# Generic MPF NFC BBT ECC HLSWAP

This scheme is a Generic Multiple Partition format. It uses skip method for bad blocks handling within each partition. Bad blocks within any partition do not affect the starting location of the follow up partitions. The binary image is reorganized from large pages (4096/2048 bytes) to small (512+16 bytes) pages, over-writing the manufacturer Bad Block marker area. To preserve the original NAND's manufacturer bad block markers, a bad block table is created and 2 copies are saved within the last two (good) device blocks.

## **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: <u>http://www.dataio.com/support/dsr.asp</u>

## Bad Block Handling Type = "Generic MPF NFC BBT ECC HLSWAP"

## Spare Area = "ECC"

• This option should be selected for this BBM.

ECC Extended Type= "ECC iMX35 HLSWAP"

• This scheme adds 9 bytes of iMX35 ECC (RS-ECC) values per 512 bytes of data to the spare area during programming.

<u>BBT Enabled</u> = "Yes"

• This will add two bad block tables (Bbt0/1tbB) in the device's last two good blocks.

Convert large to small page format? = "Yes"

• This option will reorganize large pages (2048/4096 bytes) to small pages (512). <u>Check BB marker in dataFile</u> = "Disabled"

• This setting is required since the image file contains information that will write over the manufacturer's Bad Block marker area.

Bad block detection = "BBM then BB marker"

• When RE-programming previously programmed devices where the device's natural bad block marker is over-written, this option should be selected. This option will cause the BBM algorithm scan the last four blocks of the device searching for the bad block table Signature "Bbt0..." at the address 0x200 of the first pages. If a table is found, it will be used for the bad block listing. If the bad block table signature is not found, the standard NAND bad block marker area will be searched and a new table will be created.

**NOTE** – When devices that have BB-table previously programmed within them are mixed with "fresh" devices, the "fresh" devices will be safely rejected during programming and only pre-programmed devices will be re-programmed.

Partition Table File? = "Name and Location of the PartitionTable.Mbn"

• The format of the partition table is listed further below within this user guide.

All other features are not used for this scheme.

#### **User Checksum Implications**

In this scheme, the TaskLink checksum will be different from the device checksum because of the bad block table that contains possible bad blocks list.

#### **Image Preparation**

The image data file should combine the binary data from the different partitions and "FF" is to be padded into the unused areas. The created image file should not contain any spare (OOB) area data.

### Exceptions

• If any other bad Block management (or skip) is used with the devices that are preprogrammed with this BBM, they will fail because of too many bad blocks. For this reason, any device preprogrammed with this BBM should first be erased with this BBM before using any other BBM.

#### **Special Notes**

• To erase a device that is pre-programmed with this BBM, use a blank (FF) image file with this same BBM setup.

#### **Bad Block Table Format**

- There are two copies of the Bad Block tables created and saved per device.
- These tables consist of a header followed by a table of two bits per device block.
- The headers are saved at offset 0x200 of each of the last two blocks.
- The first table has a signature of "Bbt1..." at the last device good block.
- The second table has a signature of "1tbB..." at n-1)th device good block.
- There are two bits used per device block and the value of "11" indicates a good block within the table. These bad block table pages will have RS ECC added at run time.
- As examples of bad block entries in the tables
  - 1. Block # 18 is bad: The value of "00" is saved as bits # 2 & 3 of byte # 4 in the bad block table address 0x4.
  - 2. Block # 1 (counting from 0) is bad: "00" is saved as bits 2 & 3 of byte # 0 in the bad block table address 0x0.

### Partition Table Format Partition.mbn:

- a. A binary file of YourFile.MBN with fixed length of 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=0xFFFFFFF, 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.

**Note** - The last partition should be allocated for the Bbt tables with the following settings:

- It should start with device's block count -3, for example 4096-3.
- It should end with device's block count -1, like 4096-1.
- It should be minimum size of 4 (2 bad blocks and two as spare.)

#### **TaskLink Setup Example:**

Judie died	Edit Task			Choose Special Feature:	Edit Task	?   ×
ECC Extended Type Partition Table File	Task  Process  Data  File  I/O  Serialize  Sectors  Specia    B Marker In DataFile  Prattion Added  Process  Data  File  I/O  Serialize  Sectors  Specia    bled					The second se
Check BB Marker In DataFile BB-Table Partition Added					Choose Special Feature: Spare area	nalize Sectors Special realures
BBT Enabled						
Bad block handling type	Spare area Index Table					
Generic MPF NFC BBT ECC GBBM2.3	Partition Table F			ODT Enabled	ECC Extended Type Partition Table File	
GBBM2.3_EXPIA	Check BB Mark BB-Table Partiti	Task Process	Data File I/O Serializ	Yes	Check BB Marker In DataFile BB-Table Partition Added	
GBBM2_PIA_RFS121 Generic MPF NFC BBT ECC	BBT Enabled	Choose Special F	eature:	<u> </u>	BBT Enabled Convert large to small page format?	~
Genenc MPF NFC IMX25 BBT	Spare area	Bad block handli			Conver Edit Task	
Generic MPF NFC IMX27 BBT Generic MPF NFC_IMX ECC_E	ECC	Spare area			YES	<u></u>
Generic MPF with Multiple ECC Generic Multiple Partition	Disabled ECC	Partition Table File Check BB Marke	r In DataFile		Task Process Data	File I/O Serialize Sectors Special Features
	ECC Linux Enabled	BB-Table Partitio BBT Enabled	n Added		Choose Special Feature:	
	Update ECC fiel	ECC Extended Ty	pe		EFS: Device Start block	
		ECC IMX35	dit Task		EFS: Image Start block bad block detection	
		ECC DaVinci ( ECC IMX31	Task Process Data		BBM RB: Read Bad Bloc BBM RB: Move Read Blo	cks 🛁
		ECC IMX31 Isr			BBM RB: Read Chip Bloc BBM RB: Total Read Bloc	
		ECC IMX35 ECC IMX51	Choose Special Feature		bad block detection	
		ECC Linux	Bad block handling type Spare area		BBM then BB marker	-
	· ·	ECC Type 16 ECC Type 17	ECC Extended Type Partition Table File	Choose Special Feature: Bad block handling type	BBM opposition	If re-work device set this.
		ECC Type BTS	Check BB Marker In Da BB-Table Partition Adde	Spare area ECC Extended Type	BBM then BB marker semi vendor BB marker	else : default
			BBT Enabled	Partition Table File		
		/	Partition Table File	BB Table Partition Added BBT Enabled		
	11		G:\Customers\aVNET\i	Check BB Markir In DataFil		
PartitionTest 8Gbit.MBN	<u> </u>			Disabled		
0000: 00 00 00 00 0		)1 00 00 00	FF FF FF FF			
0020: <u>1C 00</u> 00 00 1	FF 00 00 00 J	07 00 00 00 31 00 00 00	FF FF FF FF FF			
0040: FF FF FF FF	FF FF FF FF	02 00 00 00 F FF FF FF	FF FF FF FF FF FF FF FF			
0060: FF FF FF FF I	FF FF FF FF	FF FF FF FF FF FF FF FF	FF FF FF FF			OK Cancel
0080: FF FF FF FF I	FF FF FF FF H	FF FF FF FF FF FF FF FF FF FF FF FF	FF FF FF FF FF			
		FF FF FF FF	FF FF FF FF			

#### This Document's versions-

V1.0: 11/12/2018 - Original version

#### Appendix

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