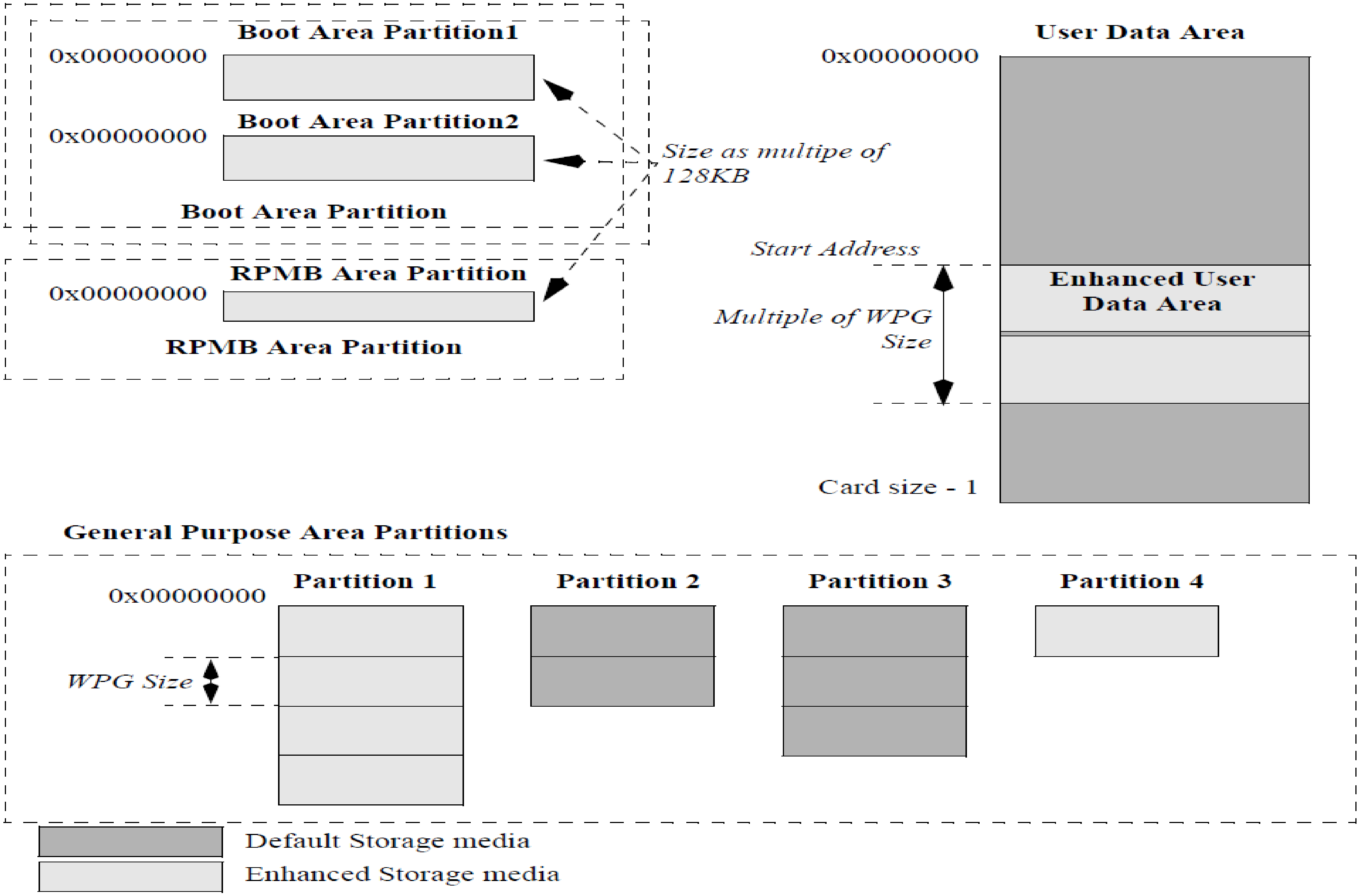
eMMC RPMB, Enhance, GP and use protection

Update On 6/19/2023

Biyong Sun

eMMC Layout



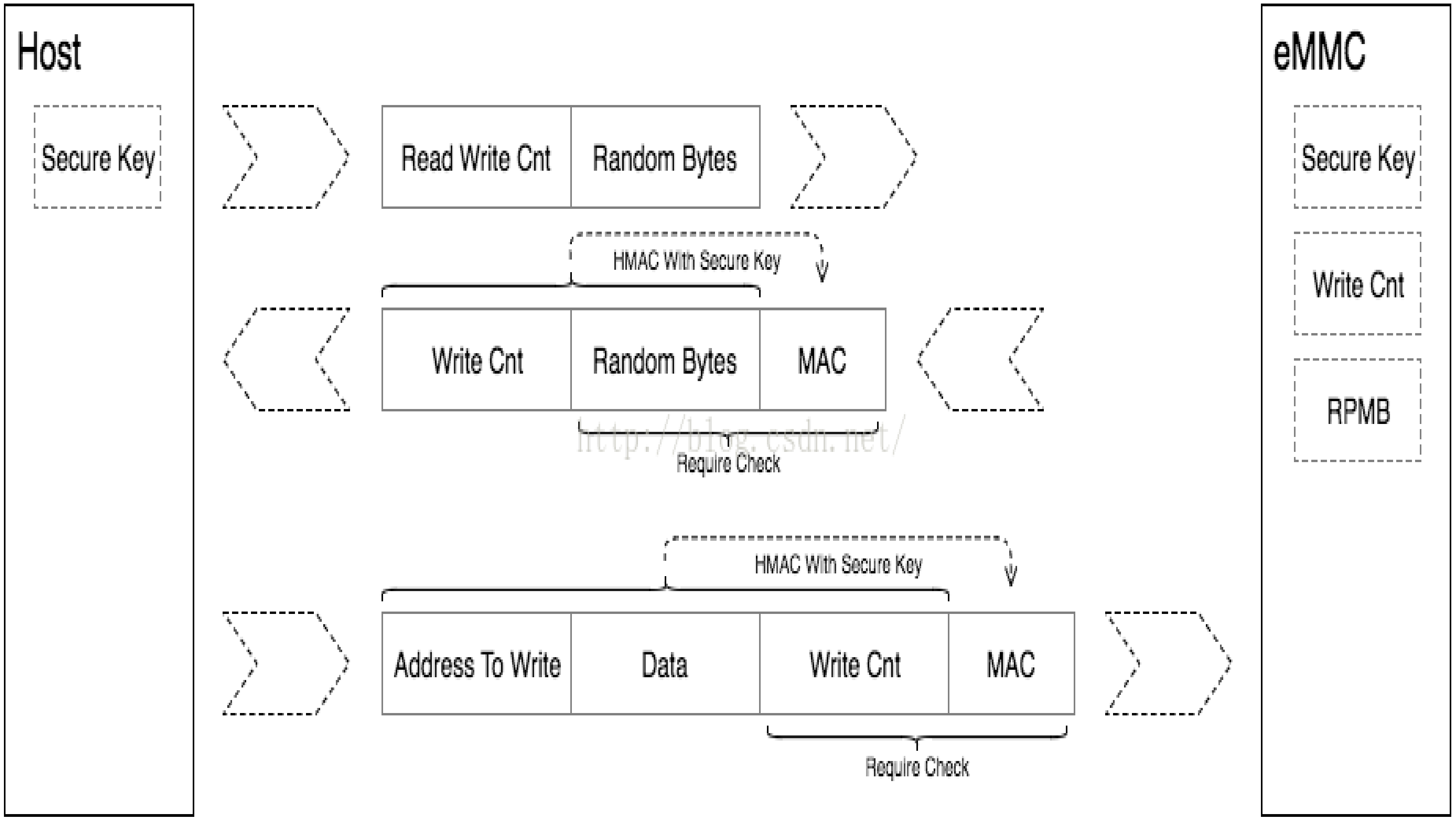
Boot Area Partition1
0x00000000
Boot Area Partition2
0x00000000
Boot Area Partition
RPMB Area Partition
0x00000000
RPMB Area Partition
Size as multipe of 128KB
User Data Area
0x00000000
Start Address
Enhanced User Data Area
Multiple of WPG Size
Card size - 1
General Purpose Area Partitions
0x00000000
Partition 1
Partition 2
Partition 3
Partition 4
WPG Size
Default Storage media
Enhanced Storage media

Test Environment

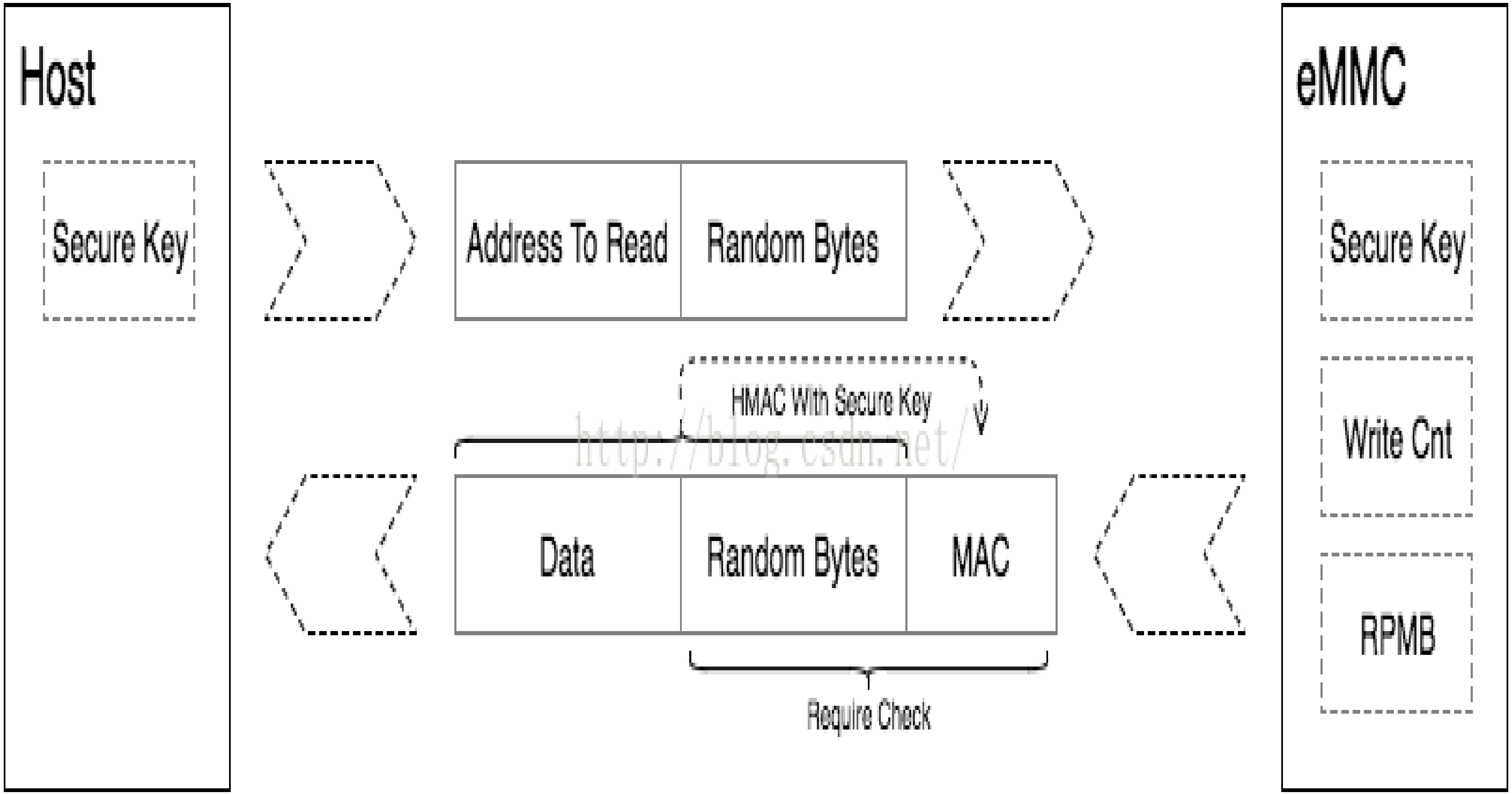
HW: i.MX6Q SDB
SW: L4.1.15_2.0.0

eMMC RPMB (Replay Protected Memory Block)

RPMB write


Host
Secure Key
Read Write Cnt
Random Bytes
HMAC With Secure Key
Write Cnt
Random Bytes
MAC
Require Check
HMAC With Secure Key
Address To Write
Data
Write Cnt
MAC
Require Check
eMMC
Secure Key
Write Cnt
RPMB

RPMB read


Host
Secure Key
Address To Read
Random Bytes
eMMC
Secure Key
Write Cnt
RPMB
HMAC With Secure Key
Data
Random Bytes
MAC
Require Check

RPMB related commands in mmc-utils

```
mmc rpmb write-key <rpmb device> <key file>

mmc rpmb write-block <rpmb device> <address> <256 byte data file> <key file>
mmc rpmb read-block <rpmb device> <address> <blocks count> <output file> [key file]
```

Please check the mmc help to get more details

Use the RPMB

- Set key (OTP)
- Write with key and wrong key to RPMB
- Read with key and wrong key from RPMB

Set key (OTP)

```
echo 'Authkeymustbe32byteslength_0000' > keyfile.txt

mmc rpmb write-key /dev/mmcblk3rpmb keyfile.txt
```

NOTE! This is a one-time programmable (unreversible) change

Needs power cycle

Write with key and wrong key to RPMB

```
echo
'256bytedatafile.256bytedatafile.256bytedatafile.256bytedatafile
.256bytedatafile.256bytedatafile.256bytedatafile.256bytedatafile
.256bytedatafile.256bytedatafile.256bytedatafile.256bytedatafile
.256bytedatafile.256bytedatafile.256bytedatafile.256bytedatafile
.' > data.txt

echo 'Authkeymustbe32byteslength_0000' > keyfile.txt
mmc rpmb write-key /dev/mmcblk3rpmb keyfile.txt

echo 'Authkeymustbe32byteslength_1111' > Wrongkeyfile.txt
mmc rpmb write-block  /dev/mmcblk3rpmb 0  data.txt
Wrongkeyfile.txt
RPMB operation failed, retcode 0x0002
```

Read with key and wrong key from RPMB

```
mmc rpmb read-block /dev/mmcblk3rpmb 0 1 out.txt
mmc rpmb read-block /dev/mmcblk3rpmb 0 1 out.txt keyfile.txt
cat out.txt
256bytedatafile.256bytedatafile.256bytedatafile.256bytedatafile.256bytedatafile.256bytedatafile.256bytedatafile.256bytedatafile.256bytedatafile.256bytedatafile.256bytedatafile.256bytedatafile.256bytedatafile.256bytedatafile.256bytedatafile.256bytedatafile.
```

```
mmc rpmb read-block /dev/mmcblk3rpmb 0 1 out.txt  Wrongkeyfile.txt
RPMB MAC mismatch
```

With Key/MAC (Message Authentication Code), will make sure the data are authenticated. Not fake data hacked or from attack.

Enhanced User Data Area (pseudoSLC Mode)

What is Enhanced User Data Area

- Simply to say pseudo SLC
- Make the area more reliable
- Capacity will be smaller after enable (MLC to SLC)
- A side effect of pSLC mode can be improved write speed

Enable the enhanced user area

- Check MAX_ENH_SIZE_MULT

```
mmc extcsd read /dev/mmcblk3 | grep MAX_ENH_SIZE_MULT -A 1
Max Enhanced Area Size [MAX_ENH_SIZE_MULT]: 0x0000ea
 i.e. 3833856 KiB
```

- Enable all area to enhanced

command:

```
mmc enh_area set <-y|-n|-c> <start KiB> <length KiB> <device>
```

```
mmc enh_area set -n 0 3833856 /dev/mmcblk3
```

Please use -n to check before you really use -y to do it

```
mmc enh_area set -y 0 3833856 /dev/mmcblk3
Enhanced User Data Area Size [ENH_SIZE_MULT]: 0x0000ea
 i.e. 3833856 KiB
 Max Enhanced Area Size [MAX_ENH_SIZE_MULT]: 0x0000ea
 i.e. 3833856 KiB
```

NOTE! This is a one-time programmable (unreversible) change
Needs power cycle

GP
(General Purpose Partition)

GP(General Purpose Partition)

- Up to 4 GPs could be created as physical partition
- GP could have Enhanced attribute

Create GPs

- Command
  ```
  mmc gp create <-y|-n|-c> <length KiB> <partition> <enh_attr> <ext_attr> <device>
  ```
- Check MAX_ENH_SIZE_MULT
  ```
  mmc extcsd read /dev/mmcblk3 | grep MAX_ENH_SIZE_MULT -A 1
  Max Enhanced Area Size [MAX_ENH_SIZE_MULT]: 0x0000ea
   i.e. 3833856 KiB
  ```
- Create two GPs

Create gp2

```
mmc gp create -n  93888 2  1 0   /dev/mmcblk3
Enhanced GP1 Partition Size [GP_SIZE_MULT_1]: 0x00000b
 i.e. 90112 KiB
Max Enhanced Area Size [MAX_ENH_SIZE_MULT]: 0x0000ea
 i.e. 3833856 KiB
```

Note: Please use -n, just check and set the eMMC register, if it is not the last gp to create

Create gp1

```
mmc gp create -y   524288  1  1 0 /dev/mmcblk3
Enhanced GP1 Partition Size [GP_SIZE_MULT_1]: 0x000040
 i.e. 524288 KiB
Max Enhanced Area Size [MAX_ENH_SIZE_MULT]: 0x0000ea
 i.e. 3833856 KiB
```

NOTE! This is a one-time programmable (unreversible) change
Needs power cycle

Use GP

```
ls /dev/mmcblk3*
mmcblk3       mmcblk3boot0  mmcblk3boot1  mmcblk3gp0   mmcblk3gp1
mmcblk3rpmb

fdisk -l
Disk /dev/mmcblk3: 6 GiB, 6476005376 bytes, 12648448 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk /dev/mmcblk3gp1: 88 MiB, 92274688 bytes, 180224 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes


Disk /dev/mmcblk3gp0: 512 MiB, 536870912 bytes, 1048576 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

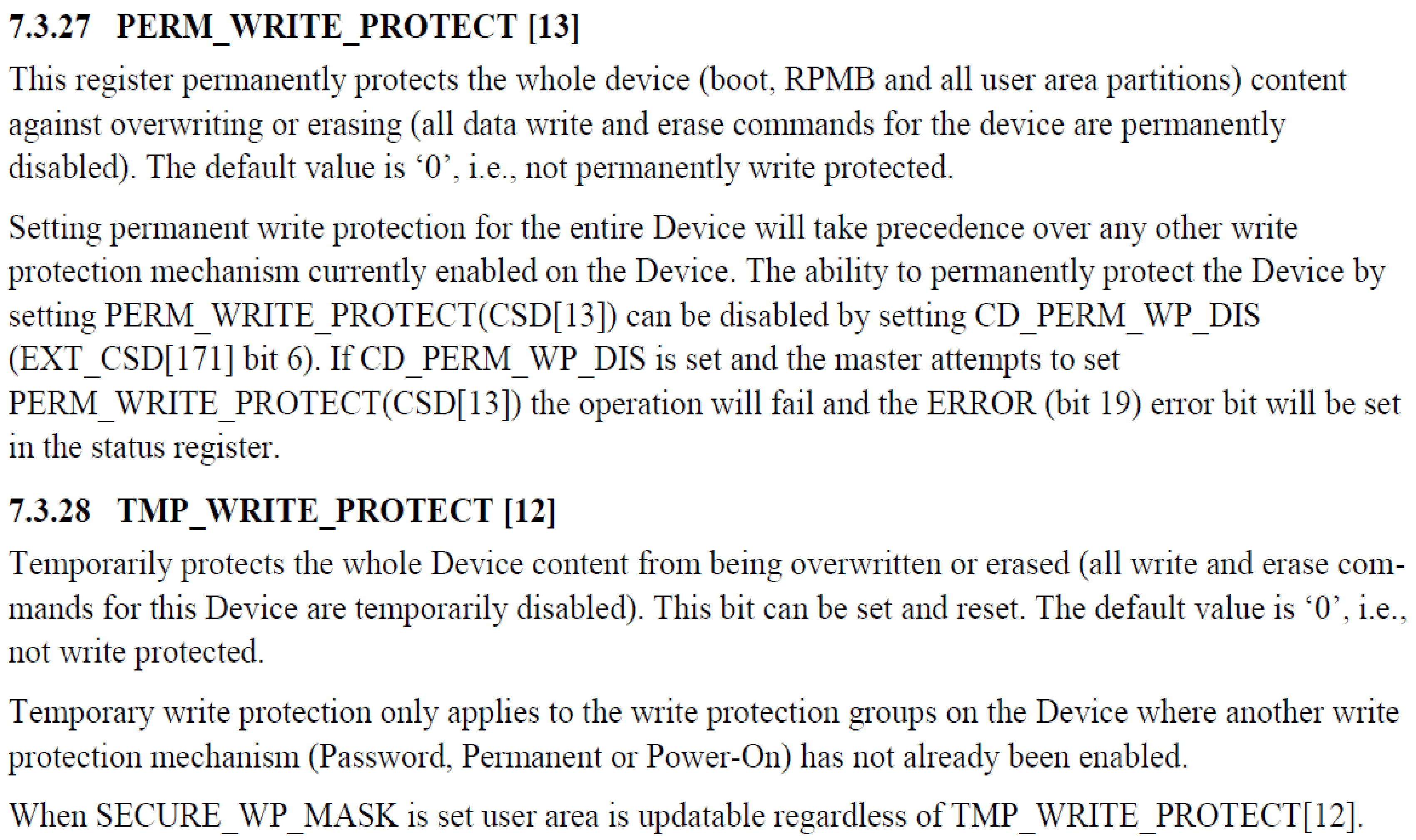
Note: eMMC spends about 2G to get the enhanced 88MiB + 512MiB = 600MiB
The total volume to about 6.6G from about 8G.

Use GP(Cont.)

```
fdisk /dev/mmcblk3gp0


Command (m for help): p
Disk /dev/mmcblk3gp0: 512 MiB, 536870912 bytes, 1048576 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xd0d48a7b

Device            Boot  Start     End Sectors  Size Id Type
/dev/mmcblk3gp0p1        2048  264191  262144  128M  c W95 FAT32 (LBA)
/dev/mmcblk3gp0p2      264192 1048575  784384  383M 83 Linux

mkfs.vfat -F 32   /dev/mmcblk3gp0p1
mkfs.ext3         /dev/mmcblk3gp0p2
```

eMMC user partition write protect

eMMC JEDEC SPEC



7.3.27 PERM_WRITE_PROTECT [13]

This register permanently protects the whole device (boot, RPMB and all user area partitions) content against overwriting or erasing (all data write and erase commands for the device are permanently disabled). The default value is ‘0’, i.e., not permanently write protected.

Setting permanent write protection for the entire Device will take precedence over any other write protection mechanism currently enabled on the Device. The ability to permanently protect the Device by setting PERM_WRITE_PROTECT(CSD[13]) can be disabled by setting CD_PERM_WP_DIS (EXT_CSD[171] bit 6). If CD_PERM_WP_DIS is set and the master attempts to set PERM_WRITE_PROTECT(CSD[13]) the operation will fail and the ERROR (bit 19) error bit will be set in the status register.

7.3.28 TMP_WRITE_PROTECT [12]

Temporarily protects the whole Device content from being overwritten or erased (all write and erase commands for this Device are temporarily disabled). This bit can be set and reset. The default value is ‘0’, i.e., not write protected.

Temporary write protection only applies to the write protection groups on the Device where another write protection mechanism (Password, Permanent or Power-On) has not already been enabled.

When SECURE_WP_MASK is set user area is updatable regardless of TMP_WRITE_PROTECT[12].

mmc tool

```
w/o DANGEROUS_COMMANDS_ENABLED (default compile)
mmc-utils
make
./mmc

    mmc writeprotect user set <type><start block><blocks><device>
        Set the write protect configuration for the specified region
        of the user area for <device>.
        <type> must be "none|temp|pwron".
          "none"  - Clear temporary write protection.
          "temp"  - Set temporary write protection.
          "pwron" - Set write protection until the next poweron.
        <start block> specifies the first block of the protected area.
        <blocks> specifies the size of the protected area in blocks.
        NOTE! The area must start and end on Write Protect Group
        boundries, Use the "writeprotect user get" command to get the
        Write Protect Group size.
```

mmc tool(Cont.)

w/ DANGEROUS_COMMANDS_ENABLED

mmc-utils

```
CFLAGS=-DDANGEROUS_COMMANDS_ENABLED make
    mmc writeprotect user set <type><start block><blocks><device>
        Set the write protect configuration for the specified region
        of the user area for <device>.
        <type> must be "none|temp|pwron|perm".
            "none"  - Clear temporary write protection.
            "temp"  - Set temporary write protection.
            "pwron" - Set write protection until the next poweron.
            "perm"  - Set permanent write protection.
        <start block> specifies the first block of the protected area.
        <blocks> specifies the size of the protected area in blocks.
        NOTE! The area must start and end on Write Protect Group
        boundries, Use the "writeprotect user get" command to get the
        Write Protect Group size.
        NOTE! "perm" is a one-time programmable (unreversible) change.
```

“temp” is enough.

Do NOT suggest to use “perm”, it is not necessary in most of cases.

mmc to protect user partition

Here is partition table on eMMC

```
device          Boot      Start      End         Sectors      Size      Id Type
/dev/mmcblk2p1 *        16384    186775     170392       83.2M    c   W95 FAT32 (LBA)
/dev/mmcblk2p2        196608  10876069 10679462  5.1G     83 Linux
```

```
mmc writeprotect user get /dev/mmcblk2

Write Protect Group size in blocks/bytes: 16384/8388608
Write Protect Groups 0-3726 (Blocks 0-61063167), No Write Protection

mmc writeprotect user set temp  16384 180224    /dev/mmcblk2
#Because the protect group size in blocks is 16384. The first multiple of 16384 to cover
#186775 is 180224(16384*11).
#If you want to use emmc protection, you'd better to align the size of partition by "Write
#Protect Group size".

mmc writeprotect user get /dev/mmcblk2
Write Protect Group size in blocks/bytes: 16384/8388608
Write Protect Groups 0-0 (Blocks 0-16383), No Write Protection
Write Protect Groups 1-11 (Blocks 16384-196607), Temporary Write Protection
Write Protect Groups 12-3726 (Blocks 196608-61063167), No Write Protection
```

mmc to protect user partition(cont.)

Using dd to verify the protection

```
dd if=/dev/zero of=/run/media/mmcblk2p1/test.bin bs=1M count=1 conv=fsync
[   222.734765] blk_update_request: I/O error, dev mmcblk2, sector 80072 op 0x1:(WRITE) flags 0x4800 phys_seg 128 prio class 0
[   222.740308] blk_update_request: I/O error, dev mmcblk2, sector 81096 op 0x1:(WRITE) flags 0x800 phys_seg 128 prio class 0
dd: fsync failed for '/run/media/mmcblk2p1/test.bin': Input/output error
1+0 records in
1+0 records out
1048576 bytes (1.0 MB, 1.0 MiB) copied, 0.139139 s, 7.5 MB/s
```

Turn back to the writable

```
mmc writeprotect user set none  16384 180224    /dev/mmcblk2
```

```
dd if=/dev/zero of=/run/media/mmcblk2p1/test.bin bs=1M count=1 conv=fsync
1+0 records in
1+0 records out
1048576 bytes (1.0 MB, 1.0 MiB) copied, 0.0237912 s, 44.1 MB/s
```

Miscellaneous

Device Reliability

```
mmc write_reliability set <-y|-n|-c> <partition> <device>
        Enable write reliability per partition for the <device>.
        Dry-run only unless -y or -c is passed.
        Use -c if more partitioning settings are still to come.
```

NOTE! This is a one-time programmable (unreversible) change.

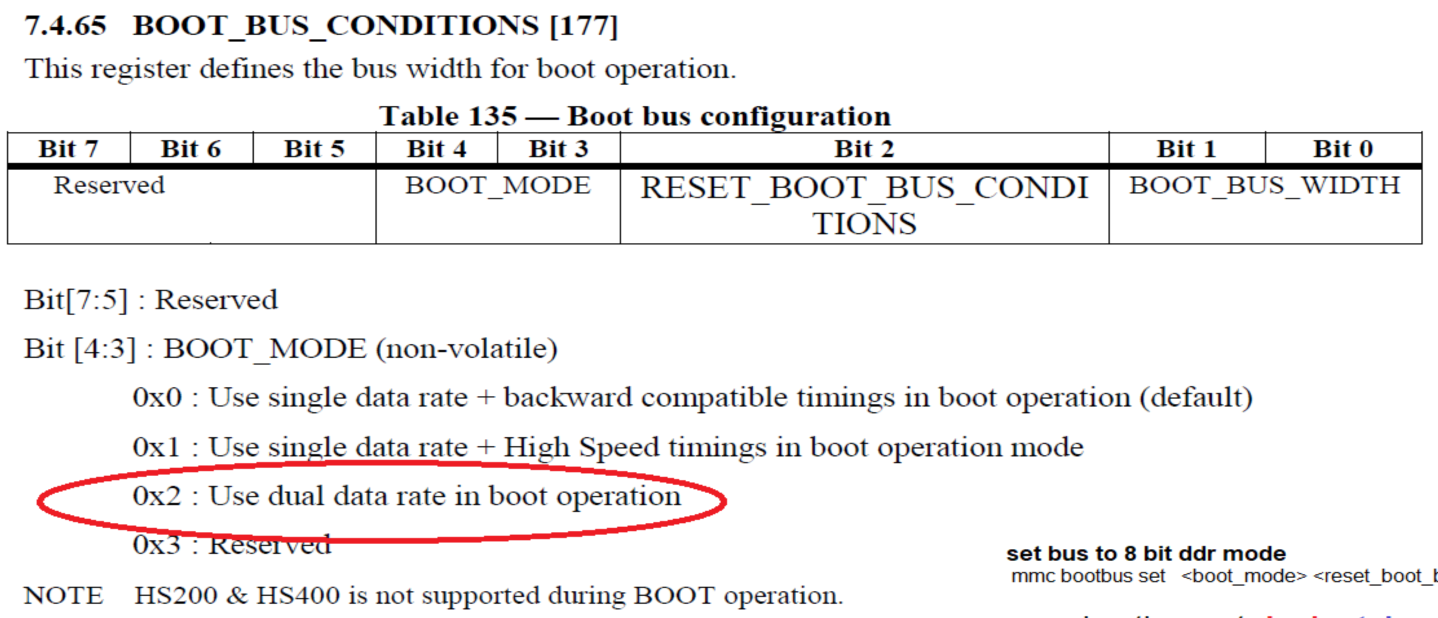
Set fast boot

```
 mmc bootbus set <boot_mode> <reset_boot_bus_conditions>
<boot_bus_width> <device>
		Set Boot Bus Conditions.
		<boot_mode> must be "single_backward|single_hs|dual"
		<reset_boot_bus_conditions> must be "x1|retain"
		<boot_bus_width> must be "x1|x4|x8“
```

set bus to 8 bit ddr mode

```
mmc bootbus set dual retain x8 /dev/mmcblk3
```

Set fast boot (Cont.)



7.4.65 BOOT_BUS_CONDITIONS [177]

This register defines the bus width for boot operation.

Table 135 — Boot bus configuration

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved			BOOT_MODE		RESET_BOOT_BUS_CONDITIONS	BOOT_BUS_WIDTH	

Bit[7:5] : Reserved

Bit [4:3] : BOOT_MODE (non-volatile)

0x0 : Use single data rate + backward compatible timings in boot operation (default)

0x1 : Use single data rate + High Speed timings in boot operation mode

[✓ 0x2 : Use dual data rate in boot operation]

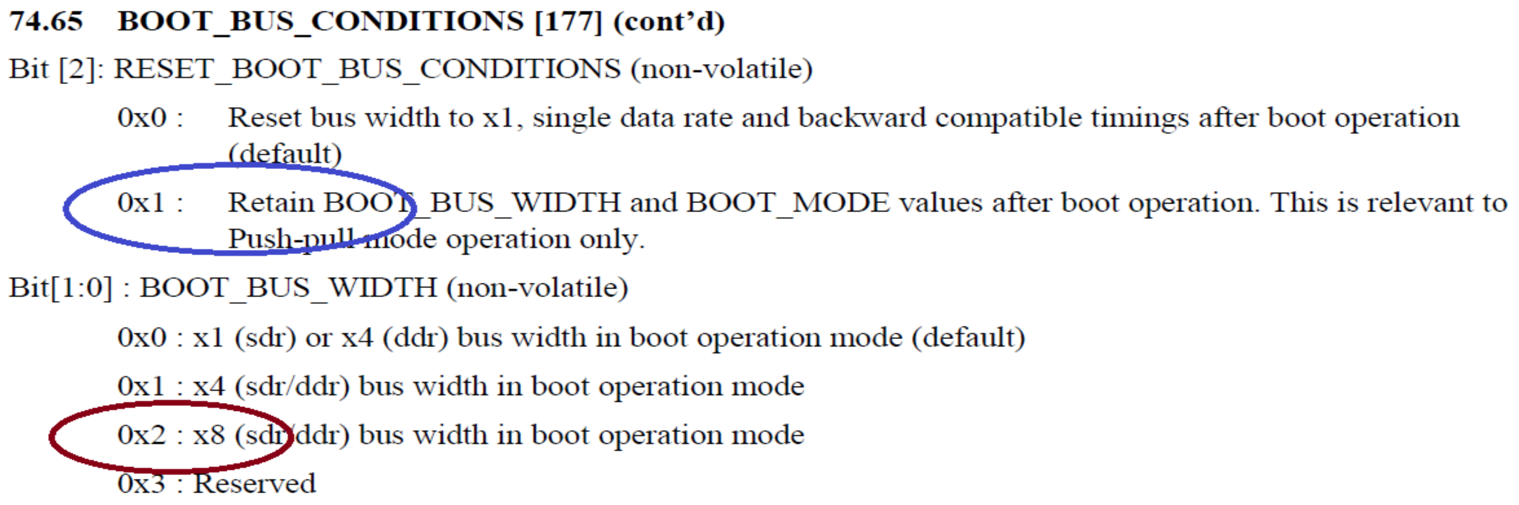
0x3 : Reserved

NOTE HS200 & HS400 is not supported during BOOT operation.

set bus to 8 bit ddr mode

```
mmc bootbus set  <boot_mode> <reset_boot_bus_conditions> <boot_bus_width> <device>
```

```
mmc bootbus set dual retain x8 /dev/mmcblk3
```



74.65 BOOT_BUS_CONDITIONS [177] (cont'd)

Bit [2]: RESET_BOOT_BUS_CONDITIONS (non-volatile)

0x0 : Reset bus width to x1, single data rate and backward compatible timings after boot operation (default)

[✓ 0x1 :] Retain BOOT_BUS_WIDTH and BOOT_MODE values after boot operation. This is relevant to Push-pull mode operation only.

Bit[1:0] : BOOT_BUS_WIDTH (non-volatile)

0x0 : x1 (sdr) or x4 (ddr) bus width in boot operation mode (default)

0x1 : x4 (sdr/ddr) bus width in boot operation mode

[✓ 0x2 : x8 (sdr/ddr)] bus width in boot operation mode

0x3 : Reserved