Description of common NAND special features

General Description

This is a conclusion of such special features which have been widely shared within NAND device supports, they might relate to device features or different handling options but not directly BBM related.

Glossary of Terms

BBM	Bad Block Management scheme
ECC	Error Correction Code
SLC	Single Level Cell technology
MLC	Multi-Level Cell technology
OOB Region	Out of Band region in a page. It is also called Spare Area.
semi vendor BB marker	Data located OOB region of a either the first or second page of a block in a device that the manufacturer uses to identify if a block is good or bad.
TLWin	TaskLink

List of special features and options

Bad Block Handling Type
This item specify which BBM (Bad block management type) will be used.
[Always required, no default value]

Spare area
Disabled - Data file does not contain Spare Area data. TLWin will insert 0xFF into the Spare Area of data file.
Enabled - Data file contains Spare Area data that will be used directly.
ECC - Data file doesn't contain Spare Area data. TLWin will compute ECC based on your Bad Blocking Scheme to fill the Spare Area.
Update ECC field - Data file contains Spare Area data but our system will update the ECC related field with proper value.
[Always required, default value not fixed.]

Spare area size in data file
How many bytes of spare area in data file. If it's value is not 65535. TLWin will reduce or expand spare area in data file to device's spare area size.(ex. If spare area bytes in data file is 64, but spare area size of device is 128. Then set it's value to 64. and TLWin wil auto expand spare area bytes in data file to 128.)
[Normally not required, default '65535' if not listed or specified.]

Error bits allowed in one page
How many error bits allowed within one page while verify, this depends on the features of NAND (SLC, MLC …).
[Normally required, default '0' if not listed or specified.]

Error bits allowed per sector
sector structure for error bits
These two items can be used to specify the sector structure under page and how many error bits allowed per sector. Works only on FC3 machine even they exist.
528B (cont. from MNA to SPA) - Each sector contains 528 continuous bytes within the page.
528B (512B of MNA + 16 of SPA) - Each sector contains 512 bytes within main area and 16 bytes within the spare area of the page.
AutoSize (continuous from main area to spare area) – Same as `528B (cont. from MNA to SPA)` but sector size is calculated automatically as 512 + (spare area / (main area size / 512)).
AutoSize (512B of main area + num of spare area) - Same as `528B (512B of MNA + 16 of SPA)` but sector size is calculated automatically as 512 + (spare area / (main area size / 512)).
[Normally not required, default ‘0’ for ‘Error bits allowed per sector’ that this feature disabled.]

*The upper three items make up the error bits specification; they should be set according to the device specification together with the ECC capability of target system. If either of these two limitations doesn’t match, the device will be failed. However for some latest NAND devices, Error bits allowed in one page will be disabled while its value > 254 and Error bits allowed per sector enabled, a log could be found within eventlog if so.

Error bits allowed per chip
How many error bits allowed per chip while verify. This function can work on FC2 and FC3.
[Normally not required, default ‘0’ for ‘Error bits allowed per chip’ that this feature is disabled.]

Check BB Marker In DataFile
Whether check the bad block mark position of the customer data file or not. Every NAND chip has its own semi-vendor specified bad block mark located in spare area, if this item enabled, programmer system will check the contents of this marker location and throw error if not all 0xFF.
[Normally not required, default ‘Enabled’ if not listed or specified.]

Required good block area: Start block
Required good block area: Number of blocks
These two items specify a range of blocks which must all be good. The device will be failed if any bad blocks found within this area.
[Always required, default ‘0’ and ‘0’.]

Force OTP Blank Check
Disable - Programmer system will do blankcheck on OTP area according to the programming flag of the corresponding sectors.
Enable - Programmer system will always do blankcheck on OTP area.

[Normally not required, default ‘Disable’ if not listed. Otherwise default value not fixed.]

<u>Skip Free Pages</u>
Disabled - Work as normal, all the data include blank pages will be programmed.
Enabled - The blank pages(all bytes with 0xFF data) of data file will skip programming operation while program function run.
[Normally not required, default ‘Disabled’.]

<u>Skip Last Free Space</u>
Disabled - Work as normal, all the data include blank pages will be programmed.
Enabled - The continuous blank pages(all bytes with 0xFF data) located at the end of a block of data file will skip programming operation while program function run.
[Normally not required, default ‘Disabled’.]

<u>Only Cal ECC for User Data</u>
YES - Work as normal, if set ‘spare area’ as ‘ECC’, static ECC will only be calculated for user data file.
NO - If set ‘spare area’ as ‘ECC’, static ECC will be calculated for all the NAND range.
[Normally not required, default ‘YES’.]

<u>Handle Empty Page</u>
YES - Work as normal.
NO - Any pages which has all 0xFF within the data file will not be handled in the related DLL of TLwin, which means ECC field or any other tag locations will not be updated, just keep the whole page includes the OOB(might be inserted according to the settings of ‘spare area’) as all 0xFF.
[Normally not required, default ‘YES’.]

<u>Fill00 to Initial BB</u>
Disabled - Not change the original(semi-vendor marked) bad blocks.
Enabled - Program the semi-vendor specified bad block marker existed page(s) to all 0x00 for all original bad blocks. PLEASE NOTE PROGRAMMER SYSTEM WILL ONLY DO THE PROGRAMMING OPERATION BUT WILL NOT CHECK THE RESULT!
[Normally not required, default ‘Disabled’.]

<u>bad block detection</u>
This item is useful when customer needs to move the bad block marker location or want to override default bad block markers but use bad block table to record the information. Then we can only use the BBM specified way to detect the bad blocks while reprogramming.
semi vendor BB marker - The normal way, found the bad bock according to semi-vendor specified bad block marker. Actually the algorithm will work as earlier source code if this option selected.

BBM specified - This option will give the detection work to BBM. So, it can use any bad block marker location even in main area or bad block table to detect the bad blocks.
BBM then BB marker - Firstly work as "*BBM specified*", if it failed to find the bad block information, then work as "*semi vendor BB marker*" to find the bad blocks.
[Normally not required, default 'semi vendor BB marker'.]

<u>BB: max allowed</u>
How many bad blocks allowed for this chip. BBM can override this setting.
[Normally not required, default as semi vendor specified.]

<u>BB: mark position</u>
<u>BB: mark mask</u>
Specify bad block mark position within spare area and its mask. Position is 0 based and LOGIC_1 of the mask means this bit is not care. BBM can override this setting.
[Normally not required, default as semi vendor specified.]

<u>forcely close safety ID check</u>
YES - If 'Electronic ID Check' under 'Process' tag is also unchecked, the system will not do any ID check operation. This option is not suggested, however, it gives the possibility if it's really needed in some special & urgent case.
NO - Work as normal.
[Normally not required, default 'NO'.]

<u>erase error handling</u>
<u>program error handling</u>
These two items only work if we had enabled the related function in the support, they can specify how to handle erase and program error. PLEASE NOTE WE WILL NOT ENABLE THE RELATED FUNCTIONS UNLESS CUSTOMER REQUIRED AND TOTALLY UNDERSTAND THE RISK AND AFFORD RESULT!!
fail the chip - fail the chip if any erase or program error met.
mark as BB and fail - mark the error occurred block of the chip as bad block and then fail the chip.
mark as BB and go on - mark the error occurred of the chip as bad block and continue process.
[Normally not required, default 'fail the chip'.]

<u>ERASE AFTER PROGRAM</u>
Actually this item not only related to NAND devices. If enabled, our programmer system will erase the whole chip after programming. This is mainly for flash testing usage.
[Normally not required, default 'Disabled' if not listed or specified.]

<u>Hardware ECC</u>
It is for device which has Hardware ECC calculation capability, e.g. OneNAND, etc.
Disabled - Hardware ECC calculation is not allowed
Enabled - Hardware ECC calculation is allowed.

[Normally existed only for capable devices, default 'Enabled' if listed or 'Disabled' if not listed.]

Special Notes

1. Customer should check and set the corresponding value per requirement, but NOT rely on the default value.
2. As devices have different features and customers have different requirement, some of these listed special features might not be added for some device. It's suggested to issue a DSR if any needed items missed.
3. All BBMs default don't support read operation by itself unless special note in its user manual indicates it supports. The BBM 'None' or 'BBM_ReadBack' can help normally.

History

V1.01
Dec 9th, 2010, added Handle Empty Page
V1.02
Jan 6th, 2011, added forcely close safety ID check
V1.03
Jan 17th, 2013, added Error bits allowed per sector and sector structure for error bits
V1.04
Jul 24th, 2013, updated the comments for error bits.
V1.05
Feb 21st, 2014, updated the comments for BBM read.
V1.06
June 27st, 2014, updated for Hardware ECC and Special notes.
V1.07
July 23rd, 2014, added Error bits allowed per chip.
V1.08
August 20th, 2014, updated for Spare area.
V1.09
December 16th, 2014, added Spare area size in data file
V1.10
September 23 rd, 2015, updated for sector structure for error bits