Bad Block Information Introduction

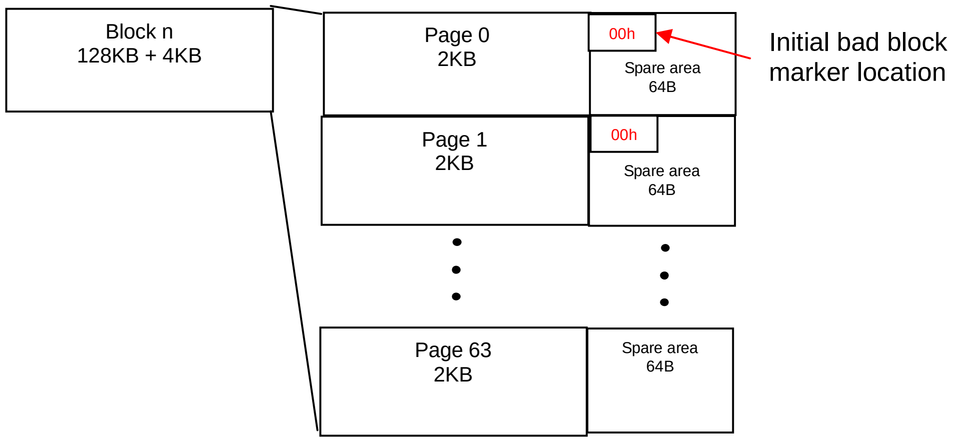
1. Introduction

Today, NAND flash is used in many fields, such as consumer, industrial, and automotive. Compared with NOR flash, NAND flash has the advantage of availability at higher densities and lower cost per bit. However, NAND flash has the disadvantage of requiring system management of bad blocks, while NOR flash does not. This application note describes how Macronix marks bad blocks in NAND flash and recommends the creation and usage of a bad block table to properly manage bad blocks.

2. Initial Bad Block Marker

NAND flash cell structure is different from NOR flash and is allowed to ship with up to 2% of its blocks initially marked as bad. Bad blocks should never be used again for data storage. It is necessary to know how the flash vendor marks bad blocks before a system can locate and avoid using them. Macronix follows the industry standard method of marking bad blocks by programming non-FFh data into the 1st byte of the 1st or 2nd page of a block's spare area. Figure 2-1 shows the initial bad block marker location in the spare area.

Figure 2-1: Macronix NAND flash Memory Array Structure



3. Bad Block Table Build Up

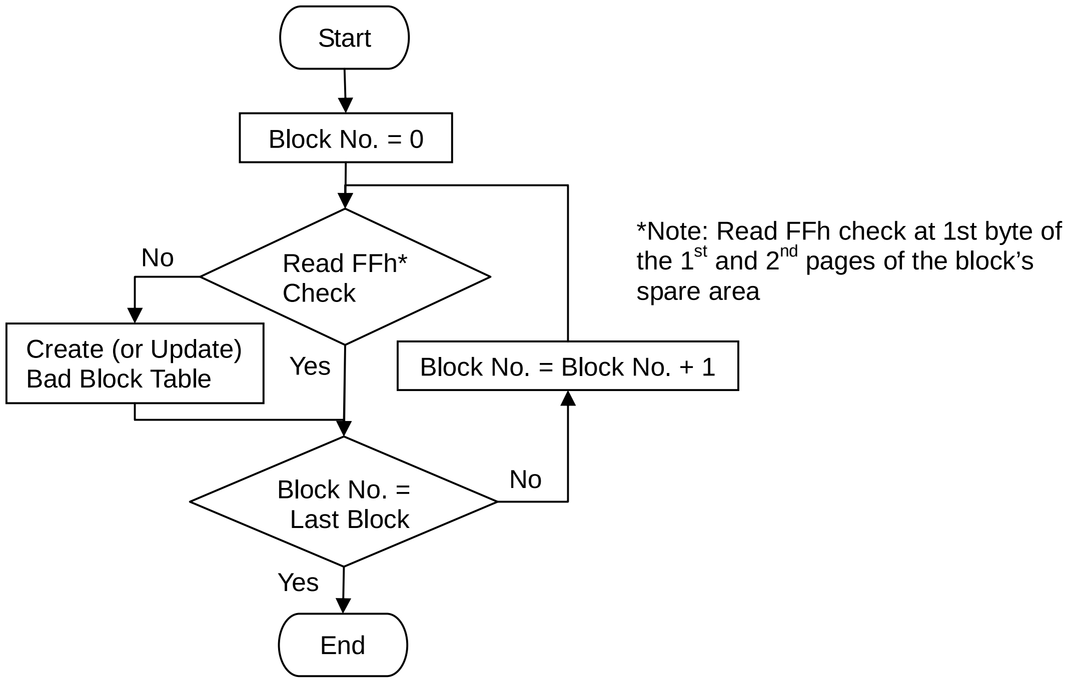
Although the initial bad blocks are marked by the flash vendor, they could be inadvertently erased and destroyed by a user that does not pay attention to them. To prevent this from occurring, it is necessary to always know where any bad blocks are located. Continually checking for bad block markers during normal use would be very time consuming, so it is highly recommended to initially locate all bad blocks and build a bad block table and reference it during normal NAND flash use. This will prevent having the initial bad block markers erased by an unexpected program or erase operation. Failure to keep track of bad blocks can be fatal for the application. For example, if boot code is programmed into a bad block, a boot up failure may occur. The following section shows the recommended flow for creating a bad block table.

Bad Block Information Introduction

3-1. Bad Block Table Creation

Before a system uses NAND flash for storage, it is highly recommended that a bad block table be created from any initial bad block markers inserted by the NAND flash vendor and any additional bad block markers that may have been inserted by the system. As shown in Figure 3-1, this can be accomplished by scanning the spare area of all memory blocks and creating or updating a bad block table whenever bad block markers are found (and programming non-FFh data into the 1st byte of the 1st or 2nd page of the block's spare area). During the normal course of use, additional bad blocks may be found by the system and should be marked as such so that they will be located during the next scan for bad block markers. This will ensure that the bad block table is always updated with the current bad block information. The system may then reference this table prior to selecting good blocks for storage operations.

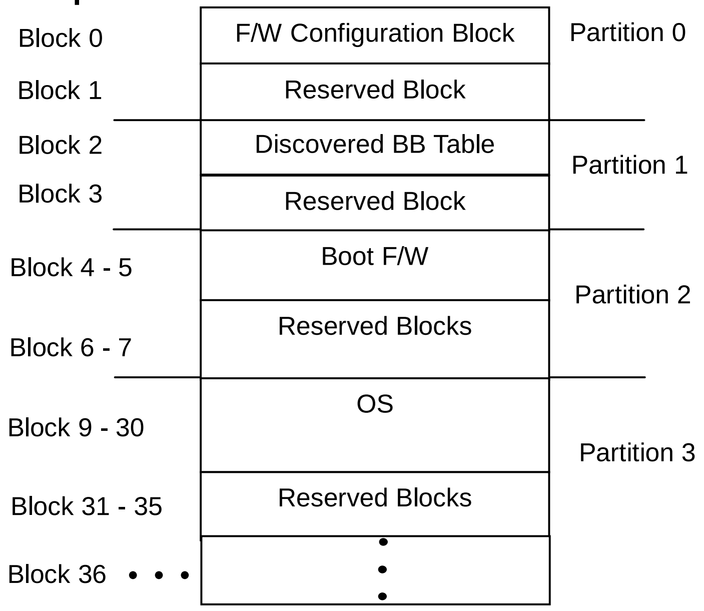
Figure 3-1: Bad Block Table Creation Flow


Start
Block No. = 0
Read FFh* Check
No
Yes
Create (or Update) Bad Block Table
Block No. = Block No. + 1
Block No. = Last Block
No
Yes
End
*Note: Read FFh check at 1st byte of the 1st and 2nd pages of the block's spare area

4. Bad Block Management – Skip Bad Block & Partitioning

Initial bad blocks are allowed in NAND flash, so bad block management is necessary to avoid data storage in a bad block. Bad blocks create a a non-continuous structure, and skipping bad blocks is the simplest method of bad block management. In addition, a Partition structure is also generally used to help with bad block management in NAND flash. It is a common practice for NAND flash to be partitioned into regions large enough to contain different types of code (ie. F/W Configuration Block, Discovered Bad Block Table, Boot F/W, OS, etc.). This is typically done to ensure that specific code can be located at a predetermined known physical address. It is important that the partition size be large enough to contain not only the code to be stored there, but also contain enough reserve block padding to prevent partition encroachment into adjacent partitions. Partition encroachment could occur if bad blocks are located in the partition during initial programming. If the partitioning is too tight, unacceptable yield loss may result during initial programming in NAND devices which shipped as good from the factory. Figure 4-1 shows one of partition example for reference.

Figure 4-1: Partition example



5. Summary

NAND flash may be shipped with some bad blocks, which will have been marked by the NAND flash vendor. Before using NAND flash, it is necessary to locate all initial bad blocks and any new bad blocks to avoid their use for storage. Creating and using a bad block table will simplify the task of managing bad blocks and prevent the accidental destruction of bad block markers.

6. Revision History

Revision	Description	Date
1.0	Initial Release	Dec. 11, 2013

Bad Block Information Introduction