Application Note

Sitara 器件上串行 NAND 闪存的坏块管理

Aryamaan Chaurasia, Vaibhav Kumar, Hong Guan, Benoit Parrot

摘要

本应用手册介绍了 TI 的 Sitara™ 器件上串行 NAND 闪存的坏块管理实现情况，并解释了在以下三种环境中检测、跟踪和避免坏块的架构方法和机制：ROM 引导加载程序、TI™ MCU+ SDK 和 TI™ 处理器 SDK。目标受众包括在 Sitara™ 处理器上使用串行 OSPI NAND 闪存的嵌入式系统工程师和开发人员。

内容

商标

1 简介

NAND 闪存器件本身包含无法可靠存储数据的缺陷块。由于制造工艺在经济上无法保证 100% 无缺陷的良率，加之 NAND 闪存的高密度特性，因此在制造过程中出现偶发缺陷在所难免。制造商接受这一现实情况，出厂的器件中会预先标记一小部分坏块，通常占总容量的 0 – 2%。

这些坏块分为两类，一类是生产制造环节产生的工厂坏块，另一类是器件的使用寿命期间逐步出现的运行时坏块。一个稳健的坏块管理系统对于串行 OSPI NAND 闪存可靠运行至关重要，它通过检测、追踪和避开坏块来确保数据完整性。

本应用手册介绍了在 OSPI NAND 闪存系统中实现坏块管理的架构方法和关键机制，为嵌入式应用提供指导。

2 坏块管理

坏块管理是一种系统级机制，用于识别、跟踪和处理串行 NAND 闪存器件中的缺陷块。坏块是指包含一个或多个无效位，无法可靠地存储和读取数据的块。

坏块标记是一种字节模式，通常位于 NAND 闪存的备用/带外区域，用于识别不应使用的缺陷存储块。坏块管理通过将正常块标记为 0xFF，将坏块标记为非 0xFF 值来实现。

- 坏块管理包含以下几个关键功能：
  - 检测：确定闪存器件中的哪些块存在缺陷。
  - 跟踪：维护坏块位置的列表或表。
  - 避免：在闪存擦除、写入和读取操作期间跳过坏块。
  - 标记：对新发现的坏块进行标记，以便日后规避。

下图展示了 Winbond W35N01JW 串行 NAND 闪存的存储器架构，该闪存默认安装在 AM62Ax 器件上。

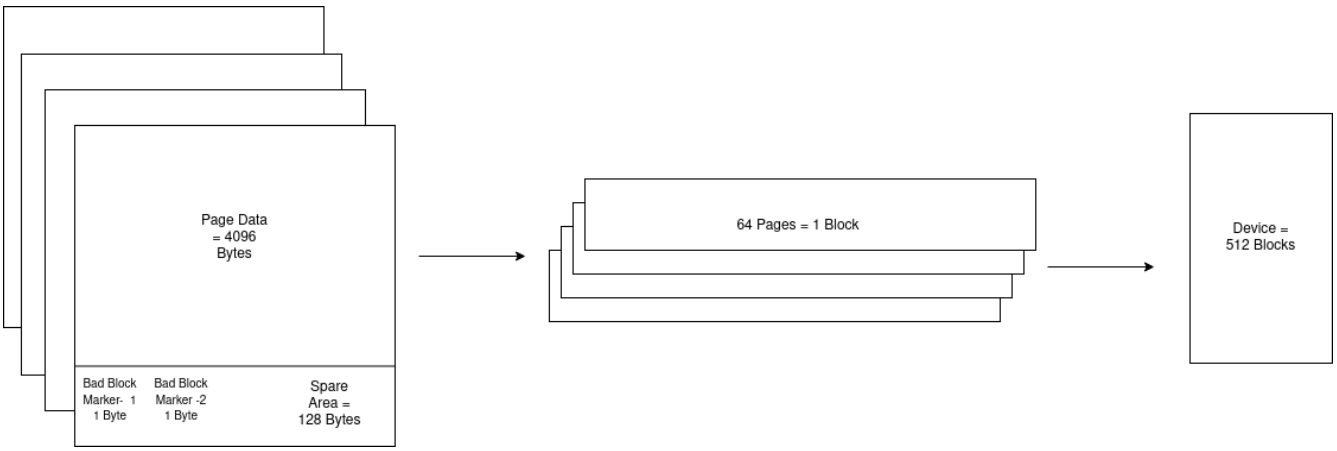

图 2-1. W35N01JW Winbond 闪存的闪存架构

2.1 块故障

NAND 闪存块可能因各种原因发生故障，这些故障可能发生在制造过程中，也可能发生在器件的使用寿命期间。以下因素会导致块故障：

- 制造缺陷：颗粒污染、光刻错误、掺杂不均匀、材料缺陷。
- 编程/擦除失败：无法完全擦除至 0xFF 状态的块将无法使用。
- 物理损坏：电气过载、极端温度、机械应力、辐射。

2.2 修复坏块的可行性

坏块一旦形成，便无法修复或恢复。导致块故障的物理机制是永久性的且不可逆。坏块属于硬性故障，这种故障会跨越电源循环持续存在，且无法清除。

3 ROM 引导加载程序中的坏块管理

ROM 引导加载程序 (RBL) 通过从闪存中加载次级引导加载程序 (SBL) 来启动引导过程，加载位置从 0x00 开始。为确保数据完整性，RBL 首先会检查初始块的坏块标记。如果块被标记为坏，RBL 会迭代检查后续块，并检查其坏块标记，直到找到一个正常块为止。一旦找到有效的块，RBL 便会从该块加载 SBL，从而确保启动过程可靠。

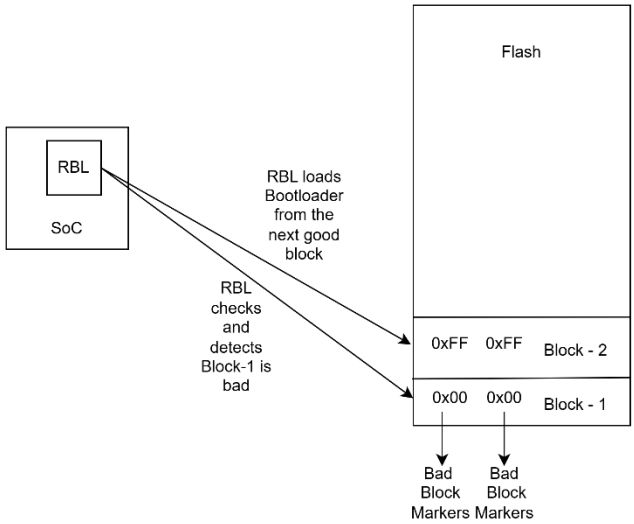

图 3-1. RBL 中的跳过坏块机制

4 TI™ MCU+ SDK 中的坏块管理

TI MCU+ SDK 实现了坏块管理方法，具有以下主要特性：

1. 坏块列表初始化
   - 在闪存驱动程序初始化期间，系统会扫描器件中的所有块。
   - 读取每个块第一页的备用区域以检查坏块标记。
   - 创建坏块列表表格，跟踪每个块的状态。
2. 跳过坏块
   - 所有读取、写入和擦除操作都会自动查阅坏块列表。
   - 遇到坏块时，操作将跳到下一个正常块。
3. 运行时坏块标记
   - 对编程和擦除操作进行故障状态监控。
   - 当操作失败时，受影响的块立即被标记为坏块。在状态寄存器中设置以下位：
     - 编程失败 (P-FAIL)：编程失败位用于指示内部控制的编程操作是执行成功还是超时，分别通过将 P-FAIL 位设置为 0 或 1 来表示。
     - 擦除失败 (E-FAIL)：擦除失败位用于指示内部控制的擦除操作是执行成功还是超时，具体通过将 E-FAIL 位分别设置为 0 和 1 来表示。
   - 大小为 2 字节的坏块标记位于备用区域的开头，该区域从每页的 pageSize 偏移量处开始。
   - 内存中的坏块列表会进行更新，以反映新的坏块。
   - 后续操作会自动避开新标记的块。

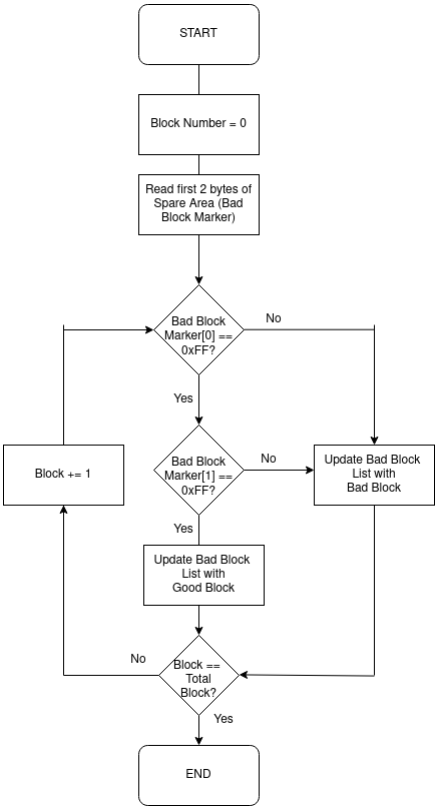

图 4-1. 坏块列表初始化

4.1 闪存读取操作流程

1. 地址计算：
   - OSPI 驱动程序根据请求的字节偏移量，计算页面内的起始块号、页码和偏移量。要读取的页数根据传输长度和页对齐情况计算得出。
2. 跳过坏块：
   - 如果启用了坏块管理，驱动程序将检查坏块列表以查找目标块：
     - 如果该块被标记为正常，则执行读取操作。
     - 如果该块被标记为坏块，则块号和页码会递增，跳到下一个块。
     - 此检查将重复进行，直到找到正常块为止。
     - 如果块号超过总块数，则返回读取错误。
3. 逐页读取。
4. 块边界处理：
   - 当页计数器跨越块边界时，驱动程序会重新计算块号并再次检查坏块列表。如果新块被标记为坏块，OSPI 驱动程序会自动跳转到下一个正常块，并相应地调整页码。
5. 完成：
   - 读取操作将持续进行，直到所有请求的数据均已传输完毕。驱动程序返回成功，应用程序将收到连续数据，即使由于跳过坏块导致底层物理块可能不连续。

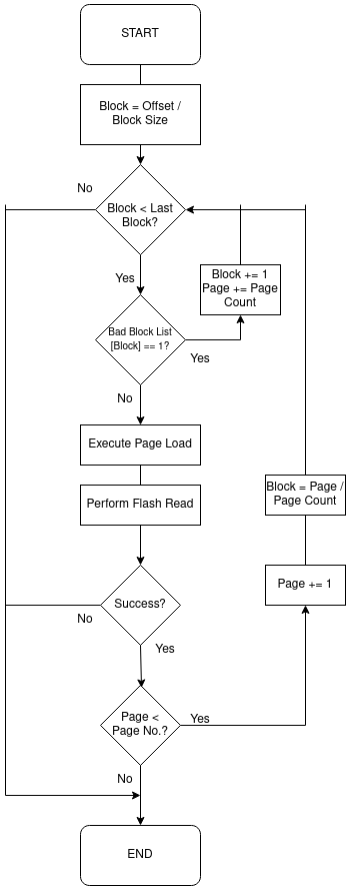

图 4-2. 闪存读取流程

4.2 闪存写入操作流程

1. 参数验证：
   - 写入操作会验证偏移量是否与页对齐（这是串行 NAND 闪存的必要条件），并验证偏移量加上长度是否不超过闪存器件的容量。
2. 地址计算：
   - 驱动程序根据字节偏移量计算起始页地址和块号。
3. 正常块的位置：
   - 如果启用了坏块管理，则 OSPI 驱动程序会使用坏块列表，从计算出的块号开始查找下一个正常块。如果目标块被标记为坏块，系统将自动跳转到下一个正常块。
4. 逐页写入。
5. 程序故障处理：
   - 如果程序状态检查显示失败：
     - 则通过将坏块标记写入备用区域，将该块标记为坏块。
     - 更新内存中的坏块列表，将该块标记为坏块。
     - 如果写入操作失败，则返回写入错误。
6. 地址递增：
   - 每次页面写入成功后，驱动程序都会递增页面地址。当跨越块边界时，驱动程序会检查坏块列表，并在必要时跳转到下一个正常块。
7. 完成：
   - 写入操作会一直持续，直到写入所有数据。成功编程所有页面后，驱动程序返回成功。

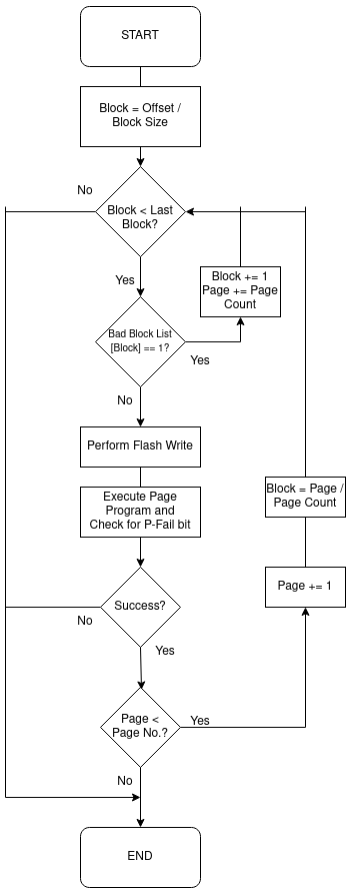

图 4-3. 闪存写入流程

4.3 闪存擦除操作流程

1. 坏块检查：
   - 如果启用了坏块管理，驱动程序将首先检查目标块是否已标记为坏块。如果是，则立即跳转到下一个块并再次检查。此过程将持续进行，直到找到一个正常块。
2. 执行擦除操作。
3. 擦除状态验证：
   - 驱动程序会设置超时并轮询擦除状态寄存器，以验证擦除操作是否成功完成。
4. 擦除失败处理：
   - 如果擦除状态指示失败且已启用坏块管理：
     - 通过将坏块标记写入备用区域，将该块标记为坏块。
     - 更新内存中的坏块列表。
     - 块号递增，转至下一个块。
     - 对下一个块重复擦除序列。
5. 未启用 BBM 时擦除失败：
   - 如果坏块管理已禁用且擦除失败，操作将立即返回错误，不进行重试。
6. 完成：
   - 成功擦除一个块后，操作退出循环并向应用程序返回成功结果。

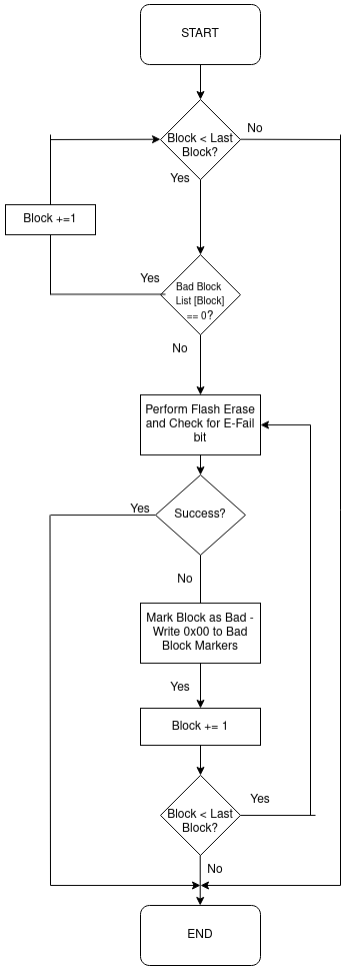

图 4-4. 闪存擦除流程

5 TI™ 处理器 SDK 中的坏块管理

Linux Kernel 通过内存技术器件 (MTD) 子系统和未排序的块映像 (UBI) 实现坏块管理，其具有以下主要功能：

1. 坏块表格初始化
   - 在闪存驱动程序初始化期间、系统使用位图结构创建一个内存中的坏块表格：
     - 读取每个块第一页的带外区域。
     - 坏块标记由 2 个字节组成，通常位于 OOB 区域的开头。
     - 两个字节的标记值均为 0xFF，则表示该块正常。
     - 任何非 0xFF 的值（通常为 0x00）均表示工厂标记的坏块或运行时坏块。
   - 在内存中构建坏块表格，每块使用 2 位对状态进行编码：
     - 00b =正常块，表示该块可供使用。
     - 01b =磨损或运行时坏块，表示在运行时擦除或写入操作失败。
     - 10b =保留块。
     - 11b =制造商标记的工厂坏块。
   - 每次系统启动时，都会通过扫描闪存中所有块的 OOB 标记来重建坏块表格。
2. 跳过坏块
   - 在访问物理块之前，所有读取、写入和擦除操作都会自动参阅存储器内的坏块表格。
   - UBI 层使用三棵红黑树维护物理擦除块 (PEB) 组织结构：
     - 空闲树：包含已擦除、可供分配的正常 PEB，按擦除次数排序以实现磨损均衡。
     - 已用树：包含当前存储卷数据的 PEB。
     - 错误树：包含标记为“不良”或“不可靠”的 PEB。
   - 当 UBI 层需要为写入操作分配一个块时：
     - 从空闲树中选择 PEB。绝不会从错误树中选择 PEB。
     - 通过 MTD 层查询 BBT 来验证坏块状态。
     - 如果该块被标记为坏块，则将其从空闲树中移除，添加到错误树中，并选择另一个 PEB。
   - BBT 查找使用高效位操作以实现快速访问：
     - BBT 数组中的字节偏移量= PEB 编号÷ 4（因为每个字节包含四个 2 位条目）
     - 字节内的位位置=（PEB 编号 mod 4）× 2
   - 2 位状态码是使用以下实现方式提取的：(BBT_byte >> bit_position) & 0x03
   - 在逻辑地址到物理地址的转换过程中遇到坏块时，操作会自动跳转到下一个正常块，从而使上层对坏块的规避过程透明。
3. 运行时坏块标记
   - 通过多种机制持续监控编程、擦除和读取操作是否存在故障情况：
     - 编程和擦除操作后检查 NAND 控制器状态寄存器。
     - 在读取操作期间由纠错码 (ECC) 引擎进行监控。
     - 通过磨损均衡算法跟踪临界块的行为。
   - 当操作失败时，将通过以下顺序立即将受影响的块标记为坏块：
     - 编程失败：每次编程操作后，将检查 NAND 控制器状态寄存器的 P-FAIL 位。如果 P-FAIL=1，则表明编程操作超时或失败，从而触发坏块标记。
     - 擦除失败：每次擦除操作后，将检查 NAND 控制器状态寄存器的 E-FAIL 位。如果 E-FAIL = 1，则表示擦除操作失败，从而触发坏块标记。
     - ECC 不可纠正的错误：当一页中的位错误数量超过 ECC 纠错能力时，即 BCH-8 超过 8 位或 BCH-16 超过 16 位，该块被标记为故障。
   - 在坏块标记过程中，会执行以下步骤：
     - 更新内存中的 BBT：将故障块的 2 位状态码从表示正常块的 00b 更改为表示运行时坏块的 01b。
     - 将坏块标记写入 OOB：坏块标记字节 (0x00,0x00) 被写入故障块第一页的备用区域或 OOB。
     - 这可确保数据在电源循环中持久保存，因为下次启动时，内存中的 BBT 将根据这些标记重新构建。
   - UBI 层更新其数据结构：
     - 根据该故障 PEB 所在的位置，将其从空闲树或已用树中移除。
     - 该故障 PEB 被添加到错误树中，以防止未来对其进行分配。

- 如果该故障 PEB 曾存储卷数据，则更新 LEB 到 PEB 的映射关系，以使受影响的逻辑擦除块失效。
- 后续操作会自动避开该新标记的块，因为它现在位于错误树中，且在 BBT 中状态为 01b。

5.1 闪存读取操作流程

1. 地址转换：
   a. UBIFS 接收来自应用程序的读取请求，并将文件偏移量转换为逻辑擦除块 (LEB) 编号。
   b. UBI 查询其 LEB 到 PEB 的映射表，以确定哪个物理擦除块包含所请求的数据。
   c. UBI 通过将 PEB 编号乘以擦除块大小，再加页面偏移量，来计算物理闪存地址。
2. 坏块检查：
   a. 在访问物理闪存之前，UBI 会验证目标 PEB 是否未被标记为坏块。
   b. MTD 层访问内存中的坏块表格，计算字节偏移量和位的位置。
   c. 提取 2 位状态码。
3. 闪存读取执行：
   a. 如果块正常，MTD 将使用物理地址调用 NAND 控制器驱动程序的读取功能。
   b. 控制器驱动程序使用闪存地址对硬件寄存器进行编程并发出读取命令。
   c. NAND 闪存芯片将页面读入其内部缓冲区，并将数据传输到系统内存。
4. ECC 处理：
   a. ECC 引擎处理从 OOB 区域接收到的数据和 ECC 奇偶校验字节。
   b. 该引擎执行综合码计算以检测位错误。
   c. 如果错误在纠错能力范围内，则引擎会纠错，读取成功。
   d. 如果错误超出纠错能力，则读取失败，该块被标记为坏块。
5. 完成：
   a. 有效数据从 MTD -> UBI -> UBIFS -> 应用程序依次传输。
   b. UBI 会验证标头中的序列号和 CRC 值。
   c. 如果出现不可校正的错误，在处理错误时会尝试从备用副本中恢复数据，或向应用程序返回 I/O 错误。

5.2 闪存写入操作流程

1. PEB 分配：
   a. UBIFS 确定写入操作是否需要分配新的逻辑擦除块。
   b. UBI 从空闲树中分配一个物理擦除块，为实现磨损均衡，选择擦除次数较少的块。
2. 写入前坏块检查：
   a. UBI 验证所选 PEB 是否未被标记为坏块。
   b. MTD 执行 BBT 查找以提取 2 位状态码。
   c. 若存在坏块，UBI 将该 PEB 从空闲树中移除，添加到错误树中，并选择另一个 PEB。
3. 写入准备：
   a. UBI 准备数据并添加 UBI 标头：卷 ID、LEB 编号、序列号、数据 CRC、标头 CRC。
   b. MTD 调用控制器驱动程序的编程函数，并传入物理地址和数据缓冲区。
4. 闪存编程操作：
   a. 驱动程序通过数据端口将数据加载到 NAND 芯片的页面缓冲区中。
   b. 驱动程序通过向控制器的命令寄存器写入数据来发出编程命令。
5. 程序状态验证：
   a. 驱动器读取 NAND 状态寄存器检查 P-FAIL 位。
   b. P-FAIL = 0 表示编程操作成功，驱动程序返回成功。
   c. P-FAIL = 1 表示编程操作失败，驱动程序返回错误。
6. 成功处理：
   a. 如果写入成功，UBI 会更新 LEB 到 PEB 映射表。
   b. UBI 会更新 PEB 的擦除计数并将 PEB 从空闲树移至已用树。
7. 故障处理：
   a. 如果写入失败，UBI 会立即将该块标记为坏块：
      i. 通过将内存中 BBT 的状态从表示正常块的 00b 更改为表示运行时坏块的 01b，来更新内存中的 BBT。
      ii. 将坏块标记 (0x00) 写入第一页的 OOB。
   b. UBI 从空闲树中删除失败的 PEB 并添加到错误树中。
   c. UBI 从空闲树中选择新的 PEB 并重试写入操作。

5.3 闪存擦除操作流程

1. 擦除请求背景：
   a. UBI 磨损均衡算法在以下情况下触发擦除操作：
      i. 垃圾回收：回收包含过时数据的块。
      ii. 磨损均衡：移动数据以均衡擦除次数的分布。
      iii. 块回收：准备已用块以供重新分配。
   b. UBI 根据磨损均衡标准选择待擦除的块。
2. 擦除前检查坏块：
   a. UBI 使用目标 PEB 编号查询坏块表格。
   b. MTD 执行 BBT 查找，提取 2 位状态码。
   c. 如果该块已被标记为坏块，则 UBI：
      i. 完全跳过擦除操作。
      ii. 从当前树中删除 PEB。
      iii. 将 PEB 添加到错误树中。
      iv. 如果操作需要继续，则选择另一个块。
3. 闪存擦除执行：
   a. 如果块正常，UBI 会使用块的物理地址调用 MTD 擦除函数。
   b. MTD 调用控制器驱动程序的擦除函数。
   c. 然后，驱动器使用块地址对地址寄存器进行编程，并向控制器的命令寄存器发出擦除命令。
4. 擦除完成和状态检查：
   a. 驱动程序轮询就绪/繁忙状态线路或读取状态寄存器，等待操作完成。
   b. 完成后，驱动程序读取状态寄存器以检查 E-FAIL 位。
   c. E-FAIL = 0 表示擦除操作成功，驱动程序返回成功。
   d. E-FAIL = 1 表示擦除失败，驱动程序返回错误。
5. 擦除成功处理：
   a. 如果擦除成功：
      i. UBI 会在跟踪结构中递增 PEB 的擦除计数。
      ii. UBI 将已擦除的 PEB 移至空闲树，使其可供分配。
      iii. 该块现在处于干净的擦除状态，所有页面均可用于编程。
6. 擦除失败处理：
   a. 如果擦除失败：
      i. UBI 会更新内存中的 BBT。
      ii. 坏块标记 (0x00,0x00) 将写入第一页的 OOB 区域。
      iii. 更新 UBI 数据结构：
         1. 故障 PEB 将从当前树中删除。
         2. 故障 PEB 被添加到错误树中。
         3. 器件容量会更新，反映可用 PEB 减少一个。
   b. 如果擦除是正在进行的操作的一部分，UBI 将选择另一个块并继续操作。

6 总结

通过实施坏块管理，嵌入式系统可以利用高容量 NAND 闪存，同时在整个器件生命周期内保持数据完整性。坏块管理系统在处理缺陷块的复杂性时会进行抽象化处理，使应用程序能够专注于其核心功能，而驱动程序则确保闪存的可靠运行。本应用手册介绍了如何在 ROM 引导加载程序、TI™ MCU+ SDK 和处理器 Linux SDK 中实现坏块管理。

7 参考资料

1. 德州仪器 (TI)，MCU+ SDK 和处理器 SDK 及软件开发套件。
2. 德州仪器 (TI)，AM62x 技术参考手册，技术参考手册。

重要通知和免责声明

TI“按原样”提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做出任何明示或暗示的担保，包括但不限于对适销性、与某特定用途的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他安全、安保法规或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的相关应用。严禁以其他方式对这些资源进行复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。对于因您对这些资源的使用而对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，您将全额赔偿，TI 对此概不负责。

TI 提供的产品受 TI 销售条款)、TI 通用质量指南 或 ti.com 上其他适用条款或 TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。 除非德州仪器 (TI) 明确将某产品指定为定制产品或客户特定产品，否则其产品均为按确定价格收入目录的标准通用器件。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

最后更新日期：2025 年 10 月