参考資料

Application Note

Sitara デバイスにおけるシリアル NAND フラッシュの不良ブロック管理

Aryamaan Chaurasia, Vaibhav Kumar, Hong Guan, Benoit Parrot

概要

このアプリケーション ノートは、TI の Sitara™ デバイス上のシリアル NAND フラッシュ メモリの不良ブロック管理の実装について説明し、以下の 3 つの環境にわたって不良ブロックを検出、追跡、回避するためのアーキテクチャのアプローチとメカニズムについて説明します。ROM ブートローダー、TI™ MCU+ SDK、TI™ プロセッサ SDK。対象読者には、Sitara™ プロセッサ上でシリアル OSPI NAND フラッシュを使用している組込みシステム エンジニアと開発者が含まれます。

目次

商標

1 概要

NAND フラッシュ メモリ デバイスには、データを確実に格納できない欠陥のあるブロックが含まれています。製造プロセスにおいて、経済的に 100% 欠陥のない歩留まりを保証することはできないため、NAND 型フラッシュの高密度特性により、製造中に不測の欠陥が生じることがあります。メーカーはこの現実を受け入れ、事前に不良とマークされたブロックを含めて (通常は総容量の 0 ～ 2% の低い割合)、デバイスを出荷します。

これらの不良ブロックは、工場出荷時の不良ブロックとして製造プロセスに起因するものか、デバイスの動作寿命中に発生するランタイム不良ブロックである可能性があります。信頼性の高いシリアル OSPI NAND フラッシュ動作には、障害ブロックの検出、追跡、回避によりデータの整合性を確保するために、堅牢な不良ブロック管理システムが不可欠です。

このアプリケーション ノートでは、OSPI NAND フラッシュ システムで不良ブロック管理を実装するためのアーキテクチャ アプローチと主要なメカニズムについて説明し、組み込みアプリケーションのガイダンスを提供します。

2 不良ブロック管理

不良ブロック管理は、シリアル NAND フラッシュ メモリ デバイスの不良ブロックを識別、追跡、処理する、システム レベルのメカニズムです。不良ブロックとは、1 つ以上の無効なビットを含むブロックであり、データを確実に格納および取得できないブロックです。

不良ブロック マーカーはバイト パターンで、通常、NAND フラッシュのスペア / Out-Of-Band 領域にあり、使用すべきでない欠陥メモリ ブロックを識別します。不良ブロック管理は、良好なブロックに 0xFF をマーキングし、不良ブロックに 0xFF 以外の値をマーキングすることで実装されます。

- 不良ブロック管理は、いくつかの主要機能で構成されています。
  - 検出:フラッシュ デバイス内のどのブロックに障害があるかを識別します。
  - 追跡:不良ブロック位置のリストまたはテーブルを維持します。
  - 回避:フラッシュ消去、書き込み、読み取り動作中にバッド ブロックをスキップします。
  - マーキング:新たに発見された不良ブロックにマークを付けることで、将来的に回避できるようにします。

以下の図は、AM62Ax デバイスにデフォルトでマウントされている Winbond W35N01JW シリアル NAND フラッシュのメモリ アーキテクチャを示しています。

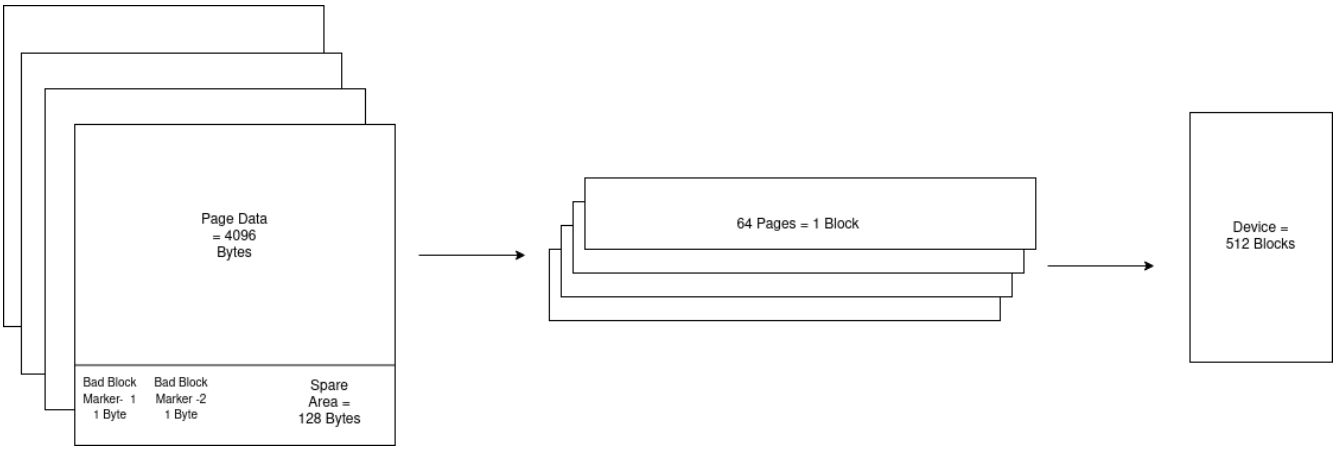

図 2-1. W35N01JW Winbond Flash 用フラッシュ メモリ アーキテクチャ

2.1 ブロックの故障

NAND フラッシュ ブロックは、製造中またはデバイスの動作寿命中に発生するさまざまな理由で故障する可能性があります。ブロックの故障には、次の要因があります。

- 製造上の欠陥:パーティクル汚染、リソグラフィ エラー、ドーピング不規則性、材料不良。
- プログラム / 消去の失敗:0xFF 状態に完全に消去できないブロックは使用できなくなります。
- 物理的損傷:電気的オーバーストレス、極値温度、機械的ストレス、放射線。

2.2 不良ブロック反転の実現可能性

不良ブロックは、一度開発した後は元に戻したり修理したりすることはできません。ブロック障害を引き起こす物理的なメカニズムは、永続的で、元に戻すことはできません。不良ブロックは、パワー サイクル全体にわたって持続するハード障害であり、消去できないものです。

3 ROM ブートローダーでの不良ブロック管理

ROM ブートローダー (RBL) は、フラッシュ メモリから (0x00 から始まる) セカンダリ ブートローダー (SBL) をロードして、ブート プロセスを開始します。データの整合性を確保するため、RBL は最初に初期ブロックの不良ブロック マーカーをチェックします。ブロックが不良としてマークされている場合、RBL は後続のブロックを繰り返しチェックし、適切なブロックが特定されるまで不良ブロック マーカーを調べます。有効なブロックが見つかると、RBL はそのブロックから SBL をロードし、信頼性の高いブート プロセスを保証します。

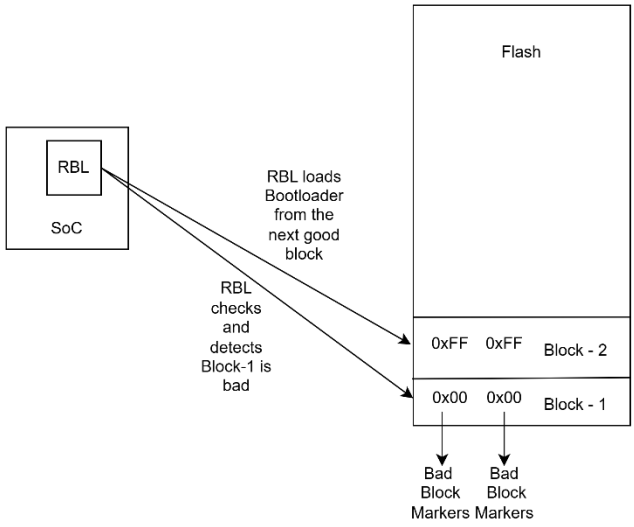

図 3-1. RBL での不良ブロックのスキップ

4 TI™ MCU+ SDK 内の不良ブロック管理

TI MCU+ SDK は、以下の主な機能を使用して不良ブロック管理アプローチを実装しています。

1. 不良ブロック リストの初期化
   - フラッシュ ドライバの初期化中、システムはデバイス内のすべてのブロックをスキャンします。
   - 各ブロックの最初のページのスペア領域は、不良ブロック マーカーをチェックするために読み取られます。
   - 不良ブロック リスト テーブルが実装され、すべてのブロックのステータスを追跡します。
2. 不良ブロックのスキップ
   - すべての読み取り、書き込み、消去の各動作は、自動的に不良ブロック リストに照会されます。
   - 不良ブロックが検出されると、動作は次の良好ブロックにスキップされます。
3. ランタイム不良ブロックのマーキング
   - 障害ステータスについては、プログラム動作と消去動作が監視されます。
   - 動作が失敗すると、影響を受けたブロックは直ちに不良としてマークされます。ステータス レジスタには次のビットが設定されます。
     – プログラム失敗 (P-FAIL): プログラム失敗ビットを使用して、P FAIL ビットをそれぞれ 0 または 1 に設定することにより、内部制御されるプログラム動作が正常に実行されたかタイムアウトされたかを示します。
     – 消去失敗 (E-FAIL): 消去失敗ビットは、E-FAIL ビットをそれぞれ 0 と 1 に設定することにより、内部で制御される消去動作が正常に実行されたかタイムアウトされたかを示すために使用されます。
   - 2 バイト サイズの不良ブロック マーカーは、各ページ内の pageSize オフセットから始まるスペア領域の先頭にあります。
   - インメモリ不良ブロック リストが更新され、新しい不良ブロックが反映されます。
   - その後の動作では、新しくマークされたブロックは自動的に回避されます。

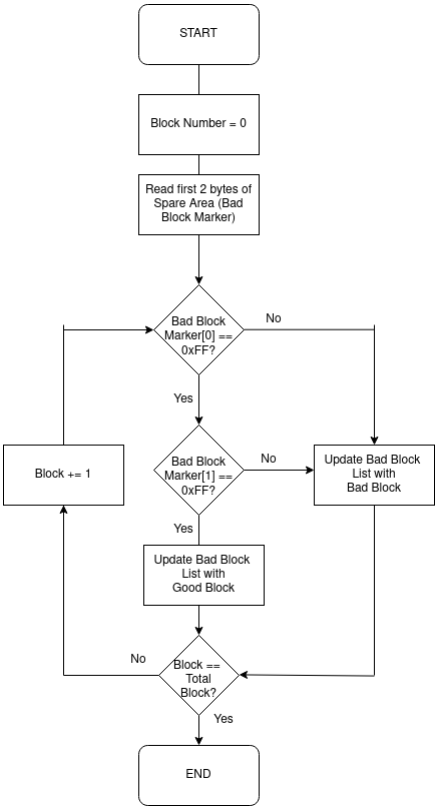

図 4-1. 不良ブロック リストの初期化

4.1 フラッシュの読み取り動作フロー

1. アドレス計算:
   - OSPI ドライバは、要求されたバイト オフセットに基づいて、ページ内の開始ブロック番号、ページ番号、およびオフセットを計算します。読み取るページ数は、転送長とページの配置に基づいて計算されます。
2. 不良ブロック スキップ:
   - 不良ブロック管理がイネーブルの場合、ドライバはターゲット ブロックの不良ブロック リストをチェックします。
     - ブロックが良好としてマークされている場合は、読み取り動作が実行されます。
     - ブロックが不良としてマークされている場合、ブロック番号とページ番号が増分して次のブロックにスキップされます。
     - このチェックは、適切なブロックが見つかるまで繰り返されます。
     - ブロック数が合計ブロック数を超えると、読み取りエラーが返されます。
3. ページごとの読み取り。
4. ブロック境界処理:
   - ページ カウンタがブロック境界を超えると、ドライバはブロック番号を再計算し、不良ブロック リストを再度チェックします。新しいブロックが不良としてマークされている場合、OSPI ドライバは自動的に次の良好なブロックにスキップし、それに応じてページ番号を調整します。
5. 完了:
   - 読み取り動作は、要求されたすべてのデータが転送されるまで続行されます。ドライバは成功を返し、不良ブロック スキップのため、基盤となる物理ブロックが連続していない可能性があったとしても、アプリケーションは連続したデータを受信します。

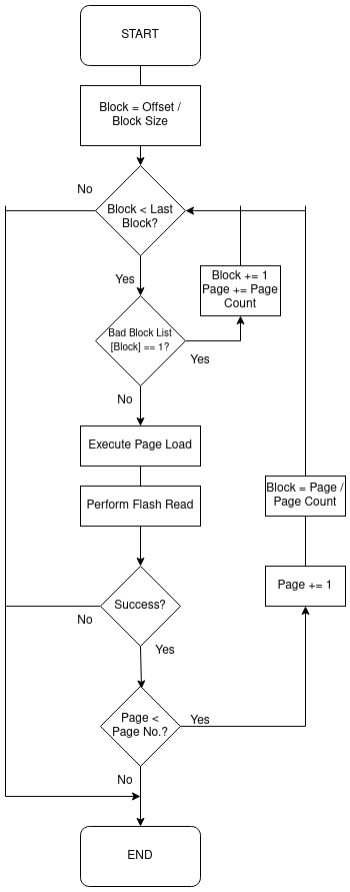

図 4-2. フラッシュの読み取りフロー

4.2 フラッシュ書き込み動作フロー

1. パラメータの検証:
   - 書き込み動作では、シリアル NAND フラッシュで必須となる「オフセットがページに整列されていること」および「オフセットに長さを加えた値がフラッシュ デバイスの容量を超えないこと」を検証します。
2. アドレス計算:
   - ドライバは、バイト オフセットから開始ページ アドレスとブロック番号を計算します。
3. 良好なブロックの配置:
   - 不良ブロック管理がイネーブルの場合、OSPI ドライバは不良ブロック リストを使用して、計算されたブロック番号から始まる次の適切なブロックを見つけます。ターゲット ブロックが不良としてマークされている場合、システムは自動的に次の良好なブロックに進みます。
4. ページごとの書き込み。
5. プログラム エラーの処理:
   - プログラム ステータス チェックでエラーが表示された場合:
     - 不良ブロック マーカーをスペア領域に書き込んだ後、ブロックは不良としてマークされます。
     - インメモリ不良ブロック リストが更新され、このブロックが不良としてマークされます。
     - 書き込み動作が失敗すると、書き込みエラーが返されます。
6. アドレスの高度化:
   - ページ書き込みが成功するたびに、ドライバはページ アドレスを増分します。ブロック境界を越えるとき、ドライバは不良ブロック リストをチェックし、必要に応じて次の適切なブロックにスキップします。
7. 完了:
   - 書き込み動作は、すべてのデータが書き込まれるまで続行されます。すべてのページが正常にプログラムされると、ドライバは成功を返します。

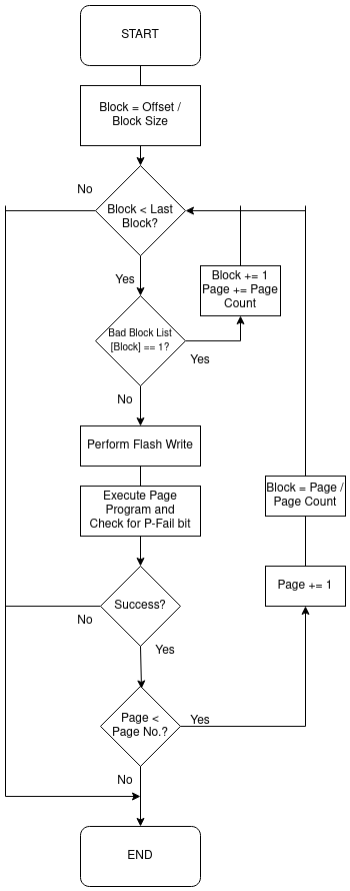

図 4-3. フラッシュ書き込みフロー

4.3 フラッシュの消去動作フロー

1. 不良ブロック チェック:
   - 不良ブロック管理がイネーブルの場合、ドライバは最初にターゲット ブロックがすでに不良としてマークされているかどうかを確認します。マークされている場合は、すぐに次のブロックにスキップして再度チェックします。これは、良好なブロックが見つかるまで続きます。
2. 消去動作を実行します。
3. 消去ステータスの検証:
   - ドライバは、タイムアウトによって消去ステータス レジスタをポーリングし、消去動作が正常に完了したことを確認します。
4. 消去エラー処理:
   - 消去ステータスが障害を示し、不良ブロック管理がイネーブルの場合:
     - 不良ブロック マーカーをスペア領域に書き込むと、ブロックは不良としてマークされます。
     - インメモリ不良ブロック リストが更新されます。
     - ブロック番号が増分され、次のブロックに移動します。
     - この消去シーケンスは次のブロックで繰り返されます。
5. BBM なしでの消去失敗:
   - 不良ブロック管理がディスエーブルになっており消去に失敗した場合、動作は再試行せず、エラーを返します。
6. 完了:
   - ブロックが正常に消去されると、動作はループを終了し、アプリケーションに成功を返します。

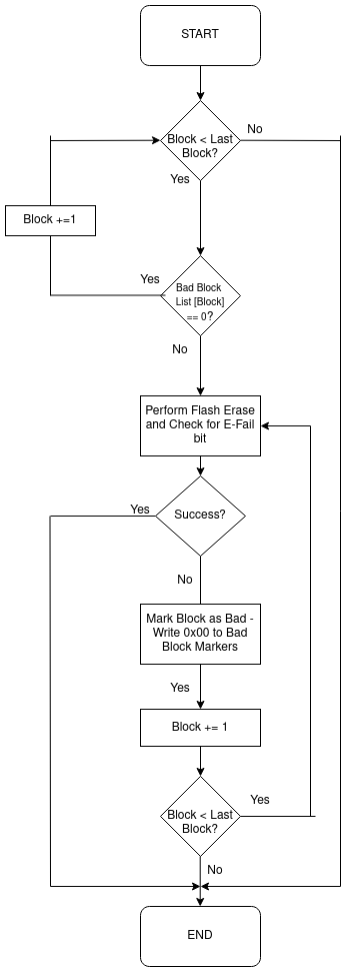

図 4-4. フラッシュの消去フロー

5 TI™ プロセッサ SDK での不良ブロック管理

Linux カーネルは、次の主な機能を備えた MTD (Memory Technology Device) サブシステムと UBI (Unsorted Block Images) を介して、不良ブロック管理を実装しています。

1. 不良ブロック テーブルの初期化
   - フラッシュドライバの初期化中に、システムはビットマップ構造体を使用してインメモリ不良ブロック テーブルを作成します。
     - 各ブロックにおける最初のページの Out-Of-Band 領域が読み取られます。
     - 不良ブロック マーカーは、通常、OOB 領域の先頭にある 2 バイトで構成されています。
     - どちらのバイトについても、マーカー値 0xFF は、適切なブロックを示します。
     - 0xFF 以外の値 (通常 0x00) は、工場出荷時にマークされた不良ブロックかランタイム不良ブロックかを示します。
   - 不良ブロック テーブルは、ブロックあたり 2 ビットを使用してステータスをエンコードするメモリで構成されます。
     - 00b = 良好なブロック。ブロックが使用可能であることを示します。
     - 01b = 摩耗またはランタイム不良ブロック。実行時に消去または書き込み動作が失敗したことを示します。
     - 10b = 予約済みブロック。
     - 11b = メーカーによってマークされた工場出荷時の不良ブロック。
   - システムが起動するたびに、フラッシュに存在するすべてのブロックの OOB マーカーをスキャンすることにより、不良ブロック テーブルが再構築されます。
2. 不良ブロックのスキップ
   - すべての読み取り動作、書き込み動作、消去動作は、物理ブロックにアクセスする前に、自動的にインメモリ不良ブロック テーブルを参照します。
   - UBI レイヤは、次の 3 つのレッド ブラック ツリーを使用して、物理消去ブロック (PEB) 組織を維持します。
     - フリー ツリー: 消去された良好な PEB がアロケーション可能で、ウェア レベリングのための消去カウントで並べかえられています。
     - 使用済みツリー: 現在ボリューム データを保存している PEB が含まれます。
     - エラー ツリー: 不良または信頼できないとマークされた PEB が含まれています。
   - UBI レイヤが書き込み用にブロックを割り当てる必要がある場合:
     - フリー ツリーから PEB が選択されます。エラー ツリーから PEB が選択されることはありません。
     - 不良ブロックのステータスは、MTD レイヤを介して BBT に照会することで確認されます。
     - ブロックが不良とマークされている場合は、フリー ツリーから除去され、エラー ツリーに追加されて、別の PEB が選択されます。
   - BBT ルックアップでは、効率的なビット操作を使用して高速アクセスを実現します。
     - BBT アレイ内のバイト オフセット = PEB 番号 ÷ 4 (各バイトが 4 つの 2 ビット エントリを保持するため)
     - バイト内のビット位置 = (PEB 番号 mod 4) ×2
   - 2 ビット ステータス コードは、次の実装を使用して抽出されます。(BBT_byte >> bit_position) & 0x03
   - 論理から物理へのアドレス変換中に不良ブロックが検出されると、動作は自動的に次の適切なブロックにスキップされ、上位レイヤに対して透過的に不良ブロックを回避します。
3. ランタイム不良ブロックのマーキング
   - プログラム、消去、読み取りの各動作の障害状態を継続的に監視するには、次の複数のメカニズムが使用されます。
     - プログラムおよび消去動作後の NAND コントローラ ステータス レジスタのチェック。
     - 読み取り動作中の ECC (Error Correction Code) エンジン モニタリング。
     - ウェア レベリング アルゴリズム限界ブロック挙動のトラッキング。
   - 動作が失敗すると、影響を受けたブロックは次の順序で即座に不良としてマークされます。
     - プログラムの失敗: 各プログラム動作の後、NAND コントローラ ステータス レジスタの P-FAIL ビットがチェックされます。P-FAIL = 1 の場合、プログラム動作がタイムアウトまたは失敗し、不良ブロックマーキングがトリガされたことを示します。
     - 消去失敗: 各消去動作の後、NAND コントローラ ステータス レジスタの E-FAIL ビットがチェックされます。E-FAIL = 1 の場合、消去操作が失敗し、不良ブロック マーキングがトリガされたことを示します。
     - ECC 修正不可能エラー: ページ内のビット エラー数が ECC 訂正機能を超える場合、つまり BCH-8 のビット数が 8 ビット以上、BCH-16 のビット数が 16 ビットを超えると、ブロックは失敗としてマークされます。

- 不良ブロックのマーキング手順は、次のステップを実行します。
  - インメモリ BBT の更新：処理に失敗したブロックの 2 ビットのステータス コードが、良好ブロックを示す 00b から、ランタイム不良ブロックを示す 01b に変更されます。
  - OOB への不良ブロック マーカーの書き込み：不良ブロック マーカー バイト (0x00,0x00) が、失敗ブロックにおける最初のページのスペア領域または OOB に書き込まれます。
  - これにより、次回のブート時にこれらのマーカーから内蔵メモリ BBT が再構築されるため、パワー サイクル間の持続が保証されます。
- UBI レイヤは、次のようにデータ構造を更新します。
  - 不良 PEB は、それが配置された場所に応じて、フリー ツリーまたは使用済みツリーから削除されます。
  - エラー ツリーに不良 PEB が追加され、将来アロケーションされないようになります。
  - 不良 PEB がボリューム データを格納していた場合、LEB と PEB のマッピングが更新され、影響を受ける論理消去ブロックが無効になります。
- 以後の動作では、新たにマークされたブロックは自動的に回避されます。このブロックがエラー ツリーに存在し、BBT ではステータス 01b になっているためです。

5.1 フラッシュの読み取り動作フロー

1. アドレス変換：
   a. UBIFS はアプリケーションからの読み取り要求を受信し、ファイル オフセットを論理消去ブロック (LEB) 番号に変換します。
   b. UBI は、LEB-to-PEB マッピング テーブルを参照して、要求されたデータを保持する物理消去ブロックを決定します。
   c. UBI は、PEB 番号に消去ブロック サイズを乗算し、ページ オフセットを追加することにより、物理フラッシュ アドレスを計算します。
2. 不良ブロック チェック：
   a. 物理フラッシュにアクセスする前に、UBI はターゲット PEB が不良とマークされていないかどうかを確認します。
   b. MTD レイヤはインメモリ バッド ブロック テーブルにアクセスし、バイト オフセットとビット位置を計算します。
   c. 2 ビットのステータス コードが抽出されます。
3. フラッシュの読み取り実行：
   a. ブロックが良好であれば、MTD は物理アドレスを使用して NAND コントローラドライバの read 関数を呼び出します。
   b. コントローラ ドライバは、フラッシュ アドレスを使用してハードウェア レジスタをプログラムし、read コマンドを発行します。
   c. NAND フラッシュ チップは、ページを内部バッファに読み取り、データをシステム メモリに転送します。
4. ECC 処理：
   a. ECC エンジンは、OOB 領域から受信したデータおよび ECC パリティ バイトを処理します。
   b. エンジンは、ビット エラーを検出するためにシンドローム計算を実行します。
   c. エラーが補正機能の範囲内にある場合、エンジンはエラーを修正し、読み取りが成功します。
   d. エラーが修正機能を超えた場合、読み取りは失敗し、ブロックは不良としてマークされます。
5. 完了：
   a. 有効なデータは MTD -> UBI->UBIFS -> アプリケーションに渡されます。
   b. UBI は、ヘッダー内のシーケンス番号と CRC 値を検証します。
   c. 修正不可能なエラーが発生した場合、エラー処理によって代替コピーからのリカバリが試行されるか、アプリケーションに I/O エラーが返されます。

5.2 フラッシュ書き込み動作フロー

1. PEB アロケーション:
   a. UBIFS は、書き込み動作で新しい論理消去ブロックの割り当てが必要かどうかを決定します。
   b. UBI はフリー ツリーから物理消去ブロックを割り当て、ウェア レベリングのために消去カウントの少ないブロックを選択します。
2. 不良ブロックの書き込み前のチェック:
   a. UBI では、選択した PEB が不良とマークされていないかどうかを確認します。
   b. MTD は BBT ルックアップを実行し、2 ビット ステータス コードを抽出します。
   c. 不良の場合、UBI はフリー ツリーから PEB を削除し、エラー ツリーに追加して、別の PEB を選択します。
3. 書き込みの準備:
   a. UBI は、ボリューム ID、LEB 番号、シーケンス番号、データ CRC、ヘッダー CRC などの UBI ヘッダーを使用してデータを準備します。
   b. MTD はコントローラ ドライバのプログラム機能を物理アドレスとデータ バッファで呼び出します。
4. フラッシュ プログラム動作:
   a. ドライバは、データ ポートを介して NAND チップのページ バッファにデータをロードします。
   b. ドライバはコントローラのコマンド レジスタに書き込み、プログラム コマンドを発行します。
5. プログラム ステータスの検証:
   a. ドライバは P FAIL ビットをチェックする NAND ステータス レジスタを読み取ります。
   b. P-FAIL = 0 は、プログラム動作が成功し、ドライバが成功を返したことを示します。
   c. P-FAIL = 1 は、プログラム動作が失敗し、ドライバがエラーを返したことを示します。
6. 成功処理:
   a. 書き込みが成功すると、UBI は LEB-to-PEB マッピン グテーブルを更新します。
   b. UBI は PEB の消去カウントを更新し、PEB をフリー ツリーから使用済みツリーに移動します。
7. エラー処理:
   a. 書き込みに失敗すると、UBI は即座にブロックを不良としてマークします。
      i. ステータスを 00b から 01b に変更して、ランタイム不良ブロックを示すインメモリ BBT を更新します。
      ii. 最初のページの OOB に不良ブロック マーカー (0x00) を書き込みます。
   b. UBI では、失敗した PEB がフリー ツリーから削除され、エラー ツリーに追加されます。
   c. UBI はフリー ツリーから新しい PEB を選択し、書き込み操作を再試行します。

5.3 フラッシュの消去動作フロー

1. 消去要求のコンテキスト:
   a. UBI ウェア レベリング アルゴリズムは、以下の期間中に消去動作をトリガします。
      i. ガベージ コレクション:無効になったデータを含むブロックを再利用します。
      ii. ウェア レベリング:データを移動して、消去カウントの分布を均等化します。
      iii. ブロック リサイクル:使用済みブロックの再アロケーションを準備します。
   b. UBI では、ウェア レベリング基準に基づいて消去するブロックを選択します。
2. 不良ブロックの消去前チェック:
   a. UBI は、ターゲット PEB 番号を使用して不良ブロック テーブルにクエリを実行します。
   b. MTD は、2 ビット ステータス コードを抽出する BBT ルックアップを実行します。
   c. ブロックがすでに不良とマークされている場合、UBI は以下を実行します。
      i. 消去動作を完全にスキップします。
      ii. 現在のツリーから PEB を除去します。
      iii. エラー ツリーに PEB を追加します。
      iv. 操作を続行する必要がある場合は、別のブロックを選択します。
3. フラッシュの消去実行:
   a. ブロックが良好な場合、UBI はブロックの物理アドレスを使用して MTD 消去機能を呼び出します。
   b. MTD はコントローラ ドライバの消去機能を呼び出します。
   c. 次に、ドライバはアドレス レジスタをブロック アドレスでプログラムし、コントローラのコマンド レジスタに erase コマンドを発行します。
4. 消去完了とステータス チェック:
   a. ドライバは、ready/busy ステータス ラインをポーリングするか、完了待ちのステータス レジスタを読み取ります。
   b. 完了後、ドライバは E-FAIL ビットをチェックするためにステータス レジスタを読み取ります。
   c. E-FAIL = 0 は、消去操作が成功し、ドライバが成功したことを示します。
   d. E-FAIL = 1 は、消去が失敗し、ドライバがエラーを返したことを示します。
5. 消去成功処理:
   a. 消去が成功した場合:
      i. UBI はトラッキング構造で PEB の消去数を増加させます。
      ii. UBI は消去された PEB をフリー ツリーに移し、アロケーション可能にします。
      iii. これで、このブロックはクリーンな消去状態になり、すべてのページはプログラミングに使用可能になりました。
6. 消去エラー処理:
   a. 消去に失敗した場合:
      i. UBI はインメモリ BBT を更新します。
      ii. 最初のページの OOB 領域に不良ブロック マーカー (0x00、0x00) が書き込まれます。
      iii. UBI データ構造を更新:
         1. 失敗した PEB が現行ツリーから除去されます。
         2. 失敗した PEB がエラー ツリーに追加されます。
         3. デバイス容量が更新され、使用可能な PEB が 1 つ少なくなります。
   b. 消去が進行中の動作の一部であった場合、UBI は異なるブロックを選択して続行します。

6 まとめ

不良ブロック管理を実装することで、組込みシステムは大容量 NAND フラッシュを活用しながらも、デバイスの寿命全体にわたってデータの整合性を維持できます。不良ブロック管理システムは、欠陥ブロックの処理の複雑さを抽象化します。これにより、ドライバが信頼性の高いフラッシュ動作を保証すると同時に、アプリケーションはコア機能に集中できるようになります。このアプリケーション ノートでは、ROM ブートローダー、TI™ MCU+ SDK、プロセッサ Linux SDK への、不良ブロック管理の実装について解説します。

7 参考資料

1. テキサス インスツルメンツ、『MCU+ SDK とプロセッサ SDK』、ソフトウェア開発キット。
2. テキサス インスツルメンツ、『AM62x テクニカル リファレンス マニュアル』、テクニカル リファレンス マニュアル。

重要なお知らせと免責事項

TI は、技術データと信頼性データ (データシートを含みます)、設計リソース (リファレンス デザインを含みます)、アプリケーションや設計に関する各種アドバイス、Web ツール、安全性情報、その他のリソースを、欠陥が存在する可能性のある「現状のまま」提供しており、商品性および特定目的に対する適合性の黙示保証、第三者の知的財産権の非侵害保証を含むいかなる保証も、明示的または黙示的にかかわらず拒否します。

これらのリソースは、TI 製品を使用する設計の経験を積んだ開発者への提供を意図したものです。(1) お客様のアプリケーションに適した TI 製品の選定、(2) お客様のアプリケーションの設計、検証、試験、(3) お客様のアプリケーションに該当する各種規格や、その他のあらゆる安全性、セキュリティ、規制、または他の要件への確実な適合に関する責任を、お客様のみが単独で負うものとします。

上記の各種リソースは、予告なく変更される可能性があります。これらのリソースは、リソースで説明されている TI 製品を使用するアプリケーションの開発の目的でのみ、TI はその使用をお客様に許諾します。これらのリソースに関して、他の目的で複製することや掲載することは禁止されています。TI や第三者の知的財産権のライセンスが付与されている訳ではありません。お客様は、これらのリソースを自身で使用した結果発生するあらゆる申し立て、損害、費用、損失、責任について、TI およびその代理人を完全に補償するものとし、TI は一切の責任を拒否します。

TI の製品は、 TI の販売条件 、 TI の総合的な品質ガイドライン 、 ti.com または TI 製品などに関連して提供される他の適用条件に従い提供されます。TI がこれらのリソースを提供することは、適用される TI の保証または他の保証の放棄の拡大や変更を意味するものではありません。 TI がカスタム、またはカスタマー仕様として明示的に指定していない限り、TI の製品は標準的なカタログに掲載される汎用機器です。

お客様がいかなる追加条項または代替条項を提案する場合も、TI はそれらに異議を唱え、拒否します。

最終更新日：2025 年 10 月