Multiple Partition UBoot

General Description and Name

This scheme is a multiple partition scheme similar to generic multiple partition. It requires an .mbn partition file with the usual block start, block end and length fields. The scheme will write a bad block table to the last 2 blocks of the device. The last block of the device will be the primary bad block table marked with Bbt0, the second to last block will be a secondary table marked with 1tbB. The last 4 blocks of the device must be good or the part will be rejected. The bad block tables written will overwrite the manufacture's bad block markers. The image data can be either in large format pages or small format pages. The bad block manager will detect the bad block marks regardless of format..

Spare Area

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. The bad block marks are always located in the spare area.

ECC

This bad block manager uses a customer provided ECC scheme to calculate the ECC bytes for the generated bad block tables. The code for the ECC generation was taken from the provided mkflash program. Mkflash was built and used on a linux system due to it's reliance on some linux disk I/O functions. The ECC is calculated on a subpage of the full 2048 byte page and includes the 1st 7 bytes of the spare area record associated with that subpage.

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

Please refer to "Description of common NAND special feature.pdf". for more details.

Bad Block Handling Type = "Multiple Partition Uboot"

Spare Area = "Enabled" if the image contains the spare area data and that data should be programmed into the device. "Disabled" if the image does not contain the spare area data and only the main array should be programmed.

Required good block area: Start block = “0” This will require the entered block to be a valid block

Required good block area: Number of blocks = “0” This will be the total number of blocks required to be valid after the start block.

PartitionTable File = <Path to .mbn file> A binary file which contains the partition information for the device. This is the same as the Generic Multiple partition partition table file. A description of the file is included later.

ERASE AFTER PROGRAM = “Disabled” for normal use. “Enabled” this will cause the algorithm to perform an erase after the program cycle. This will set a device back to the virgin state with no bad block tables. Because the bad block tables overwrite the bad block marker they will be seen as bad with other algorithms and not be erased.

bad block detection = “BBM Specified” Since this scheme uses a subpaged solution, a custom bad block detection method must be used.

Check BB Marker In DataFile = “Disabled” Since this scheme uses a subpages solution, the FlashCore must be instructed to ignore overwriting the default bad block location.

All other features are not used for this scheme.

Image Preparation

The image must be prepared similarly to Generic Multiple Partition. There must be 0xFF padding data between each partition so that the beginning of the partition directly aligns with the block specified in the start address of the partition. The image data can be either in large format pages or small format pages. The bad block manager will detect the bad block marks regardless of format.

Format of PartitionTable.mbn:

a. Binary file fixed length 256 bytes.
b. Organization:16 rows x 4 columns. Each table item is 32-bits, little endian byte ordering.
c. Each row of the table describes configuration for one partition. Up to 16 partitions can be used.
d. Partition configuration:
    i. **Start Adr**: address of start of partition in flash blocks. The programmer will set the file read pointer and the programmer write pointer to Start Adr. If Start Adr=0xFFFFFFFF, skip to the next partition.
    ii. **End Adr**: last valid block in the current partition. The last data block programmed must be equal to or less than End Adr, otherwise the programmer will reject the flash device.
    iii. **Actual Data Length**: number of blocks of data to read from the input file and write to the flash in the current partition

Please note to keep: **Actual Data Length + max bad blocks allowed <= End Adr - Start Adr + 1**

    iv. Example PartitionTable.mbn file:

NAND Flash Block			
Start Adr	End Adr	Actual Data Length	Reserved
0x0	0x7FF	0x360	0xFFFFFFFF
0x800	0xFFF	0x30	0xFFFFFFFF
0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF
0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF
0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF
0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF
0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF
0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF
0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF
0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF
0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF
0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF
0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF
0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF
0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF
0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF

Revision History

- **V 1.0 – 1/6/2010**

Appendix

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