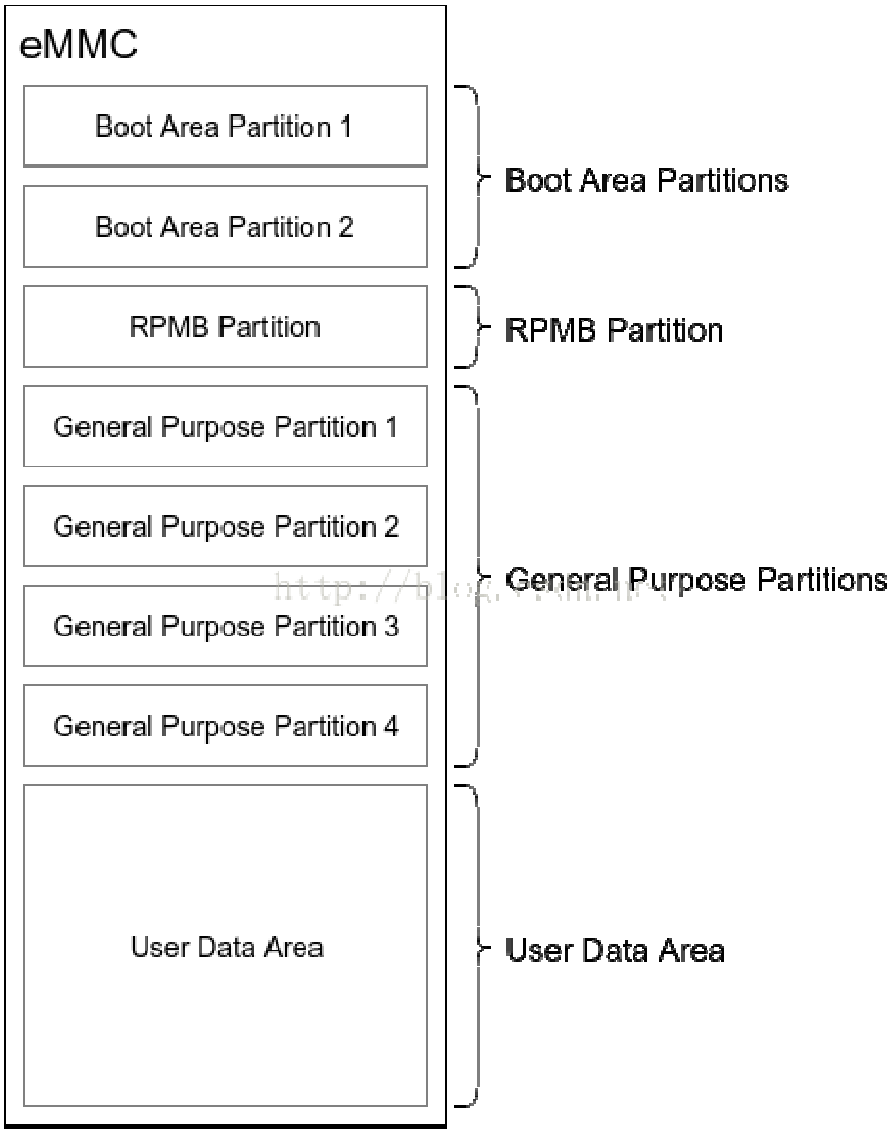
eMMC 分區管理

Partitions Overview

eMMC 標準中，將內部的 Flash Memory 劃分為 4 類區域，最多可以支持 8 個硬件分區，如下圖所示：



概述

一般情況下，Boot Area Partitions 和 RPMB Partition 的容量大小通常都為 4MB，部分芯片廠家也會提供配置的機會。General Purpose Partitions (GPP) 則在出廠時默認不被支持，即不存在這些分區，需要用戶主動使能，並配置其所要使用的 GPP 的容量大小，GPP 的數量可以為 1 - 4 個，各個 GPP 的容量大小可以不一樣。User Data Area (UDA) 的容量大小則為總容量大小減去其他分區所佔用的容量。更多各個分區的細節將在後續章節中描述。

分區編址

eMMC 的每一個硬件分區的存儲空間都是獨立編址的，即訪問地址為 0 - partition size。具體的數據讀寫操作實際訪問哪一個硬件分區，是由 eMMC 的 Extended CSD register 的 PARTITION_CONFIG Field 中 的 Bit[2:0]: PARTITION_ACCESS 決定的，用戶可以通過配置 PARTITION_ACCESS 來切換硬件分區的訪問。也就是說，用戶在訪問特定的分區前，需要先發送命令，配置 PARTITION_ACCESS，然後再發送相關的數據訪問請求。更多數據讀寫相關的細節，請參考 eMMC 總線協議 章節。

eMMC 的各個硬件分區有其自身的功能特性，多分區的設計，為不同的應用場景提供了便利。

Boot Area Partitions

Boot Area 包含兩個 Boot Area Partitions，主要用於存儲 Bootloader，支持 SOC 從 eMMC 啟動系統。

容量大小

兩個 Boot Area Partitions 的大小是完全一致的，由 Extended CSD register 的 BOOT_SIZE_MULT Field 決定，大小的計算公式如下：

Size = 128Kbytes x BOOT_SIZE_MULT

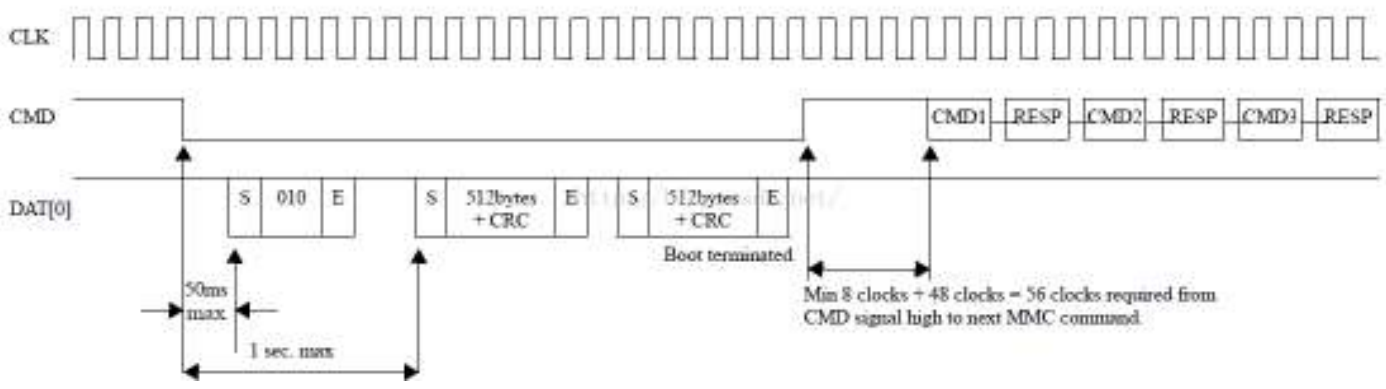
一般情況下，Boot Area Partition 的大小都為 4 MB，即 BOOT_SIZE_MULT 為 32，部分芯片廠家會提供改寫 BOOT_SIZE_MULT 的功能來改變 Boot Area Partition 的容量大小。BOOT_SIZE_MULT 最大可以為 255，即 Boot Area Partition 的最大容量大小可以為 255 x 128 KB = 32640 KB = 31.875 MB。

從 Boot Area 啟動

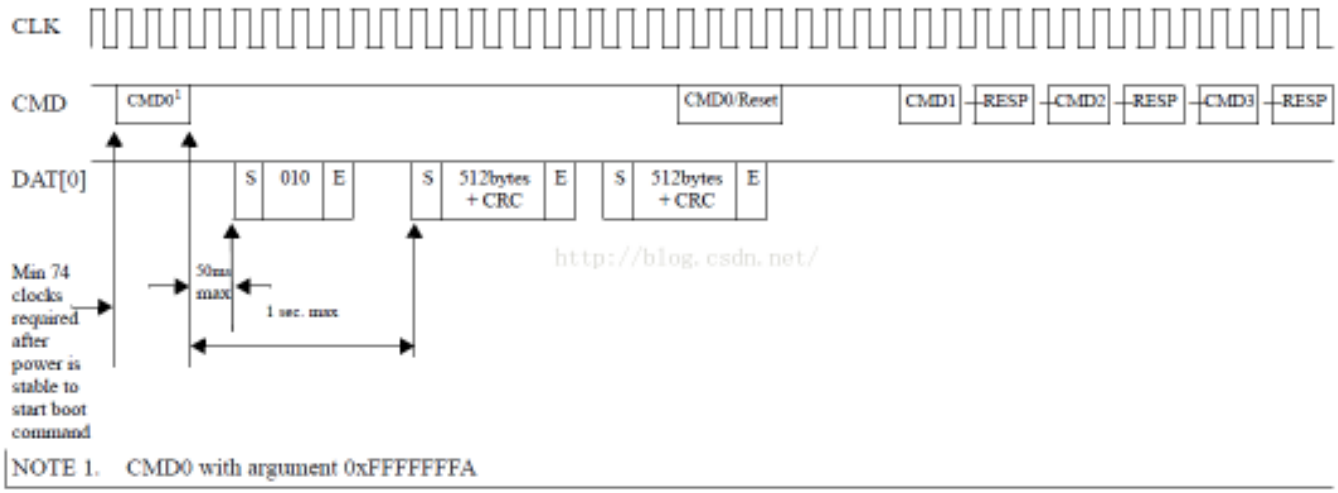
eMMC 中定義了 Boot State，在 Power-up、HW reset 或者 SW reset 後，如果滿足一定的條件，eMMC 就會進入該 State。進入 Boot State 的條件如下：

Original Boot Operation

CMD 信號保持低電平不少於 74 個時鐘週期，會觸發 Original Boot Operation，進入 Boot State。



Alternative Boot Operation
在 74 個時鐘週期後，在 CMD 信號首次拉低或者 Host 發送 CMD1 之前，Host 發送參數為 0xFFFFFFFA 的 COM0 時，會觸發 Alternative Boot Operation，進入 Boot State。



在 Boot State 下，如果有配置 BOOT_ACK，eMMC 會先發送 「010」 的 ACK 包，接著 eMMC 會將最大為 128Kbytes x BOOT_SIZE_MULT 的 Boot Data 發送給 Host。傳輸過程中，Host 可以通過拉高 CMD 信號 (Original Boot 中)，或者發送 Reset 命令 (Alternative Boot 中) 來中斷 eMMC 的數據發送，完成 Boot Data 傳輸。
Boot Data 根據 Extended CSD register 的 PARTITION_CONFIG Field 的 Bit[5:3]:BOOT_PARTITION_ENABLE 的設定，可以從 Boot Area Partition 1、Boot Area Partition 2 或者 User Data Area 讀出。
Boot Data 存儲在 Boot Area 比在 User Data Area 中要更加的安全，可以減少意外修改導致系統無法啟動，同時無法更新系統的情況出現。
（更多 Boot State 的細節，請參考 eMMC 總線協議 的 Boot State 章節）
寫保護

通過設定 Extended CSD register 的 BOOT_WP Field，可以為兩個 Boot Area Partition 獨立配置寫保護功能,以防止數據被意外改寫或者擦出。
eMMC 中定義了兩種 Boot Area 的寫保護模式：
Power-on write protection，使能後，如果 eMMC 掉電，寫保護功能失效，需要每次 Power on 後進行配置
Permanent write protection，使能後，即使掉電也不會失效，主動進行關閉才會失效

RPMB Partition

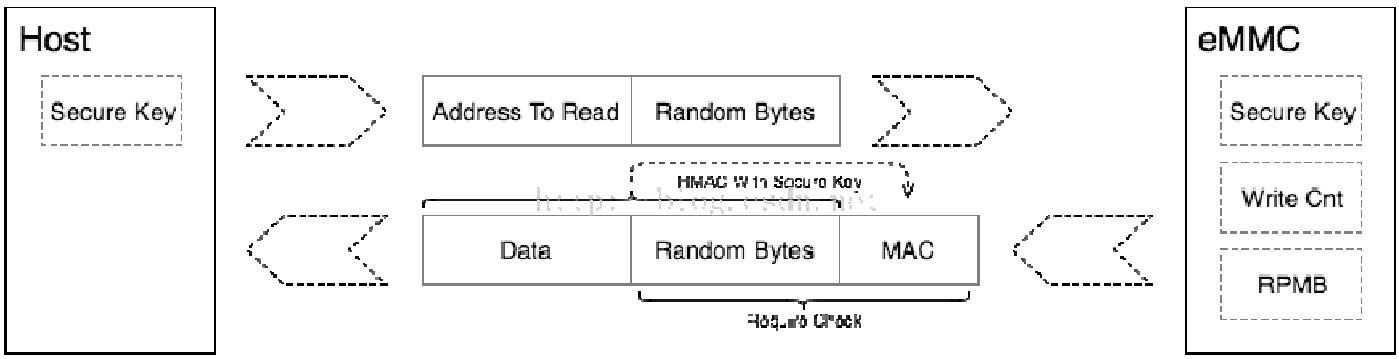
RPMB（Replay Protected Memory Block）Partition 是 eMMC 中的一個具有安全特性的

分區。
eMMC 在寫入數據到 RPMB 時，會校驗數據的合法性，只有指定的 Host 才能夠寫入，同時在讀數據時，也提供了簽名機制，保證 Host 讀取到的數據是 RPMB 內部數據，而不是攻擊者偽造的數據。
RPMB 在實際應用中，通常用於存儲一些有防止非法篡改需求的數據，例如手機上指紋支付相關的公鑰、序列號等。RPMB 可以對寫入操作進行鑑權，但是讀取並不需要鑑權，任何人都可以進行讀取的操作，因此存儲到 RPMB 的數據通常會進行加密後再存儲。
容量大小

兩個 RPMB Partition 的大小是由 Extended CSD register 的 BOOT_SIZE_MULT Field 決定，大小的計算公式如下：
Size = 128Kbytes x BOOT_SIZE_MULT
一般情況下，Boot Area Partition 的大小為 4 MB，即 RPMB_SIZE_MULT 為 32，部分芯片廠家會提供改寫 RPMB_SIZE_MULT 的功能來改變 RPMB Partition 的容量大小。RPMB_SIZE_MULT 最大可以為 128，即 Boot Area Partition 的最大容量大小可以為 128 x 128 KB = 16384 KB = 16 MB。
Replay Protect 原理

使用 eMMC 的產品，在產線生產時，會為每一個產品生產一個唯一的 256 bits 的 Secure Key，燒寫到 eMMC 的 OTP 區域（只能燒寫一次的區域），同時 Host 在安全區域中（例如：TEE）也會保留該 Secure Key。
在 eMMC 內部，還有一個 RPMB Write Counter。RPMB 每進行一次合法的寫入操作時，Write Counter 就會自動加一 。
通過 Secure Key 和 Write Counter 的應用，RMPB 可以實現數據讀取和寫入的 Replay Protect。
RPMB 數據讀取

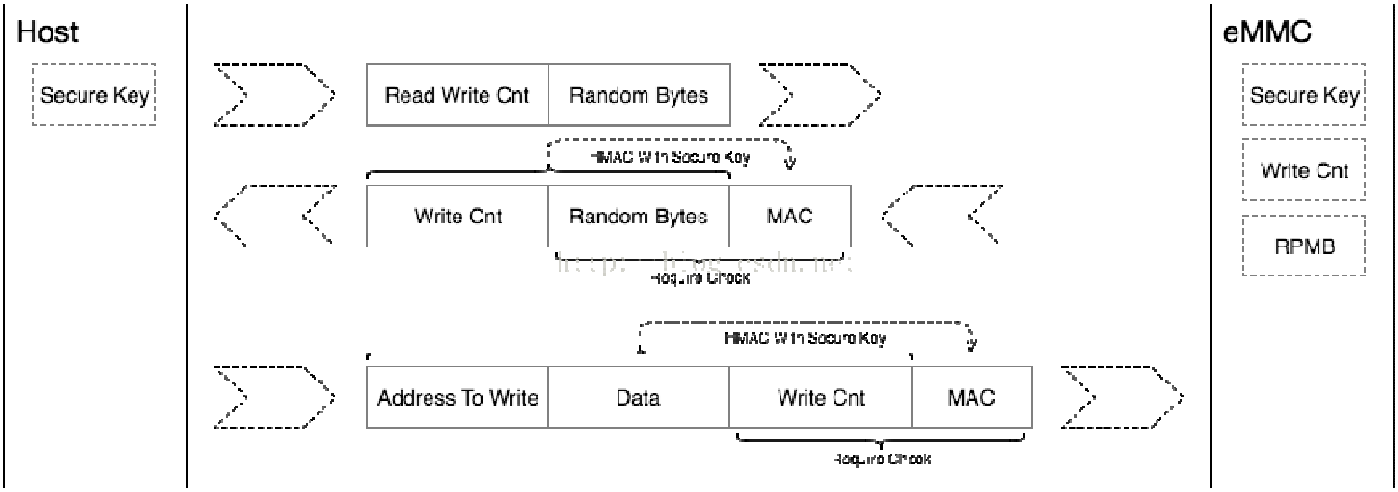
RPMB 數據讀取的流程如下：



Host 向 eMMC 發起讀 RPMB 的請求，同時生成一個 16 bytes 的隨機數，發送給

eMMC。
eMMC 將請求的數據從 RPMB 中讀出，並使用 Secure Key 通過 HMAC SHA-256 算法，計算讀取到的數據和接收到的隨機數拼接到一起後的簽名。然後，eMMC 將讀取到的數據、接收到的隨機數、計算得到的簽名一併發送給 Host。
Host 接收到 RPMB 的數據、隨機數以及簽名後，首先比較隨機數是否與自己發送的一致，如果一致，再用同樣的 Secure Key 通過 HMAC SHA-256 算法對數據和隨機數組合到一起進行簽名，如果簽名與 eMMC 發送的簽名是一致的，那麼就可以確定該數據是從 RPMB 中讀取到的正確數據，而不是攻擊者偽造的數據。
通過上述的讀取流程，可以保證 Host 正確的讀取到 RPMB 的數據。
RPMB 數據寫入

RPMB 數據寫入的流程如下：



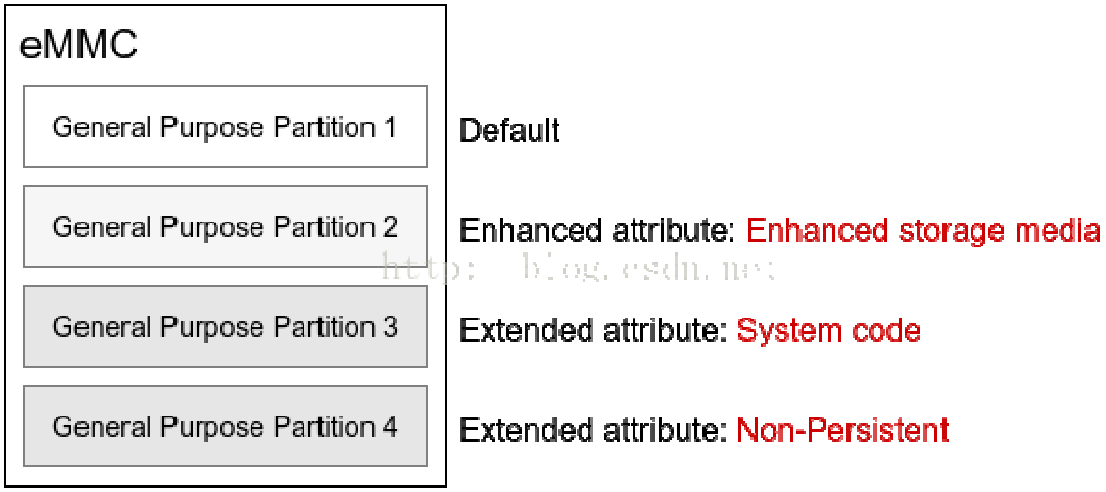
Host 按照上面的讀數據流程，讀取 RPMB 的 Write Counter。
Host 將需要寫入的數據和 Write Counter 拼接到一起並計算簽名，然後將數據、Write Counter 以及簽名一併發給 eMMC。
eMMC 接收到數據後，先對比 Write Counter 是否與當前的值相同，如果相同那麼再對數據和 Write Counter 的組合進行簽名，然後和 Host 發送過來的簽名進行比較，如果簽名相同則鑑權通過，將數據寫入到 RPMB 中。
通過上述的寫入流程，可以保證 RPMB 不會被非法篡改。
更多 RPMB 相關的細節，可以參考 eMMC RPMB 章節。

General Purpose Partitions

eMMC 提供了 General Purpose Partitions (GPP)，主要用於存儲系統和應用數據。在很多使用 eMMC 的產品中，GPP 都沒有被啟用，因為它在功能上與 UDA 類似，產品上直接使用 UDA 就可以滿足需求。
容量大小

eMMC 最多可以支持 4 個 GPPs，每一個 GPP 的大小可以單獨配置。用戶可以通過設定 Extended CSD register 的以下三個 Field 來設 GPPx (x=1~4) 的容量大小：
GP_SIZE_MULT_x_2
GP_SIZE_MULT_x_1
GP_SIZE_MULT_x_0
GPPx 的容量計算公式如下：
Size = (GP_SIZE_MULT_x_2 * 2^16 + GP_SIZE_MULT_x_1 * 2^8 +
GP_SIZE_MULT_x_0 * 2^0) * (Write protect group size)
Write protect group size = 512KB * HC_ERASE_GRP_SIZE * HC_WP_GRP_SIZE
eMMC 中，擦除和寫保護都是按塊進行的，上述表達式中的 HC_WP_GRP_SIZE 為寫保護的操作塊大小，HC_ERASE_GRP_SIZE 則為擦除操作的快的大小。
eMMC 芯片的 GPP 的配置通常是只能進行一次 (OTP)，一般會在產品量產階段，在產線上進行。
分區屬性

eMMC 標準中，為 GPP 定義了兩類屬性，Enhanced attribute 和 Extended attribute。每個 GPP 可以設定兩類屬性中的一種屬性，不可以同時設定多個屬性。



Enhanced attribute
Default, 未設定 Enhanced attribute。
Enhanced storage media， 設定 GPP 為 Enhanced storage media。
在 eMMC 標準中，實際上並未定義設定 Enhanced attribute 後對 eMMC 的影響。Enhanced attribute 的具體作用，由芯片製造商定義。
在實際的產品中，設定 Enhanced storage media 後，一般是把該分區的存儲介質從 MLC 改變為 SLC，提高該分區的讀寫性能、壽命以及穩定性。由於 1 個存儲單元下，MLC 的容量是 SLC 的兩倍，所以在總的存儲單元數量一定的情況下，如果把原本為 MLC 的分區改變為 SLC，會減少 eMMC 的容量，就是說，此時 eMMC 的實際總容量比標稱的總

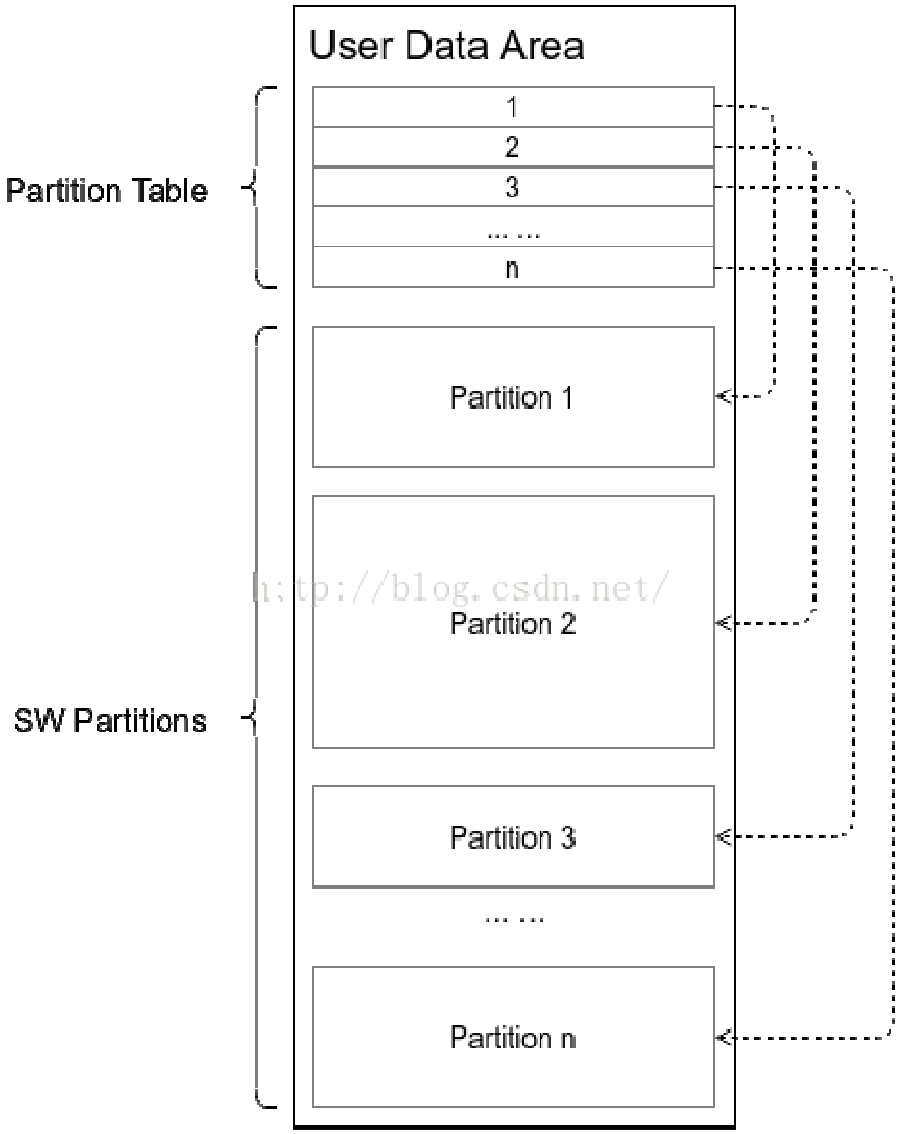
容量會小一點。（MLC 和 SLC 的細節可以參考 Flash Memory 章節內容）
Extended attribute
Default, 未設定 Extended attribute。
System code， 設定 GPP 為 System code 屬性，該屬性主要用在存放操作系統類的、很少進行擦寫更新的分區。
Non-Persistent，設定 GPP 為 Non-Persistent 屬性，該屬性主要用於存儲臨時數據的分區，例如 tmp 目錄所在分區、 swap 分區等。
在 eMMC 標準中，同樣也沒有定義設定 Extended attribute 後對 eMMC 的影響。
Extended attribute 的具體作用，由芯片製造商定義。Extended attribute 主要是跟分區的應用場景有關，廠商可以為不用應用場景的分區做不同的優化處理。

User Data Area

User Data Area (UDA) 通常是 eMMC 中最大的一個分區，是實際產品中，最主要的存儲區域。
容量大小

UDA 的容量大小不需要設置，在配置完其他分區大小後，再扣除設置 Enhanced attribute 所損耗的容量，剩下的容量就是 UDA 的容量。
軟件分區

為了更合理的管理數據，滿足不同的應用需求，UDA 在實際產品中，會進行軟件再分區。目前主流的軟件分區技術有 MBR（Master Boot Record）和 GPT（GUID Partition Table）兩種。這兩種分區技術的基本原理類似，如下圖所示：



軟件分區技術一般是將存儲介質劃分為多個區域，既 SW Partitions，然後通過一個 Partition Table 來維護這些 SW Partitions。在 Partition Table 中，每一個條目都保存著一個 SW Partition 的起始地址、大小等的屬性信息。軟件系統在啟動後，會去掃瞄 Partition Table，獲取存儲介質上的各個 SW Partitions 信息，然後根據這些信息，將各個 Partitions 加載到系統中，進行數據存取。
MBR 和 GPT 此處不展開詳細介紹，更多的細節可以參考 wikipedia 上 MBR 和 GPT 相關介紹。
區域屬性

eMMC 標準中，支持為 UDA 中一個特定大小的區域設定 Enhanced attribute。與 GPP 中的 Enhanced attribute 相同，eMMC 標準也沒有定義該區域設定 Enhanced attribute 後對 eMMC 的影響。Enhanced attribute 的具體作用，由芯片製造商定義。
Enhanced attribute
Default, 未設定 Enhanced attribute。

Enhanced storage media， 設定該區域為 Enhanced storage media。
在實際的產品中，UDA 區域設定為 Enhanced storage media 後，一般是把該區域的存儲介質從 MLC 改變為 SLC。通常，產品中可以將某一個 SW Partition 設定為 Enhanced storage media，以獲得更好的性能和健壯性。

eMMC 分區應用實例

在一個 Android 手機系統中，各個分區的呈現形式如下：
mmcblk0 為 eMMC 的塊設備;
mmcblk0boot0 和 mmcblk0boot1 對應兩個 Boot Area Partitions;
mmcblk0rpmb 則為 RPMB Partition，
mmcblk0px 為 UDA 劃分出來的 SW Partitions;
如果存在 GPP，名稱則為 mmcblk0gp1、mmcblk0gp2、mmcblk0gp3、mmcblk0gp4;

```
root@xxx:/ # ls /dev/block/mmcblk0*
/dev/block/mmcblk0
/dev/block/mmcblk0boot0
/dev/block/mmcblk0boot1
/dev/block/mmcblk0rpmb
/dev/block/mmcblk0p1
/dev/block/mmcblk0p2
/dev/block/mmcblk0p3
/dev/block/mmcblk0p4
/dev/block/mmcblk0p5
/dev/block/mmcblk0p6
/dev/block/mmcblk0p7
/dev/block/mmcblk0p8
/dev/block/mmcblk0p9
/dev/block/mmcblk0p10
/dev/block/mmcblk0p11
/dev/block/mmcblk0p12
/dev/block/mmcblk0p13
/dev/block/mmcblk0p14
/dev/block/mmcblk0p15
/dev/block/mmcblk0p16
/dev/block/mmcblk0p17
/dev/block/mmcblk0p18
/dev/block/mmcblk0p19
/dev/block/mmcblk0p20
/dev/block/mmcblk0p21
```

/dev/block/mmcblk0p22
/dev/block/mmcblk0p23
/dev/block/mmcblk0p24
/dev/block/mmcblk0p25
/dev/block/mmcblk0p26
/dev/block/mmcblk0p27
/dev/block/mmcblk0p28
/dev/block/mmcblk0p29
/dev/block/mmcblk0p30
/dev/block/mmcblk0p31
/dev/block/mmcblk0p32
每一個分區會根據實際的功能來設定名稱。
root@xxx:/ # ls -l /dev/block/platform/mtk-msdc.0/11230000.msdc0/by-name/
lrwxrwxrwx root root 2015-01-03 04:03 boot -> /dev/block/mmcblk0p22
lrwxrwxrwx root root 2015-01-03 04:03 cache -> /dev/block/mmcblk0p30
lrwxrwxrwx root root 2015-01-03 04:03 custom -> /dev/block/mmcblk0p3
lrwxrwxrwx root root 2015-01-03 04:03 devinfo -> /dev/block/mmcblk0p28
lrwxrwxrwx root root 2015-01-03 04:03 expdb -> /dev/block/mmcblk0p4
lrwxrwxrwx root root 2015-01-03 04:03 flashinfo -> /dev/block/mmcblk0p32
lrwxrwxrwx root root 2015-01-03 04:03 frp -> /dev/block/mmcblk0p5
lrwxrwxrwx root root 2015-01-03 04:03 keystore -> /dev/block/mmcblk0p27
lrwxrwxrwx root root 2015-01-03 04:03 lk -> /dev/block/mmcblk0p20
lrwxrwxrwx root root 2015-01-03 04:03 lk2 -> /dev/block/mmcblk0p21
lrwxrwxrwx root root 2015-01-03 04:03 logo -> /dev/block/mmcblk0p23
lrwxrwxrwx root root 2015-01-03 04:03 md1arm7 -> /dev/block/mmcblk0p17
lrwxrwxrwx root root 2015-01-03 04:03 md1dsp -> /dev/block/mmcblk0p16
lrwxrwxrwx root root 2015-01-03 04:03 md1img -> /dev/block/mmcblk0p15
lrwxrwxrwx root root 2015-01-03 04:03 md3img -> /dev/block/mmcblk0p18
lrwxrwxrwx root root 2015-01-03 04:03 metadata -> /dev/block/mmcblk0p8
lrwxrwxrwx root root 2015-01-03 04:03 nvdata -> /dev/block/mmcblk0p7
lrwxrwxrwx root root 2015-01-03 04:03 nvram -> /dev/block/mmcblk0p19
lrwxrwxrwx root root 2015-01-03 04:03 oemkeystore -> /dev/block/mmcblk0p12
lrwxrwxrwx root root 2015-01-03 04:03 para -> /dev/block/mmcblk0p2
lrwxrwxrwx root root 2015-01-03 04:03 ppl -> /dev/block/mmcblk0p6
lrwxrwxrwx root root 2015-01-03 04:03 proinfo -> /dev/block/mmcblk0p13
lrwxrwxrwx root root 2015-01-03 04:03 protect1 -> /dev/block/mmcblk0p9
lrwxrwxrwx root root 2015-01-03 04:03 protect2 -> /dev/block/mmcblk0p10
lrwxrwxrwx root root 2015-01-03 04:03 recovery -> /dev/block/mmcblk0p1
lrwxrwxrwx root root 2015-01-03 04:03 rstinfo -> /dev/block/mmcblk0p14

lrwxrwxrwx root root 2015-01-03 04:03 seccfg -> /dev/block/mmcblk0p11
lrwxrwxrwx root root 2015-01-03 04:03 secro -> /dev/block/mmcblk0p26
lrwxrwxrwx root root 2015-01-03 04:03 system -> /dev/block/mmcblk0p29
lrwxrwxrwx root root 2015-01-03 04:03 tee1 -> /dev/block/mmcblk0p24
lrwxrwxrwx root root 2015-01-03 04:03 tee2 -> /dev/block/mmcblk0p25
lrwxrwxrwx root root 2015-01-03 04:03 userdata -> /dev/block/mmcblk0p31

參考資料

Embedded Multi-Media Card (e•MMC) Electrical Standard (5.1) [PDF]
Disk partitioning [Web]
Master Boot Record [Web]
GUID Partition Table [Web]