MA35D1 EMMC/SD card 增加分區

NuMicro® 64 位系列微控制器範例代碼介紹

文件資訊

應用簡述	本範例程式碼介紹 MA35D1 的 buildroot 平臺，SD card/EMMC 新增主要磁碟分割、擴展分區及邏輯分區的方法。
BSP 版本	Buildroot_2024
開發平臺	NuMaker-HMI-MA35D1-S1 V1.0

The information described in this document is the exclusive intellectual property of Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.

Nuvoton is providing this document only for reference purposes of NuMicro microcontroller and microprocessor based system design. Nuvoton assumes no responsibility for errors or omissions.

All data and specifications are subject to change without notice.

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

1. 概述

MA35D1 的 SD 卡及 EMMC 目前軟體不支援 GPT（GUID Partition Table）分區啟動，僅使用 MBR（Master Boot Record）分區模式，這種分區模式最多支援 4 個主要磁碟分割。如果想要更多的分區，就需要在主要磁碟分割外，再分一個擴展分區，然後在這個擴展分區上再分多個邏輯分區，進而突破 MBR 的 4 分區限制。

1.1 原理

MBR（主引導記錄）分區結構主要由三部分組成，446 位元組引導程式，64 位元組分區表及 2 位元組的結束標記，這 512 位元組通常在 EMMC 的第一個磁區，其核心作用是檢查分區表完整性，定位使用中的磁碟分割等。如表：

偏移	描述	長度（位元組）
0	程式碼區	440（最大 446）
440	選用磁片標誌	4
444	一般為空值，0x0000	2
446	MBR 分區表（四個 16 位元組的主要磁碟分割表入口）	64
511	MBR 結束標誌：0x55AA	2

表 1-1 MBR 結構

其中 MBR 分區表（偏移 446 到 550 位元組）占 64 位元組，共 4 個表項，每個表項各位元組定義如下表：

偏移	長度	意義
00H	1	分區狀態：00->非使用中的磁碟分割；80->使用中的磁碟分割；其它數值沒意義

01H	1	分區起始磁頭號（HEAD），用到全部 8 位元
02H	2	分區起始磁區號（SECTOR），佔據 02H 的位 0~5；該分區的起始磁柱號（CYLINDER）,佔據 02H 的位 6~7 和 03H 的全部 8 位元
04H	1	檔案系統標誌，0x83：Linux ext2/ext3，0x0C：FAT32（LBA）
05H	1	分區結束磁頭號（HEAD），用到全部 8 位元
06H	2	分區結束磁區號（SECTOR），佔據 06H 的位 0~5；該分區的結束磁柱號（CYLINDER），佔據 06H 的位 6~7 和 07H 的全部 8 位元
08H	4	分區起始相對磁區號，用於定位分區在磁片中的位置
0CH	4	分區的磁區總數，用於計算分區大小

表 1-2 MBR 分區表結構

MBR 分區模式，最多支援 4 個主要磁碟分割。如果想要更多的分區，就需要在主要磁碟分割外，再分一個擴展分區，然後在這個擴展分區上可以分多個邏輯分區。

擴展分區表在 MBR 中和主要磁碟分割一樣的格式，可以理解為一種特殊的主要磁碟分割，每個 MBR 僅允許存在一個擴展分區，必須在其中建立邏輯分區後才能使用，擴展分區內的邏輯分區資訊通過 EBR（Extended Boot Record）鏈表進行管理。

第一個邏輯分區的 EBR 位於擴展分區的頭部，其餘每個邏輯分區的起始磁區前的 64 KB 位置包含一個 EBR，占 512 位元組，EBR 的前 16 位元組，描述當前邏輯分區的起始位置和大小，後 16 個位元組指向下一個邏輯分區的 EBR，形成鏈式結構。

在 MA35D1 上，預設的分區只有 MBR 管理一個主要磁碟分割（Primary1），用以儲存 rootfs，整片 EMMC 儲存程式分佈如圖：



圖 1-1 EMMC 預設分區及程式儲存

本文將在此基礎上，增加三個主要磁碟分割、一個擴展分區，再在擴展分區上分 3 個邏輯分區，以此為例分區如下图：

硬碟 (Disk)

├── MBR ：記錄了 主要磁碟分割 1、主要磁碟分割 2、主要磁碟分割 3 和 擴展分區 的位置

　　├── 主要磁碟分割 1 (對應 /dev/sda1)

　　├── 主要磁碟分割 2 (對應 /dev/sda2)

　　├── 主要磁碟分割 3 (對應 /dev/sda3)

　　└── 擴展分區 (對應 /dev/sda4，不實際儲存資料，分區第 0 磁區儲存 EBR1)

　　　　├── 邏輯分區 1 (EBR 1 管理，對應 /dev/sda5)

　　　　├── 邏輯分區 2 (EBR 2 管理，對應 /dev/sda6，分區 6 起始位址減 64 KB 的位置儲存 EBR2)

　　　　└── 邏輯分區 3 (EBR 3 管理，對應 /dev/sda7，分區7 起始位址減64 KB的位置儲存EBR3)

圖 1-2 EMMC 示例分區結構

增加分區後的 EMMC 儲存程式分佈如圖 1-3：

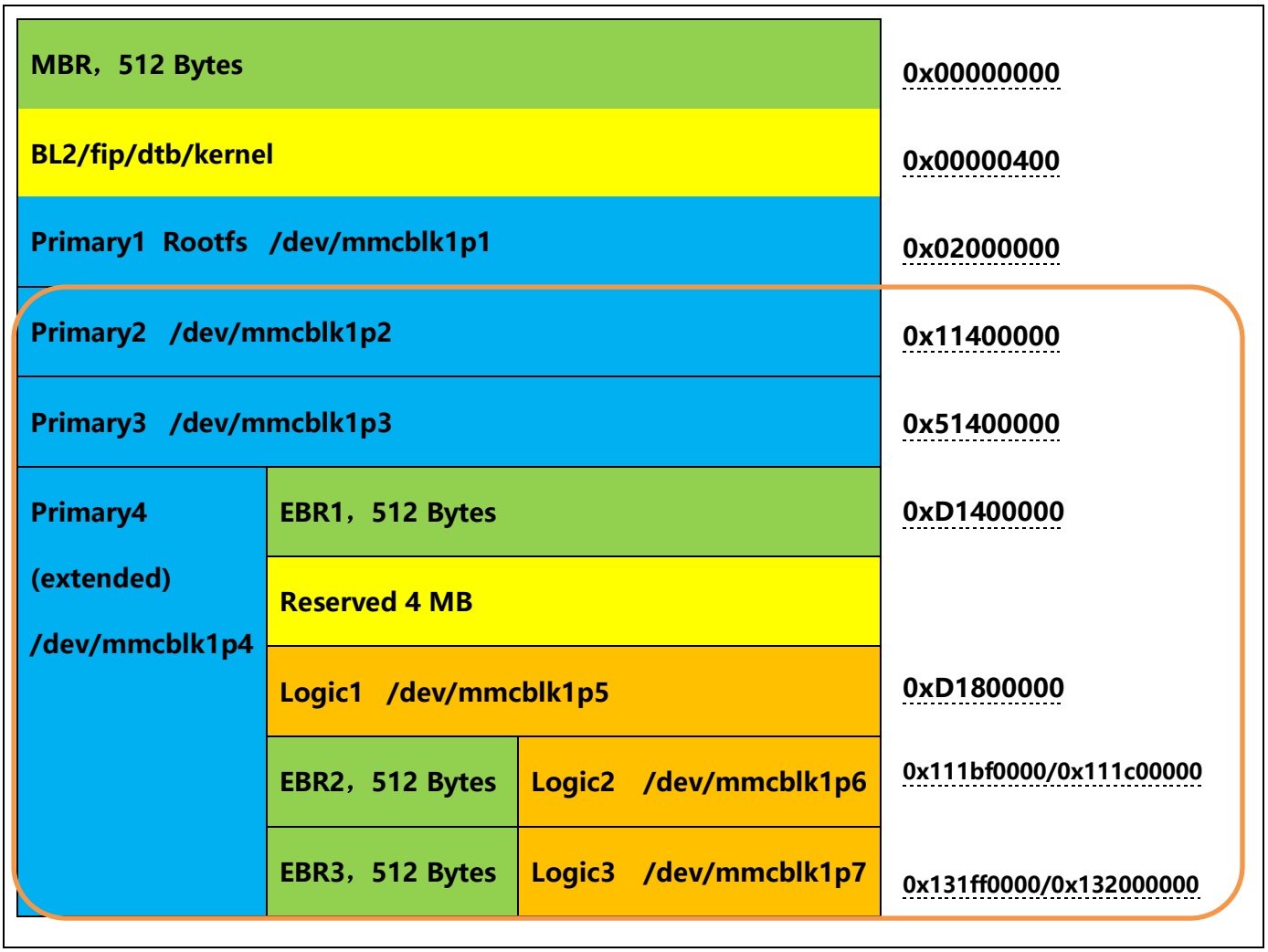


圖 1-3 增加分區後 EMMC 程式分佈圖

圖 1-3 中，MBR、BL2/fip/dtb/kernel、rootfs、EBR1、EBR2、EBR3 均需要打包至 pack 燒錄，MBR 管理三個主要磁碟分割及一個擴展分區，EBR 管理三個邏輯分區，需要注意的是 EBR 的位置，EBR1 在擴展分區 offset 為 0 的位置，EBR2 和 EBR3 在管理邏輯分區起始位址減 64 KB 的位置，均占 512 位元組。

1.2 執行結果

將新增了新分區表的 pack 映像檔燒錄至 EMMC 中，查看分區表，執行結果如下圖，圖中分區 mmcblk1p2~7 即為新加的分區。

```
# fdisk -l
Disk /dev/mmcblk1: 15 GB, 15634268160 bytes, 30535680 sectors
238560 cylinders, 4 heads, 32 sectors/track
Units: sectors of 1 * 512 = 512 bytes

Device       Boot StartCHS    EndCHS        StartLBA     EndLBA    Sectors  Size Id Type
/dev/mmcblk1p1    512,0,1     1023,3,32        65536     565247     499712  244M 83 Linux
/dev/mmcblk1p2    1023,3,32   1023,3,32       565248    2662399    2097152 1024M 83 Linux
/dev/mmcblk1p3    1023,3,32   1023,3,32      2662400    6856703    4194304 2048M 83 Linux
/dev/mmcblk1p4    1023,3,32   1023,3,32      6856704   23633919   16777216 8192M  f Win95 Ext'd (LBA)
/dev/mmcblk1p5    1023,3,32   1023,3,32      6864896    8962047    2097152 1024M 83 Linux
/dev/mmcblk1p6    1023,3,32   1023,3,32      8970240   10018815    1048576  512M 83 Linux
/dev/mmcblk1p7    1023,3,32   1023,3,32     10027008   12124159    2097152 1024M 83 Linux
```

圖 1-4 增加分區後 fdisk 指令執行結果

```
# parted /dev/mmcblk1 print
Model: MMC MMC16G (sd/mmc)
Disk /dev/mmcblk1: 15.6GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type      File system  Flags
 1      33.6MB  289MB   256MB   primary   ext4
 2      289MB   1363MB  1074MB  primary   ext4
 3      1363MB  3511MB  2147MB  primary   ext4
 4      3511MB  12.1GB  8590MB  extended               lba
 5      3515MB  4589MB  1074MB  logical   ext4
 6      4593MB  5130MB  537MB   logical   ext4
 7      5134MB  6208MB  1074MB  logical   ext4
```

圖 1-5 增加分區後 parted 指令執行結果

2. 程式碼介紹

本節內容將完整介紹如何生成帶擴展分區及邏輯分區表的 pack 映像檔，分區方案以图 1-3 為準。涉及到的修改檔包：

- *board/nuvoton/ma35d1/post-image.sh*、
- *board/nuvoton/ma35d1/nuwriter/pack-sdcard.json*

1. 修改 post-image.sh，確定擴展分區及各邏輯分區的起始位址，分區大小等資訊，以 MB 為基本單位，確認大小，不能越界到其它分區。

```
IMAGE_CMD_sdcard()
{
    EXT2_SIZE=$(sed -n -e 's/^BR2_TARGET_ROOTFS_EXT2_SIZE=/ /p' ${BR2_CONFIG} | sed 's/M/
/g' | sed 's/\"/ /g')

    BOOT_SPACE_ALIGNED=$BOOT_SPACE #32MB
    SDCARD_SIZE=$((16*1024*1024*1024)) #EMMC 16G
 ################################################################################
    MMC_BLK1_START=$BOOT_SPACE  #0x2000000
    MMC_BLK1_END=$(($BOOT_SPACE_ALIGNED+$IMAGE_ROOTFS_ALIGNMENT+$EXT2_SIZE))#32+4+240M

    MMC_BLK2_START=$MMC_BLK1_END #0x11400000
    MMC_BLK2_SIZE=1024 #1G
    MMC_BLK2_END=$(($MMC_BLK2_START+$MMC_BLK2_SIZE))

    MMC_BLK3_START=$MMC_BLK2_END #0x51400000
    MMC_BLK3_SIZE=2048 #2G
    MMC_BLK3_END=$(($MMC_BLK3_START+$MMC_BLK3_SIZE))

    MMC_EXTENED_START=$MMC_BLK3_END #0xd1400000
    MMC_EXTENED_SIZE=8192 #8G
    MMC_EXTENED_END=$(($MMC_EXTENED_START+$MMC_EXTENED_SIZE))

    MMC_LOGIC_RESERVE_SIZE=4 #4M
    MMC_LOGIC1_START=$(($MMC_EXTENED_START+$MMC_LOGIC_RESERVE_SIZE)) #0xd1800000
    MMC_LOGIC1_SIZE=1024 #1G
    MMC_LOGIC1_END=$(($MMC_LOGIC1_START+$MMC_LOGIC1_SIZE))

    MMC_LOGIC2_START=$(($MMC_LOGIC1_END+$MMC_LOGIC_RESERVE_SIZE)) #0x111c00000
    MMC_LOGIC2_SIZE=512 #512M
    MMC_LOGIC2_END=$(($MMC_LOGIC2_START+$MMC_LOGIC2_SIZE))

    MMC_LOGIC3_START=$(($MMC_LOGIC2_END+$MMC_LOGIC_RESERVE_SIZE)) #0x132000000
    MMC_LOGIC3_SIZE=1024 #1G
    MMC_LOGIC3_END=$(($MMC_LOGIC3_START+$MMC_LOGIC3_SIZE))
################################################################################
    if grep -Eq "^BR2_TARGET_MA35D1_SECURE_BOOT=y$" ${BR2_CONFIG}; then
     SDCARD=${BINARIES_DIR}/${IMGDEPLOYDIR}/${IMAGE_BASENAME}-${MACHINE}-
enc.rootfs.sdcard
    fi
```

2. 使用 dd 指令創建一個完整的空映像檔，大小為待分區 SD 卡/EMMC 大小，本例為 16 G。再使用 parted 指令對以上映像檔根據图 1-3 的方案完成分區，包括主要磁碟分割、擴展分區，及多個邏輯分區。

```
  # Initialize a sparse file
  dd if=/dev/zero of=${SDCARD} bs=1 count=0 seek=$SDCARD_SIZE &>${NULLDEV}
  ${HOST_DIR}/bin/fakeroot -u ${HOST_DIR}/sbin/parted ${SDCARD} -s mklabel msdos
################################################################################

  #PRIMARY1
  ${HOST_DIR}/bin/fakeroot -u ${HOST_DIR}/sbin/parted ${SDCARD} -s unit MiB \
mkpart primary $MMC_BLK1_START $MMC_BLK1_END

  #PRIMARY2
  ${HOST_DIR}/bin/fakeroot -u ${HOST_DIR}/sbin/parted ${SDCARD} -s unit MiB \
mkpart primary $MMC_BLK2_START $MMC_BLK2_END

  #PRIMARY3
  ${HOST_DIR}/bin/fakeroot -u ${HOST_DIR}/sbin/parted ${SDCARD} -s unit MiB \
mkpart primary $MMC_BLK3_START $MMC_BLK3_END

  #EXTENDED
  ${HOST_DIR}/bin/fakeroot -u ${HOST_DIR}/sbin/parted ${SDCARD} -s unit MiB \
mkpart extended $MMC_EXTENED_START $MMC_EXTENED_END

  #LOGIC1
  ${HOST_DIR}/bin/fakeroot -u ${HOST_DIR}/sbin/parted ${SDCARD} -s unit MiB \
mkpart logical $MMC_LOGIC1_START $MMC_LOGIC1_END

  #LOGIC2
  ${HOST_DIR}/bin/fakeroot -u ${HOST_DIR}/sbin/parted ${SDCARD} -s unit MiB \
mkpart logical $MMC_LOGIC2_START $MMC_LOGIC2_END

  #LOGIC3
  ${HOST_DIR}/bin/fakeroot -u ${HOST_DIR}/sbin/parted ${SDCARD} -s unit MiB \
mkpart logical $MMC_LOGIC3_START $MMC_LOGIC3_END

# print the result
  ${HOST_DIR}/bin/fakeroot -u ${HOST_DIR}/sbin/parted ${SDCARD} print
```

3. 使用dd指令從已分區完成的映像檔中讀取主要磁碟分割，擴展分區及邏輯分區分區表，讀取位置參考图 1-3，注意 MBR 在 EMMC 最前面 offset 為 0 的位置，EBR1 在擴展分區 offset 為 0 的位置，EBR2 和 EBR3 在邏輯分區起始位址減 64 KB 的位置，均占 512 位元組，最終生成 4 個分區表檔。

```
# MBR
dd if=${SDCARD} of=${BINARIES_DIR}/MBR.sdcard.bin \ conv=notrunc,fsync seek=0
count=512 bs=1 &>${NULLDEV}

#EBR1
```

```
dd if=${SDCARD} of=${BINARIES_DIR}/EBR1.sdcard.bin \ conv=notrunc,fsync
skip=$((1024*1024*$MMC_EXTENED_START)) seek=0 \ count=512 bs=1 &>${NULLDEV}

#EBR2
dd if=${SDCARD} of=${BINARIES_DIR}/EBR2.sdcard.bin \ conv=notrunc,fsync
skip=$((1024*1024*$MMC_LOGIC2_START-65536)) \ seek=0 count=512 bs=1 &>${NULLDEV}

#EBR3
dd if=${SDCARD} of=${BINARIES_DIR}/EBR3.sdcard.bin \ conv=notrunc,fsync
skip=$((1024*1024*$MMC_LOGIC3_START-65536)) \ seek=0 count=512 bs=1 &>${NULLDEV}

printf "EBR1 addr: %x\n" $((1024*1024*$MMC_EXTENED_START))
printf "EBR2 addr: %x\n" $((1024*1024*$MMC_LOGIC2_START-65536))
printf "EBR3 addr: %x\n" $((1024*1024*$MMC_LOGIC3_START-65536))
```

4. 在 pack-sdcard.json 腳本新增上面新生成的邏輯分區的分區表檔，offset 欄位按圖 1-3 規劃，即上面讀取的位置填寫。

```
{
  "offset": "0x2400000",
  "file": "rootfs.ext4",
  "type": 0
},
{
  "offset": "0xD1400000",
  "file": "EBR1.sdcard.bin",
  "type": 0
},
{
  "offset": "0x111bf0000",
  "file": "EBR2.sdcard.bin",
  "type": 0
},
{
  "offset": "0x131ff0000",
  "file": "EBR3.sdcard.bin",
  "type": 0
}
```

5. NuWriter 会使用最新的 json 腳本，壓縮生成帶有分區表檔的 pack 映像檔。

```
#pack all file in json
${HOST_DIR}/bin/nuwriter.py -p ${NUWRITER_TARGET}/pack-sdcard.json;
cp pack/pack.bin pack-${IMAGE_BASENAME}-${MACHINE}-sdcard.bin;
```

3. 軟體與硬體需求

3.1 軟體需求

- BSP 版本

  - Buildroot_2024

- NuWriter 版本

  - NuWriter GUI v1.09

3.2 硬體需求

- 電路元件

  - NuMaker-HMI-MA35D1-S1 V1.0

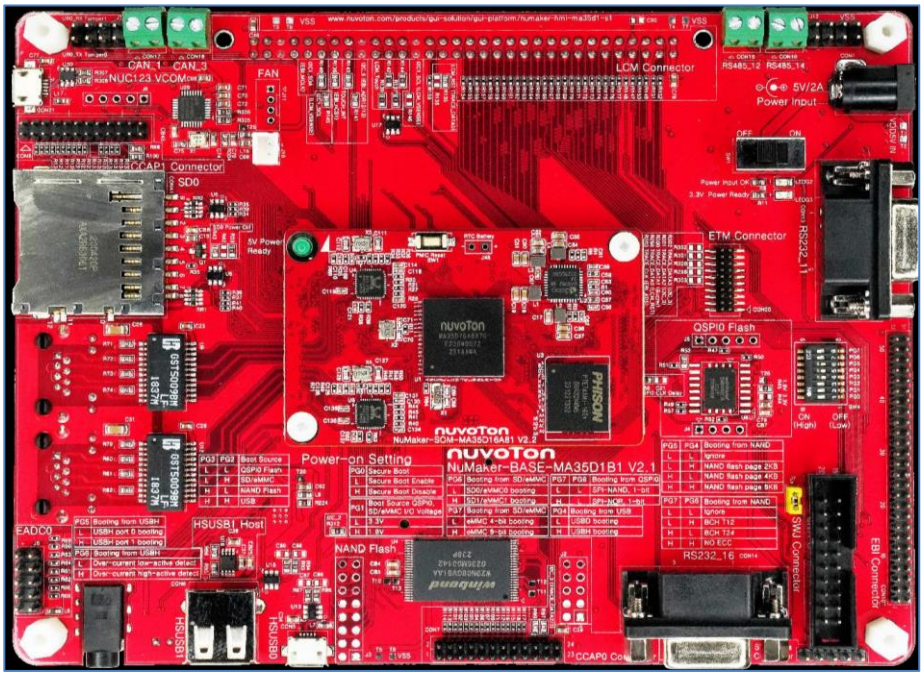


圖 3-1 NuMaker-HMI-MA35D1-S1 V1.0

4. 目錄資訊

目錄結構如下圖所示：

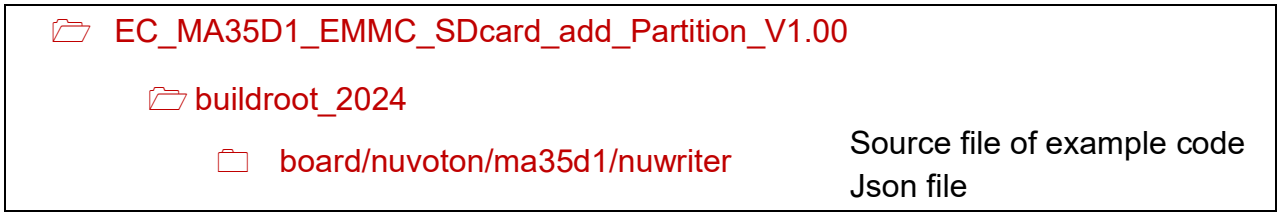


圖 4-1 目錄資訊

5. 範例程式執行

1. 根據第 2 章修改完成 *board/nuvoton/ma35d1/post-image.sh* 、*board/nuvoton/ma35d1/nuwriter/pack-sdcard.json* 兩個檔後，進入 buildroot 根目錄下，執行 make，最後會顯示分區方案在虛擬機器下的執行結果，以及 EBR 值，如下圖。

```
>>>   Executing post-image script board/nuvoton/ma35d1/post-image.sh
Model:  (file)
Disk /home/user/buildroot_2024/output/images/core-image-buildroot-ma35d1-som-512m.rootfs.sdc
rd: 17.2GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type      File system  Flags
 1      33.6MB  289MB   256MB   primary
 2      289MB   1363MB  1074MB  primary
 3      1363MB  3511MB  2147MB  primary
 4      3511MB  12.1GB  8590MB  extended               lba
 5      3515MB  4589MB  1074MB  logical
 6      4593MB  5130MB  537MB   logical
 7      5134MB  6208MB  1074MB  logical

EBR1 addr: d1400000
EBR2 addr: 111bf0000
EBR3 addr: 131ff0000
Generate output image(s) in directory 0415-162804164311 complete
Generate pack file in directory 0415-162805162393 complete
```

圖 5-1 buildroot 下編譯資訊

2. 編譯完成後，output/images 路徑下將生成包含 MBR 及 EBR 的 pack 鏡像，通過 NuWriter 燒錄至開發板。
3. 啟動後執行結果如圖 5-2，圖 5-3，圖中分區 mmcblk1p2~7 即為新加的分區，其中 mmcblk1p4 為擴展分區，不能實際儲存資料。

```
# fdisk -l
Disk /dev/mmcblk1: 15 GB, 15634268160 bytes, 30535680 sectors
238560 cylinders, 4 heads, 32 sectors/track
Units: sectors of 1 * 512 = 512 bytes

Device       Boot StartCHS    EndCHS        StartLBA     EndLBA    Sectors  Size Id Type
/dev/mmcblk1p1    512,0,1     1023,3,32        65536     565247     499712  244M 83 Linux
/dev/mmcblk1p2    1023,3,32   1023,3,32       565248    2662399    2097152 1024M 83 Linux
/dev/mmcblk1p3    1023,3,32   1023,3,32      2662400    6856703    4194304 2048M 83 Linux
/dev/mmcblk1p4    1023,3,32   1023,3,32      6856704   23633919   16777216 8192M  f Win95 Ext'd (LBA)
/dev/mmcblk1p5    1023,3,32   1023,3,32      6864896    8962047    2097152 1024M 83 Linux
/dev/mmcblk1p6    1023,3,32   1023,3,32      8970240   10018815    1048576  512M 83 Linux
/dev/mmcblk1p7    1023,3,32   1023,3,32     10027008   12124159    2097152 1024M 83 Linux
```

圖 5-2 增加分區後 fdisk 執行結果

```
# parted /dev/mmcblk1 print
Model: MMC MMC16G (sd/mmc)
Disk /dev/mmcblk1: 15.6GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type      File system  Flags
 1      33.6MB  289MB   256MB   primary   ext4
 2      289MB   1363MB  1074MB  primary   ext4
 3      1363MB  3511MB  2147MB  primary   ext4
 4      3511MB  12.1GB  8590MB  extended               lba
 5      3515MB  4589MB  1074MB  logical   ext4
 6      4593MB  5130MB  537MB   logical   ext4
 7      5134MB  6208MB  1074MB  logical   ext4
```

圖 5-3 增加分區後 parted 指令執行結果

4. 以上主要磁碟分割及邏輯分區可以使用 mkfs.ext4 格式化之後，掛載至合適目錄使用。
5. 使用本文方案時，可靈活選擇，如果需要的分區數小於或等於 4 個，則直接分幾個主要磁碟分割，不需要再分擴展分區和邏輯分區。
6. 本文方案以 MA35D1 平臺為例完成，也適用於 MA35 series 其他平臺，可根據實際需求參考使用。

6. 修訂記錄

Date	Revision	Description
2026.04.15	1.00	初始發佈

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.

Please note that all data and specifications are subject to change without notice.
All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.