MA35D1 OTA Update by Uboot Stage

NuMicro® 64/32位系列微控制器範例程式碼介紹

文件資訊

應用簡述	本範例程式說明如何於 MA35D1 系列開發平台下在 uboot stage 中進行 OTA 更新。
BSP 版本	Linux-5.10.x
開發平台	NuMaker-HMI-MA35D1-S1

1. 概述

本篇應用將演示如何於 uboot 實現 over-the-air (OTA)對 kernel、device-tree 及 root filesystem 韌體更新。新唐提供一種簡易更新方式: 透過 mtd (Memory Technology Device)指令及 mmc(Multi Media Card)指令對相應硬體儲存設備進行覆寫，達到更新之目的。

本應用於 MA35D1 系列開發板平台上演示 OTA 更新。MA35D1 系列開發板硬體支持 SDcard 、eMMC、 NAND 和 SPI-NAND Flash，用戶可針對其硬體配置設計，並更新對應裝置的軟體包。如，於 NAND/SPI-NAND Flash 儲存空間而言，使用 mtd 方式，覆寫硬體儲存裝置的對應位置，達到局部更新 Linux kernel 或 device-tree 等功能；同時支持覆寫 SDcard 等硬體設備局部更新等功能。對於記憶體分區如圖 1-1 所示。

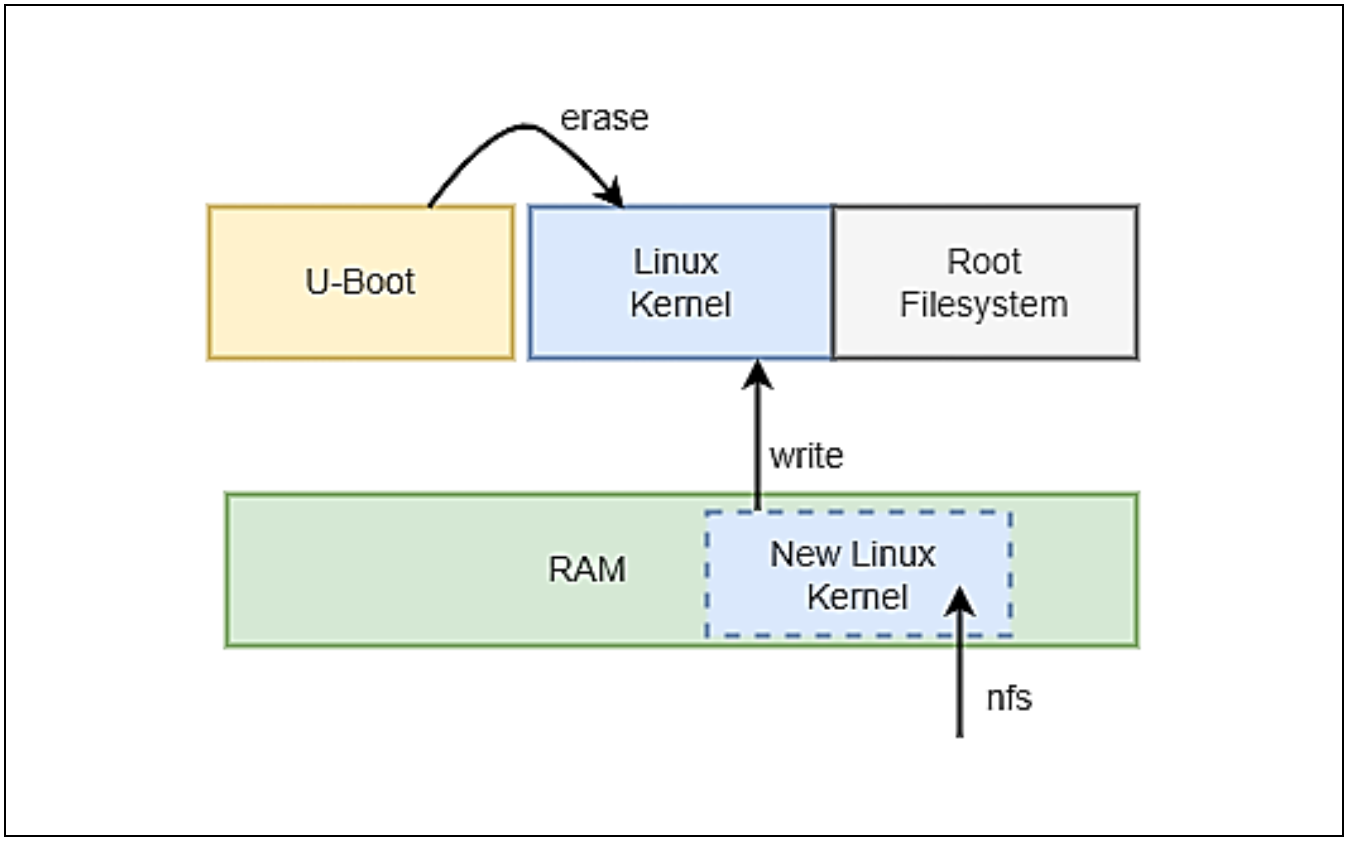

圖 1-1 記憶體配置

於 uboot 空間中 OTA 更新步驟流程如圖 1-2 所示，將預更新的檔案(linux kernel、device-tree 及 root filesystem)，下載至嵌入式設備，為了避免更新失敗請確保更新檔案之正常運作，於命令視窗中執行 OTA 指令，將會清除舊的檔案包並且將新的檔案包覆寫至原始檔案位置。OTA 指令提供抹除原始資料區及將暫存空間(0x80800000)裡的檔案寫入原始資料區的功能設計，重新啟動完成更新動作。

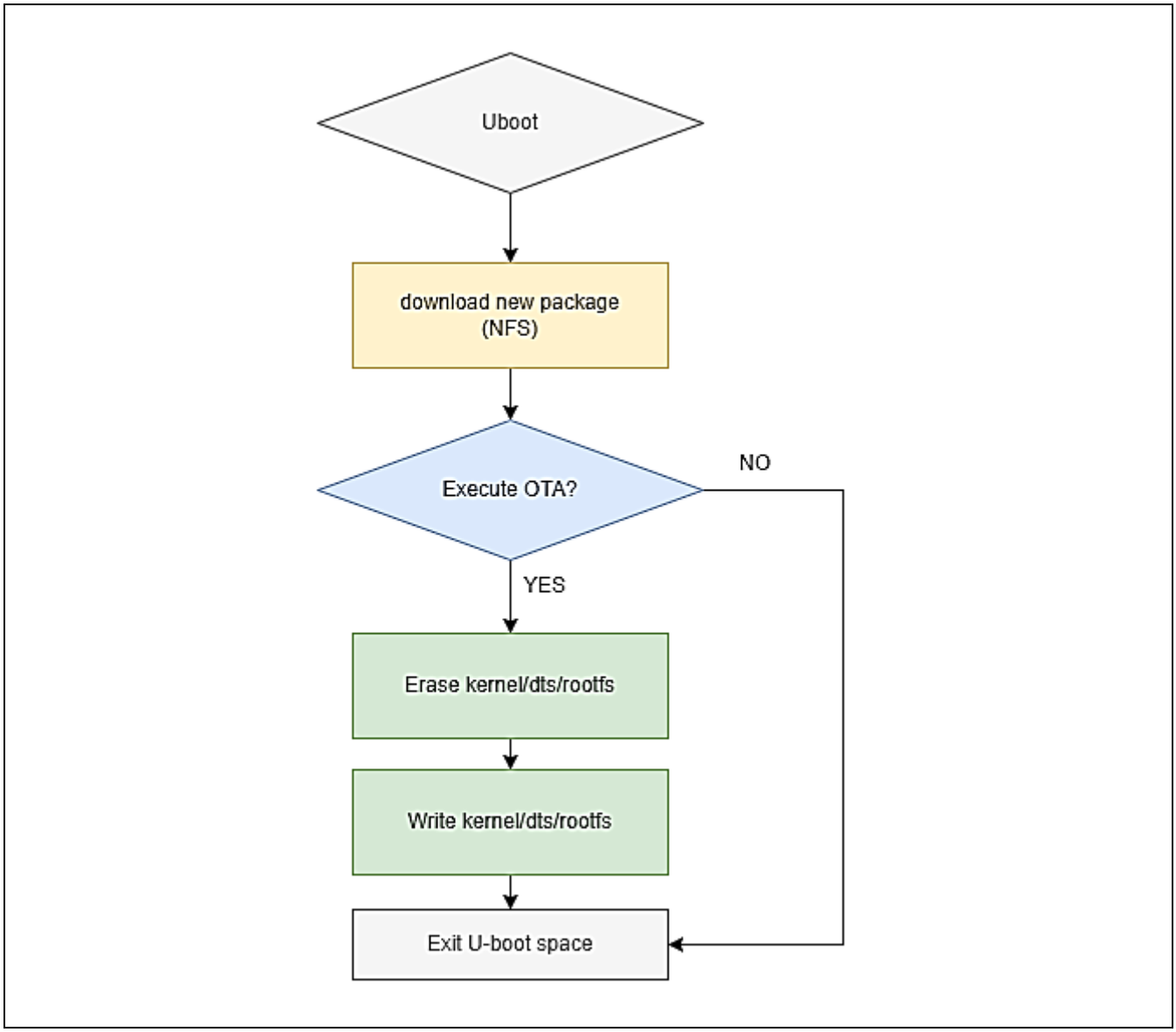

圖 1-2 本範例流程圖

1.1 環境設定

範例程式 OTA cmd: ma35d1_ota_update 可於 Nuvoton MA35D1 官網下載範例程式，新增功能指令於 uboot cmd 並於編譯時將其編入 U_BOOT_CMD 作為指令使用。本範例將展示於 uboot 中執行 device-tree、kernel 及 root filesystem 的更新例程。MA35D1 環境架設，產生映像檔可參考新唐 MA35D1 Evaluation Board Quick Start 。

於更新作業前，首要準備預更新的檔案(如 kernel、device-tree 或 root filesystem)，移入至 MA35D1 裝置上，用戶可自行選擇移入的方式，如 TFTP、NFS 等方式都是可以下載至 MA35D1 裝置上。本應用使用 network file system(NFS)將欲更新的 kernel 移入 MA35D1 裝置開發平台，並於裝置上進入 uboot 程序，覆寫舊的 kernel 軟體包，再重新執行開機程序，載入新的 kernel。以下說明 buildroot 及 yocto project 的操作說明:

- **Buildroot**

將 ma35d1_ota_update.c 放到 MA35D1 Buildroot uboot-custom/cmd 資料夾下:

```
Buildroot u-boot path: ~MA35D1_Buildroot/output/build/uboot-custom/cmd
```

修改 cmd/Makefile，並添加下列指令:

```
obj-y += ma35d1_ota_update.o
```

儲存並退出。接著安裝 NFS (network file system)，NFS 是一個 RPC service，能夠共享檔案，本範例使用 NFS 將預更新的軟體包放置到 MA35D1 開發平台上，若您使用別的方式掛載更新軟體包，則可以略過此步驟。

buildroot u-boot 下勾選 nfs 功能及 mtd 功能指令，如下所示:

```
~/MA35D1_Buildroot$ make uboot-menuconfig
.
Command line interface --->
      [*] Network commands --->
      -*- dhcp
      .....
      [*] nfs

      Device access commands--->
      [*] mtd
```

儲存並退出，執行 make 指令更新 image:

```
~/MA35D1_Buildroot$ make uboot-rebuild arm-trusted-firmware-rebuild
~/MA35D1_Buildroot$ make
```

編譯完成後請更新並透過 NuWriter 燒錄及啟動 MA35D1 開發平台。

- **Yocto Project**

將 ma35d1_ota_update.c 放到 MA35D1 yocto u-boot/git/cmd 資料夾下：

```
Yocto u-boot path:  ~yocto/build/tmp-glibc/work/numaker_som_ma35d16a81-poky-linux/u-boot-ma35d1/2020.07-r0/git/cmd
```

修改 /git/cmd/Makefile，並添加下列指令:

```
obj-y += ma35d1_ota_update.o
```

儲存並退出，接著安裝 NFS (network file system)，NFS 是一個 RPC service，能夠共享檔案，本範例使用 NFS 將預更新的軟體包放置到 MA35D1 開發平台上，若您使用別的方式掛載更新軟體包，則可以略過此步驟。

MA35D1 支持 NAND Flash/SDcard/SPI-NAND Flash …等 ，yocto project 提供對應的 u-boot 設定檔，用戶可以針對各自的開機方式選擇至對應的設定檔:

- ma35d1_nand_defconfig
- ma35d1_sdcard0_defconfig
- ma35d1_spinand_defconfig

前往 yocto project u-boot file ，選擇對應的 u-boot 資料夾下，(本範例展示以 sdcard0 開機方式):

```
~$ cd ~/shared/yocto/build/tmp-glibc/work/numaker_som_ma35d16a81-poky-linux/u-boot-ma35d1/2020.07-r0/build/

~$ cd ma35d1_sdcard0_defconfig
```

並執行下方指令，進入 u-boot-menuconfig 勾選 nfs 功能及 mtd 功能指令，如下所示:

```
~ ma35d1_sdcard0_defconfig $ make menuconfig
.
Command line interface --->
      [*] Network commands --->
          -*- dhcp
     ....
      [*] nfs

      Device access commands--->
      [*] mtd
```

注意: 編譯時會使用到 toolchain: source /usr/local/oecore-x86_64/environment-setup-aarch64-poky-linux。請參考新唐 MA35D1 Evaluation Board Quick Start 。

儲存並退出，執行 bitbake 指令編譯 u-boot 及更新 image:

```
~/shared/yocto/build$ bitbake u-boot-ma35d1 -C compile
~/shared/yocto/build$ bitbake nvt-image-qt5 -c clean
~/shared/yocto/build$ bitbake nvt-image-qt5
```

編譯完成後請更新並透過 NuWriter 燒錄及啟動 MA35D1 開發平台。

1.2 OTA 命令說明

啟動MA35D1開發板，在進入kernel之前，按任意鍵進入 uboot 環境，輸入下列指令查看OTA指令:

```
MA35D1> ota help
```

OTA cmd 結構如下圖 1-3 所示:

```
MA35D1> ota
ota - ota utils

Usage:
ota - Nuvoton MA35D1 OTA Update by Uboot stage

------------------------------------------------
ota ┬ nand   -- ┬ kernel  ${kernel size <hex>}  --   Update linux kernel
    ├ spiflash ├ dts     ${dts size <hex>}     --   Update dts
    ├ emmc     └ rootfs  ${rootfs size <hex>}  --   Update rootfs
    └ sdcard

 ** If cmd has no parameter size, the default value is used.
------------------------------------------------
```

圖 1-3 OTA 指令結構

此命令結構固定從 RAM addr:0x80800000 寫入對應區塊，size 大小，則採用 partition 設置:

CMD Format	Description
ota nand [part] [size]	抹除NAND中對應的位置，並將RAM(0x80800000)寫入NAND中對應區域。 part ： kernel、dts、rootfs size：更新的檔案大小(hex)，default: kernel=24M、dts=256K、rootfs=100M
ota spinand [part] [size]	抹除SPI-NAND中對應的位置，並將RAM (0x80800000)寫入SPI-NAND中對應區域。

	part ： kernel、dts、rootfs size：更新的檔案大小(hex)，default: kernel=24M、dts=256K、rootfs=100M
ota sdcard [part] [size]	抹除 SDcard 中對應的位置，並將 RAM (0x80800000) 寫入 SDcard中對應區域。 part ： kernel、dts size : 更新的檔案大小(hex)，default: kernel=16M、dts=64K
ota emmc [part] [size]	抹除eMMC中對應的位置，並將RAM (0x80800000)寫入eMMC中對應區域。 part ： kernel、dts size : 更新的檔案大小(hex)，default: kernel=16M、dts=64K

表 1-1 OTA 命令表

1.3 執行結果

步驟一: 將更新軟體包透過NFS方式下載至開發平台。

本應用使用 network file system(NFS)將欲更新的 kernel 移入 MA35D1 裝置開發平台，於 PC host 安裝 nfs service:

```
ubuntu:~$ sudo apt-get install nfs-kernel-server rpcbind
```

設定共享資料夾:

```
~$ sudo vi /etc/exports
```

新增 ${NFS_FOLDER_PATH}路徑為欲分享的資料夾:

```
${NFS_FOLDER_PATH} *(rw,sync,no_root_squash)
```

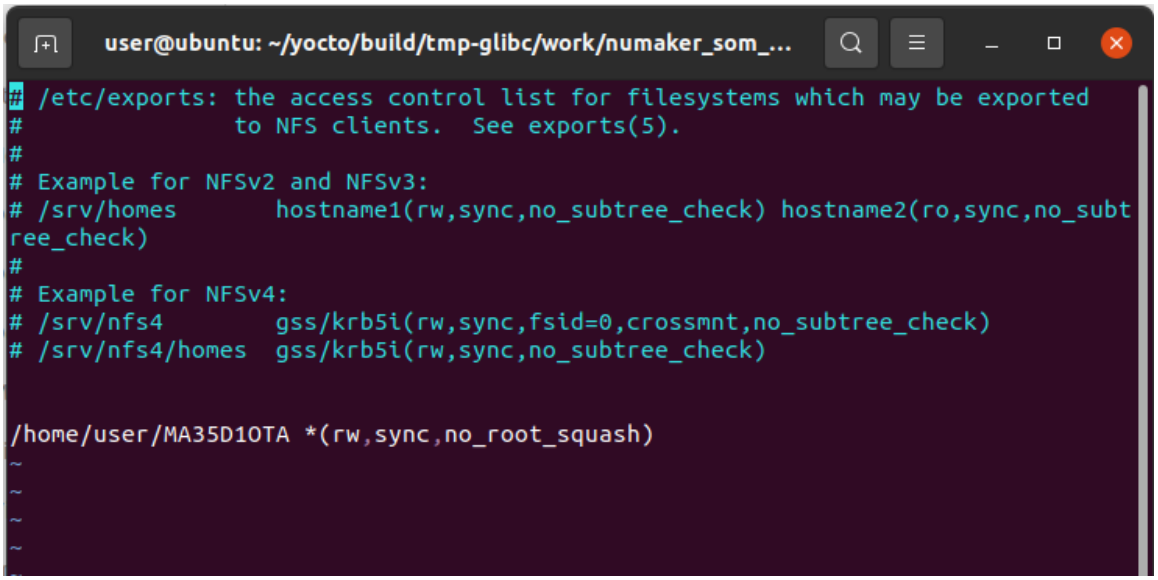

圖 1-4 NFS 設定

重新啟動 nfs service:

```
$sudo /etc/init.d/nfs-kernel-server restart
```

回到 MA35D1 開發平台，於進入 kernel 之前，按任意鍵進入 uboot 環境:

並接上網路線取得 IP 位置:

```
MA35D1> dhcp
```

使用 nfs cmd 下載欲更新的檔案:

```
MA35D1>nfs [loadAddress] [[host IP addr]:[filename]
```

loadAddress: 0x80800000 MA35D1 RAM 的暫存空間。

host IP addr: PC host 的網路 ip 位置。

filename: 更新的檔案路徑。

下載 device-tree:

```
MA35D1> nfs 0x80800000 192.168.10.103:/home/user/buildroot/output/images/Image.dtb
```

下載 kernel:

```
MA35D1> nfs 0x80800000 192.168.10.103:/home/user/buildroot/output/images/Image
```

下載 rootfs:

```
MA35D1> nfs 0x80800000 192.168.10.103:/home/user/buildroot/output/images/rootfs.ubi
```

```
MA35D1> nfs 0x80800000 192.168.10.100:/home/user/MA35D1OTA/HMI/nand/Image
Speed: 100, full duplex
Using ethernet@40120000 device
File transfer via NFS from server 192.168.10.100; our IP address is 192.168.10.101
Filename '/home/user/MA35D1OTA/HMI/nand/Image'.
Load address: 0x80800000
Loading: #################################################################
         ##############
done
Bytes transferred = 14051840 (d66a00 hex)
MA35D1>
```

圖 1-5 執行 NFS 執行頁面

步驟二: 清除對應區域及寫入檔案。

依 NAND 為例並更新 kernel :

```
MA35D1> ota nand kernel d67000
```

注意，輸入$size 以 page 為單位(0x800)。程式將透過下列指令覆蓋:

1. 抹除 kernel (NAND) :

   MA35D1> *mtd erase kernel*

2. 從 0x80800000 寫入 kernel (size: 0xd67000 bytes):

   MA35D1> *mtd write kernel 0x80800000 0x0 0xd67000*

```
MA35D1> ota nand kernel d67000
Erase the kernel (nand)
Erasing 0x00000000 ... 0x017fffff (192 eraseblock(s))
Overwrite kernel from 0x80800000 size: (d67000)
Writing 14053376 byte(s) (6862 page(s)) at offset 0x00000000
MA35D1>
```

圖 1-6 執行 OTA 畫面

2. 代碼介紹

設定 uboot 命令列名稱及所對應的函式，說明如註解所示。

```
/* Uboot CMD */
/* Define a U-Boot command "ota" with subcommands */
U_BOOT_CMD_WITH_SUBCMDS(ota, "ota utils", help_text,

    /* Subcommand list starts */
    /* Subcommand name, arguments, flags, Function to execute for this subcommand */
    U_BOOT_SUBCMD_MKENT(help,    2, 0, print_help_text),
    U_BOOT_SUBCMD_MKENT(nand,    4, 0, do_ma35d1_nand_ota),
    U_BOOT_SUBCMD_MKENT(spinand, 4, 0, do_ma35d1_spinand_ota),
    U_BOOT_SUBCMD_MKENT(emmc,    4, 0, do_ma35d1_emmc_ota),
    U_BOOT_SUBCMD_MKENT(sdcard,  4, 0, do_ma35d1_sdcard_ota)
    );
```

do_ma35d1_nand_ota:對 NAND 處理執行命令處理。

```
static void do_ma35d1_nand_ota(struct cmd_tbl *cmdtp, int flag, int argc, char *const
argv[])
{
    if(argc == 2) {
        if ( !(strcmp(argv[1], "kernel")) ){
            /* Erase the kernel (nand) */
               printf("Erase the kernel (nand)");
            run_command("mtd erase kernel", 0);

            /* Overwrite to kernel from 0x80800000 size:(24M) */
                printf("Overwrite to kernel from 0x80800000 size:(24M)");
            run_command("mtd write kernel 0x80800000 0x0 0x1800000", 0);
        }
        else if ( !(strcmp(argv[1], "dts")) ){
            /* Erase the device-tree (nand) */
               printf("Erase the device-tree (nand)");
            run_command("mtd erase device-tree", 0);

            /* Overwrite to device-tree from 0x80800000 size:(256k) */
               printf("Overwrite to device-tree from 0x80800000 size:(256k)");
            run_command("mtd write device-tree 0x80800000 0x0  0x40000", 0);
        }
        else if ( !(strcmp(argv[1], "rootfs")) ){
            /* Erase the rootfs (nand) */
               printf("Erase the rootfs (nand)");
            run_command("mtd erase rootfs", 0);

            /* Overwrite to rootfs from 0x80800000 size:(100m) */
                printf("Overwrite to rootfs from 0x80800000 size:(100m)");
            run_command("mtd write rootfs 0x80800000 0x0 0x6400000", 0);
        }
        else
            printf("%s: parameters not found.\n", argv[1]);
    }
     else if (argc == 3) {
           char *size = argv[2];
        if (!(strcmp(argv[1], "kernel"))) {
            /* Erase the kernel (nand) */
            printf("Erase the kernel (nand)\r\n");
            run_command("mtd erase kernel", 0);

            /* Overwrite kernel from 0x80800000 with the specified size */
```

```
            printf("Overwrite kernel from 0x80800000 size: (%s)\r\n", size);
            char command[100];
            snprintf(command, sizeof(command), "mtd write kernel 0x80800000 0x0 %s",
size);
            run_command(command, 0);
        }
        else if (!(strcmp(argv[1], "dts"))) {
            /* Erase the device-tree (nand) */
            printf("Erase the device-tree (nand) \r\n");
            run_command("mtd erase device-tree", 0);

            /* Overwrite device-tree from 0x80800000 with the specified size */
            printf("Overwrite device-tree from 0x80800000 size: (%s) \r\n", size);
            char command[100];
            snprintf(command, sizeof(command), "mtd write device-tree 0x80800000 0x0 %s",
size);
            run_command(command, 0);
        }
        else if (!(strcmp(argv[1], "rootfs"))) {
            /* Erase the rootfs (nand) */
            printf("Erase the rootfs (nand)\r\n");
            run_command("mtd erase rootfs", 0);

            /* Overwrite rootfs from 0x80800000 with the specified size */
            printf("Overwrite rootfs from 0x80800000 size: (%s)\r\n", size);
            char command[100];
            snprintf(command, sizeof(command), "mtd write rootfs 0x80800000 0x0 %s",
size);
            run_command(command, 0);
        }
        else
            printf("%s: parameters not found.\n", argv[1]);
     }
    else
        printf("Invalid number of arguments.\n");
}
```

do_ma35d1_sdcard_ota:對 SDcard 處理執行命令處理。

```
static void do_ma35d1_sdcard_ota(struct cmd_tbl *cmdtp, int flag, int argc, char *const
argv[])
{
    if(argc == 2) {
        if ( !(strcmp(argv[1], "kernel")) ){
            /* "Erase the kernel (sdcard) */
            printf("Erase the kernel (sdcard)");
            run_command("mmc erase 0x1800 0x8000", 0);

            /* Overwrite to kernel from 0x80800000 size:(16M) */
                printf("Overwrite to kernel from 0x80800000 size:(16M)");
            run_command("mmc write 0x80800000 0x1800 0x8000", 0);
        }
        else if ( !(strcmp(argv[1], "dts")) ){
            /* Erase the device-tree (sdcard) */
               printf("Erase the device-tree (sdcard)");
            run_command("mmc erase 0x1600 0x80", 0);

            /* Overwrite to device-tree from 0x80800000 size:(64k) */
               printf("Overwrite to device-tree from 0x80800000 size:(64k)");
            run_command("mmc write 0x80800000 0x1600 0x80", 0);
        }
        else if ( !(strcmp(argv[1], "rootfs")) ){
            /* The rootfs.ext4 is too big. */
            printf("[E]:Non-function ! \n");
```

```
        }
        else
            printf("%s: parameters not found.\n", argv[1]);
    }
     else if (argc == 3) {
        char *size = argv[2];

        if (!(strcmp(argv[1], "kernel"))) {
            /* Erase the kernel (sdcard) */
            printf("Erase the kernel (sdcard)\r\n");
            run_command("mmc erase 0x1800 0x8000", 0);

            /* Overwrite kernel from 0x80800000 with the specified size */
            printf("Overwrite kernel from 0x80800000 size: (%s)\r\n", size);
            char command[100];
            snprintf(command, sizeof(command), "mtd write kernel 0x80800000 0x1800  %s",
size);
            run_command(command, 0);
        }
        else if (!(strcmp(argv[1], "dts"))) {
            /* Erase the device-tree (sdcard) */
            printf("Erase the device-tree (sdcard) \r\n");
            run_command("mtd erase device-tree", 0);

            /* Overwrite device-tree from 0x80800000 with the specified size */
            printf("Overwrite device-tree from 0x80800000 size: (%s) \r\n", size);
            char command[100];
            snprintf(command, sizeof(command), "mtd write device-tree 0x80800000
0x1600 %s", size);
            run_command(command, 0);
        }
        else if (!(strcmp(argv[1], "rootfs"))) {
            /* The rootfs.ext4 is too big. */
            printf("[E]: Non-function!\n");
        }
        else
            printf("%s: parameters not found.\n", argv[1]);
    }
    else
        printf("Invalid number of arguments.\n");

}
```

3. 軟體與硬體需求

3.1 軟體需求

- BSP 版本
  - Linux-5.10.x

3.2 硬體需求

- 電路元件
  - NuMaker-HMI-MA35D1-S1

4. 目錄資訊

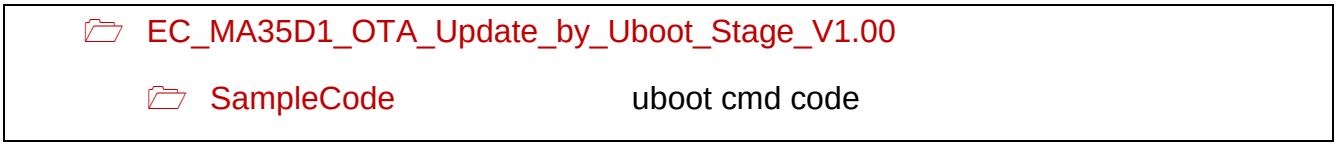

圖 4-1 目錄資訊

5. 範例程式執行

1. 將更新軟體包透過 NFS 方式下載至開發平台。

2. 清除對應區域及寫入檔案。

6. 修訂紀錄

Date	Revision	Description
2023.03.13	1.00	初始發布。

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.