DRV_ST_READY		
		消去エラー			DF_WRITE_ST_READY	
ビジー	ドライバ初期化処理関数実行中			DF_DRV_ST_INITIALIZATION	DF_WRITE_ST_BUSY	DF_ERASE_ST_BUSY
	データフラッシュフォーマット処理関数実行中			DF_DRV_ST_FORMAT		
	有効データ読み出し処理関数実行中			DF_DRV_ST_BUSY		
	データ更新処理関数実行中			-		
	ブロック消去処理関数実行中				DF_WRITE_ST_READY	
サスペンド	書き込みサスペンド中			-	DF_WRITE_ST_SUSPEND	DF_ERASE_ST_BUSY
	消去サスペンド中				DF_WRITE_ST_READY	DF_ERASE_ST_SUSPEND

1.11.2 各ドライバ状態で実行可能なドライバ関数

各ドライバ状態でコールできる（実行可能な）ドライバ関数を表 1.19 に示します。

表 1.19 ドライバ状態ー実行可能なドライバ関数対応表

ドライバ状態			ドライバ関数										
			DF_Drv_Init()	DF_Error_Management()	DF_Format()	DF_Read_Data()	DF_Write_Data()	DF_Erase_block()	DF_Blank_Number()	DF_Data_Size()	DF_Read_Protect _Cancel()	DF_WriteErase_Protect_Cancel()	DF_Read_Protect()	DF_WriteErase_Protect()
初期化前			○	×*1						○			○*2	
待機	正常状態		△*3	○	△*4	○	○	○*5	○	○			○*6	
	警告状態	未フォーマット		○*7	○	△*8	△*9	×*10						
	エラー状態	初期化実行時エラー 読み出しのみ可		○*11	×*16	○	×*12		○					
		初期化実行時エラー アクセス不可			×*1									
		フォーマット実行時エラー		○*7	△*3	×*10								
		書き込みエラー		○*11		○	×*12		○					
		消去エラー					○*13	×*12						
ビジー	ドライバ初期化処理関数実行中		×*15	×*1	×*15	×*1				○			×*14	
	データフラッシュフォーマット処理関数実行中			×*16		×*10							○*6	×*14
	有効データ読み出し処理関数実行中					×*12			△*17				×*14	○*6
	データ更新処理関数実行中					△*18	×*12						×*14	
	ブロック消去処理関数実行中					○*19	○*20	×*12					○*6	×*14
サスペンド	書き込みサスペンド中			△*19		×*12							×*14	
	消去サスペンド中													

【注】 *1 ドライバが初期化が完了されていないため、コールしないでください。

*2 ドライバ初期化処理関数内で自動的にプロテクト解除されます。

*3 再実行となります。戻り値により必要な処理を行ってください。

*4 データフラッシュフォーマット処理の再実行となります。すべてのデータが不定値となります。

*5 消去対象ブロックが存在しない場合、ドライバは、"DF_ST_OK_ERASE_BLOCK_NONE" を戻り値として返します。

*6 以後、プロテクトを解除するまでプロテクト状態に関連するドライバ関数実行時には、プロテクトエラーを戻り値として返します。

*7 エラー処理関数実行後、戻り値が "DF_ST_OK" "DF_ST_OK_REQUEST_ERASE" であった場合、データフラッシュフォーマット処理関数をコールし、フォーマット処理を行ってください。

*8 データフラッシュが未フォーマット状態のため、基本的にはコール不可ですが、一部のデータ紛失と考えられる場合、応急処置としてエラー処理関数実行後、戻り値が "DF_ST_OK" "DF_ST_OK_REQUEST_ERASE" であれば、コール可能とな

ります。ただし、紛失したデータ番号の有効データ読み出し処理を行った場合、"DF_ST_ERR_LOST_ID" を戻り値として返します。

*9 データフラッシュが未フォーマット状態のため、基本的にはコール不可ですが、*7 と同様の応急処置後、コール可能となります。ただし、リクレーム処理を伴ったうえ、リクレームデータが紛失していた場合、"DF_ST_ERR_LOST_ID" を戻り値として返します。

*10 データフラッシュが未フォーマット状態のため、コールしないでください。

*11 デバイスの状態により正常復帰しない場合があります。

*12 ドライバは、"DF_ST_WARNING_DRIVER_BUSY" を戻り値として返します。

*13 書き込み可能領域がなくなった場合、ドライバは、"DF_ST_WARNING_NO_BLANK_AREA" を戻り値として返します。

*14 実行中の関数処理が正常に行えなくなる可能性があります。コールしないでください。

*15 ドライバの状態を確認せずに実行しますので、コールしないでください。

*16 各処理中のエラー検出が正常に行えなくなる可能性があります。コールしないでください。

*17 ドライバの実行状態により正しい値が返されない場合があります。（書き込み可能領域数の増減にかかわるドライバ関数実行の場合、増減前の値が返される場合があります。）

*18 有効データ読み出し処理が優先され、ドライバが書き込み処理中であった場合、書き込みサスペンド状態となります。読み出し処理終了後、自動的にデータ更新処理を再開します。ただし、消去サスペンド中のデータ更新であった場合は、読み出し処理は行われず、ドライバは、"DF_ST_WARNING_DRIVER_BUSY" を戻り値として返します。読み出すデータ番号が更新中であった場合、更新前のデータが読み出されます。

*19 有効データ読み出し処理が優先され、ドライバが消去処理中、もしくは管理情報の書き込み中であった場合、サスペンド状態となります。読み出し処理終了後、自動的にブロック消去処理を再開します。

*20 データ更新処理が優先され、ドライバが消去処理中、もしくは管理情報の書き込み中であった場合、サスペンド状態となります。データ更新処理終了後、自動的にブロック消去処理を再開します。

1.11.3 ドライバ関数の実行優先順位

前述で示したとおり、コールできるドライバ関数の内、有効データ読み出し処理関数、データ更新処理関数、ブロック消去処理関数は、コール時に各関数内で処理が実行可能な状態かを確認し、実行できない状態の場合、ドライバは、"DF_ST_WARNING_DRIVER_BUSY" を戻り値として返します。

各関数のドライバ内での基本的な実行優先順位は、高いほうから、

① 有効データ読み出し処理関数

② データ更新処理関数

③ ブロック消去処理関数

となっており、ドライバの状態を示す RAM の値が表 1.20 に示す状態の場合、実行可能と判断します。

表 1.20 ドライバ状態－実行可能なドライバ関数対応表

ドライバ関数	実行可能と判断されるドライバ状態（RAM の値）*1		
	drv_status	drv_write_status	drv_erase_status
DF_Read_Data()	DF_DRV_ST_READY	（DF_WRITE_ST_BUSY かつ DF_ERASE_ST_SUSPEND）の状態ではない。*2	
DF_Write_Data()	-	DF_WRITE_ST_READY	-
DF_Erase_block()	-	-	DF_ERASE_ST_READY

【注】 *1 ドライバ状態を表す RAM の値の遷移は、「表 1.18 ドライバ状態-RAM 変数対応表」を参照ください。

*2 消去サスペンドを伴った書き込み処理中ではない状態

【注】 その他のドライバ関数は、コールされた場合、ドライバの状態を確認せずに実行します。

1.11.4 ドライバ関数の戻り値と必要な処理

ドライバ関数の戻り値と、その戻り値が返された場合に必要な処理を表 1.21 に示します。

表 1.21 ドライバ関数の戻り値と必要な処理

戻り値	内容	必要な処理
DF_ST_OK_ERASE_BLOCK_NONE	消去対象ブロックなし。	-
DF_ST_OK	正常終了。	-
DF_ST_OK_REQUEST_ERASE	消去対象ブロックあり。	ブロック消去処理関数をコールし、消去対象ブロックを消去してください。
DF_ST_WARNING_DRIVER_BUSY	ドライバビジー状態。	実行中のドライバ関数処理が終了してから再度実行してください。 本戻り値が返された割り込み処理からは復帰してください。
DF_ST_WARNING_NO_BLANK_AREA	書き込み対象ブロックグループ内に書き込み可能領域が残っていない状態。	ブロック消去処理関数をコールし、消去対象ブロックを消去してください。消去処理中の場合には、消去処理を完了してください。
DF_ST_WARNING_READ_PROTECT	データフラッシュ読み出しプロテクト状態。	データフラッシュ読み出しプロテクト解除処理関数をコールし、プロテクトを解除してください。
DF_ST_WARNING_WRITE_ERASE_PROTECT	データフラッシュ書き込み／消去プロテクト状態。	データフラッシュ書き込み／消去プロテクト解除処理関数をコールし、プロテクトを解除してください。
DF_ST_WARNING_INVALID_ARGUMENT	引数エラー。	引数を確認してください。
DF_ST_ERR_REQUEST_FORMAT	データ紛失。 未フォーマット状態。	データフラッシュフォーマット処理関数をコールし、ドライバにより制御するためのデータ構造を構築してください。 その際、すべてのデータ番号のデータは不定値となります。
DF_ST_ERR_LOST_ID	データ紛失。	
DF_ST_ERR_DEVICE	デバイスエラー検出。	エラー処理関数をコールしてください。 ただし、ブロック消去処理関数実行中のデータ更新処理、有効データ読み出し処理でエラーが検出された場合は、エラー処理関数をコールせず、ブロック消去処理関数の戻り値により必要な処理を行ってください。また、データ更新処理関数実行中の有効データ読み出し処理でエラーが検出された場合にも、エラー処理関数をコールせず、データ更新処理関数の戻り値により必要な処理を行ってください。 ドライバ関数の戻り値が、頻繁に本戻り値を返す場合や、エラー処理関数の戻り値が本戻り値となった場合、デバイスの限界であることが考えられます。
DF_ST_ERR_DEVICE_TIMEOUT	デバイスタイムアウトエラー検出。	
DF_ST_ERR_FCU	FCU エラー検出。	
DF_ST_ERR_INIT	ドライバ初期化エラー。	エラー処理完了後、初期化処理を再実行してください。エラー処理関数の戻り値が本戻り値となった場合、デバイスの限界であることが考えられます。
DF_ST_ERR_FORMAT	データフラッシュフォーマット実行時エラー。	エラー処理完了後、フォーマット処理を再実行してください。エラー処理関数の戻り値が "DF_ST_ERR_INIT" となった場合や、フォーマット処理再実行時の戻り値が再度本戻り値となった場合、デバイスの限界であることが考えられます。
DF_ST_ERR_PARAMETER_INVALID	パラメータエラー。	DF_user.h, DF_user.c 内の設定値を確認し、コンパイルし直してください。

1.12 ROM/RAM 容量

データフラッシュドライバソフトウェアサンプルの ROM/RAM 使用容量を以下に示します。

【注】 本項に記載しているデータは、参考データです。データ数、使用条件により異なる場合があります。

1.12.1 使用した開発ツール

High-performance Embedded Workshop Ver4.06.00

1.12.2 使用したコンパイラ

Rebesas Technology SHC/C++ Vompiler Ver 9.03.00

1.12.3 ROM/RAM 使用容量

上記に示すコンパイル条件においてデータ（データ番号）数を 8 とし、データフラッシュドライバソフトウェアサンプルをコンパイルした際の ROM/RAM 容量を以下に表 1.22 に示します。下記 RAM 使用容量には使用するスタックサイズは含まれません。

表 1.22 ROM/RAM 使用容量

ROM 使用容量	7K バイト	
RAM 使用容量	1K バイト	390 + 4 × データ数

2. データフラッシュ・ドライバソフトウェアサンプル使用時の設定方法について

本項では、本ドライバ使用時に必要な設定項目とその設定方法を示します。

2.1 設定項目

本ドライバ使用時に必要な設定項目を表 2.1 に示します。

表 2.1 ドライバ設定項目

項目	内容
データ数	データフラッシュ内に格納するデータ（データ番号）の総数。
動作周波数	使用するマイコン（SH7216）内部の動作周波数。[MHz]単位。 データフラッシュ（FLD）動作クロック 内蔵周辺モジュール用クロック
ドライバ関数内での 割り込み禁止レベル*1	各ドライバ関数処理実行時の割り込みマスクレベル。 ドライバ初期化処理関数内割り込みマスクレベル データフラッシュフォーマット処理関数内割り込みマスクレベル 有効データ読み出し処理関数内割り込みマスクレベル データ更新処理関数内割り込みマスクレベル ブロック消去処理関数内割り込みマスクレベル エラー処理関数内割り込みマスクレベル 書き込み可能領域数取得関数内割り込みマスクレベル

【注】 *1 ドライバ関数処理内でデータフラッシュへの書き込み／消去／ブランクチェックを実行する際の FCU コマンド発行時には、一時的に他の割り込みを禁止の状態（割り込みマスクレベルを"7"）にします。

2.2 データ数の設定

DF_user.h 内にある以下の定義により、データフラッシュ内に格納するデータ数を設定してください。

```
#define DF_DATA_ID_NUM          (6)
```

2.3 データサイズと格納領域の設定

DF_user.c 内にある以下の定義により、各データのサイズと各データを格納する領域（ブロックグループ）を設定してください。格納領域（ブロックグループ）の番号は、DF_user.h 内で定義されており、割り当ては、「1.4 データフラッシュ内ブロック構成」を参照ください。

```
const unsigned short DF_DATA_SIZE_GROUP[ DF_DATA_ID_NUM ][2] =
{
/*                      データサイズ                格納領域（ブロックグループ） */
/*                      0 - 256byte                 Group_A 、B                 */
/* データ番号  0 */      0,                          GROUP_A,
/* データ番号  1 */      1,                          GROUP_B,
/* データ番号  2 */      8,                          GROUP_B,
/* データ番号  3 */      9,                          GROUP_A,
/* データ番号  4 */      128,                        GROUP_A,
/* データ番号  5 */      256,                        GROUP_A,
};
```

2.4 ドライバ関数内での割り込み禁止レベルの設定

DF_user.h 内にある以下の定義により、各定義に対応するドライバ関数実行時の割り込みマスクレベルを設定することができます。

割り込みマスクレベルの設定・変更を必要としないドライバ関数に対応した定義は、コメントアウトしてください。

```
//                                  割り込みマスクレベル      対応するドライバ関数名
#define INT_LVL_df_driver_init      (7)               // (DF_Drv_Init)
#define INT_LVL_df_format           (7)               // (DF_Format)
//#define INT_LVL_df_data_read      (7)               // (DF_Read_Data)
//#define INT_LVL_df_data_write     (6)               // (DF_Write_Data)
//#define INT_LVL_df_data_erase     (5)               // (DF_Erase_block)
#define INT_LVL_df_error_management (4)               // (DF_Error_Management)
//#define INT_LVL_df_blank_number   (3)               // (DF_Blank_Number)
```

各ドライバ関数内で、上記により設定された割り込みマスクレベルに設定（変更）し、関数リターン時に設定（変更）直前のもとの割り込みマスクレベルに戻します。

3. データフラッシュ（FLD）使用上の注意事項

データフラッシュ（FLD）を使用する場合の注意事項があります。

「SH7216 グループ　ハードウェアマニュアル」の「28.8　使用上の注意事項」をご参照ください。

4. 参考ドキュメント

- ソフトウェアマニュアル
  SH-2A, SH2A-FPU ソフトウェアマニュアル　Rev.3.00
  （最新版をルネサステクノロジホームページから入手してください。）
- ハードウェアマニュアル
  SH7216 グループ ハードウェアマニュアル　Rev.1.01
  （最新版をルネサステクノロジ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

ルネサステクノロジホームページ
http://japan.renesas.com/

お問合せ先
http://japan.renesas.com/inquiry
csc@renesas.com

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2010.03.02	—	初版発行

本資料ご利用に際しての留意事項

1. 本資料は、お客様に用途に応じた適切な弊社製品をご購入いただくための参考資料であり、本資料中に記載の技術情報について弊社または第三者の知的財産権その他の権利の実施、使用を許諾または保証するものではありません。
2. 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他応用回路例など全ての情報の使用に起因する損害、第三者の知的財産権その他の権利に対する侵害に関し、弊社は責任を負いません。
3. 本資料に記載の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、あるいはその他軍事用途の目的で使用しないでください。また、輸出に際しては、「外国為替および外国貿易法」その他輸出関連法令を遵守し、それらの定めるところにより必要な手続を行ってください。
4. 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他応用回路例などの全ての情報は本資料発行時点のものであり、弊社は本資料に記載した製品または仕様等を予告なしに変更することがあります。弊社の半導体製品のご購入およびご使用に当たりましては、事前に弊社営業窓口で最新の情報をご確認いただきますとともに、弊社ホームページ（http://www.renesas.com）などを通じて公開される情報に常にご注意ください。
5. 本資料に記載した情報は、正確を期すため慎重に制作したものですが、万一本資料の記述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、弊社はその責任を負いません。
6. 本資料に記載の製品データ、図、表などに示す技術的な内容、プログラム、アルゴリズムその他応用回路例などの情報を流用する場合は、流用する情報を単独で評価するだけでなく、システム全体で十分に評価し、お客様の責任において適用可否を判断してください。弊社は、適用可否に対する責任は負いません。
7. 本資料に記載された製品は、各種安全装置や運輸・交通用、医療用、燃焼制御用、航空宇宙用、原子力、海底中継用の機器・システムなど、その故障や誤動作が直接人命を脅かしあるいは人体に危害を及ぼすおそれのあるような機器・システムや特に高度な品質・信頼性が要求される機器・システムでの使用を意図して設計、製造されたものではありません（弊社が自動車用と指定する製品を自動車に使用する場合を除きます）。これらの用途に利用されることをご検討の際には、必ず事前に弊社営業窓口へご照会ください。なお、上記用途に使用されたことにより発生した損害等について弊社はその責任を負いかねますのでご了承願います。
8. 第7項にかかわらず、本資料に記載された製品は、下記の用途には使用しないでください。これらの用途に使用されたことにより発生した損害等につきましては、弊社は一切の責任を負いません。
   1）生命維持装置。
   2）人体に埋め込み使用するもの。
   3）治療行為（患部切り出し、薬剤投与等）を行うもの。
   4）その他、直接人命に影響を与えるもの。
9. 本資料に記載された製品のご使用につき、特に最大定格、動作電源電圧範囲、放熱特性、実装条件およびその他諸条件につきましては、弊社保証範囲内でご使用ください。弊社保証値を越えて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、弊社はその責任を負いません。
10. 弊社は製品の品質および信頼性の向上に努めておりますが、特に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。弊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、社会的損害など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、誤動作防止設計などの安全設計（含むハードウェアおよびソフトウェア）およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。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、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
11. 本資料に記載の製品は、これを搭載した製品から剥がれた場合、幼児が口に入れて誤飲する等の事故の危険性があります。お客様の製品への実装後に容易に本製品が剥がれることがなきよう、お客様の責任において十分な安全設計をお願いします。お客様の製品から剥がれた場合の事故につきましては、弊社はその責任を負いません。
12. 本資料の全部または一部を弊社の文書による事前の承諾なしに転載または複製することを固くお断りいたします。
13. 本資料に関する詳細についてのお問い合わせ、その他お気付きの点等がございましたら弊社営業窓口までご照会ください。

D039444