1000 Logical Block Skip User Manual

General Description and Name

This scheme treats each group of 1024 blocks as a "Logical Block" in the device. The first 1000 blocks make up the main array and the last 24 blocks act as a buffer area. However, the first block on the device contains boot code so the buffer area of the first Logical Block is only 23 blocks in size. Any bad block found within each Logical Block is simply skipped and then the buffer at the end is shrunk by the amount of bad blocks found in each Logical Block.

The customer's data file does not account for the 24 block buffer area so each time the programmer reaches the border of a Logical Block it must then skip ahead by 24 blocks (minus the number of bad blocks in the current Logical Block) to the start of the next Logical Block.

Relevant User Options

The following special features on the special features tab apply to this scheme. The default values might work in some cases but please make sure to set the right value according to your system.
Please note only the below special feature items are related to this scheme and ignore any others. If any of below items doesn't exist, please check whether the right version has been installed or contact Data I/O for support by submitting Device Support Request through this address:
http://www.dataio.com/support/dsr.asp

Bad Block Handling Type = "1000 Logical Block Skip"

Spare area : Please refer to "Description of common NAND special features.pdf". ***Normally set as " Enabled" or "Disabled" for this BBM***.[Default 'Disabled']

Required good block area: Start block = "0" Please refer to "Description of common NAND special features.pdf".

Required good block area: Number of blocks = "0" Please refer to "Description of common NAND special features.pdf".

Special Notes

This scheme is very simple. However, it should be known that the algorithm will automatically reserve a certain amount of space for bad blocks. This number depends on the device. For example, if a certain device can have up to 70 bad blocks before failing,

then the algorithm will program the total # of blocks – 70 into the device. Once it has successfully programmed that many, it will stop programming. In other words, you cannot program an image that requires every single block of the device due to the nature of NAND. The same idea applies to the load from master operation.

The spare area in this scheme can either be programmed with the customer's image file, or it can be ignored. ECC is not an option with this particular scheme. However, the bad block marks are always located in the spare area (Byte 517 for x8 devices). Generally the customer's data file will contain the Spare Area information but it is not necessary.

The data file doesn't have to be arranged in any special way for this scheme. The binary that should be placed into the device is all that is needed. However, special care should be taken into account if the spare area option is set to "enabled". In that case, byte 517 of each page (the sixth byte of the spare area in each page) needs to be left at 0xFF. This is because byte 517 is used to identify bad blocks in the device. If you program one of these bytes to something other than 0xFF, there will be no way for anyone to distinguish a factory marked bad block from a block that has had byte 517 programmed by the programmer.

If the spare area is not to be programmed, then the image should not contain any data for the spare area.

Revision History

V1.0 Date
Create this spec.

Appendix

You can get the file "Description of common NAND special features.pdf" from http://ftp.dataio.com/FCNotes/BBM/