MA35D1 OTA Update by Uboot Stage

NuMicro® 64/32位系列微控制器范例代码介绍

文件信息

应用简述	本范例代码说明如何于 MA35D1 系列开发平台下，在 uboot stage 中进行 OTA 更新。
BSP 版本	Linux-5.10.x
开发平台	NuMaker-HMI-MA35D1-S1

1. 概述

本篇应用将演示如何于 uboot 实现 over-the-air (OTA)对 kernel、device-tree 及 root filesystem 韧体更新。新唐提供一种简易更新方式: 透过 mtd (Memory Technology Device)指令及 mmc(Multi Media Card)指令对相应硬件储存设备进行覆写，达到更新之目的。

本应用于 MA35D1 系列开发板平台上演示 OTA 更新。MA35D1 系列开发板硬件支持 SDcard 、eMMC、 NAND 和 SPI-NAND Flash，用户可针对其硬件配置设计，并更新对应装置的软件包。如，于 nand/spinand flash 储存空间而言，使用 mtd 方式，覆写硬件储存装置的对应位置，达到局部更新 Linux kernel 或 device-tree 等功能；同时支持覆写 SDcard 等硬件设备局部更新等功能。对于内存分区如图 1-1 所示。

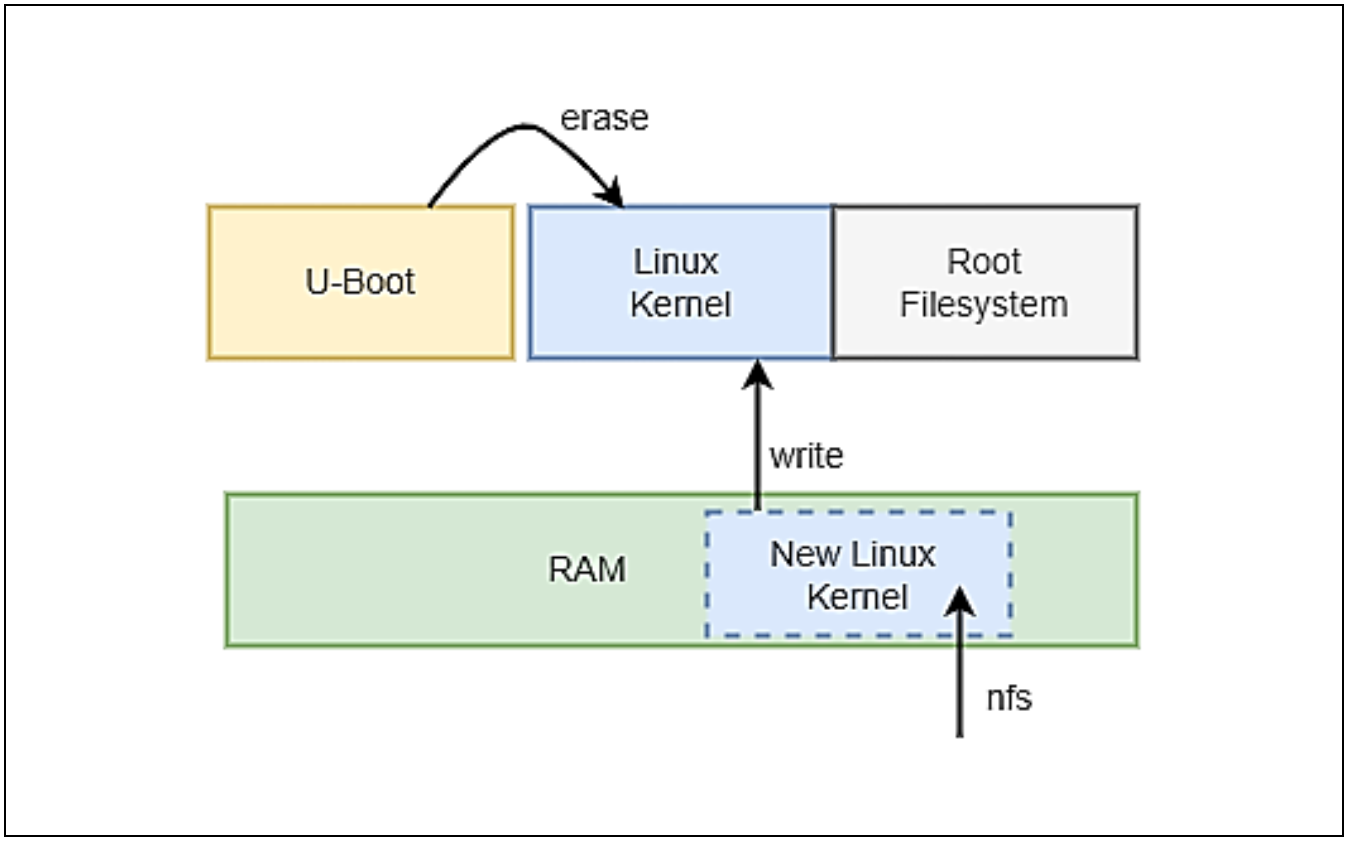

图 1-1 内存配置

于 uboot 空间中 OTA 更新步骤流程如图 1-2 所示，将预更新的档案(linux kernel、device-tree 及 root filesystem)，下载至嵌入式设备，为了避免更新失败请确保更新档案之正常运作，在命令窗口中执行 OTA 指令，将会清除旧的档案包并且将新的档案包覆写至源文件位置。OTA 指令提供抹除原始数据区及将暂存空间(0x80800000)里的档案写入原始数据区的功能设计，重新启动完成更新动作。

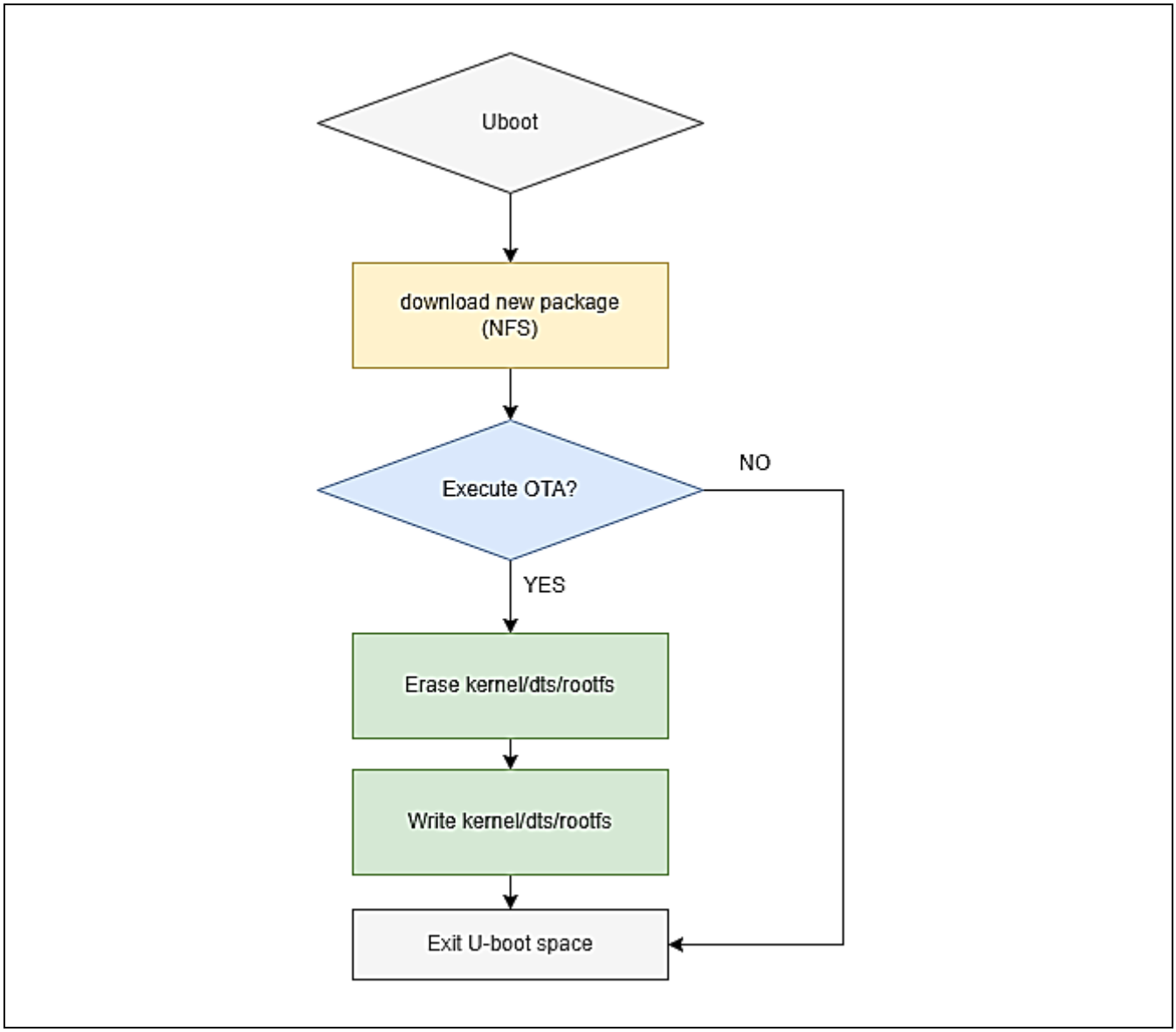

图 1-2 本范例流程图

1.1 环境设定

范例程序 OTA cmd: ma35d1_ota_update 可于 Nuvoton MA35D1 官网下载范例程序，新增功能指令于 uboot cmd 并于编译时将其编入 U_BOOT_CMD 作为指令使用。本范例将展示于 uboot 中执行 device-tree、kernel 及 root filesystem 的更新例程。MA35D1 环境架设，产生映像档可参考新唐 MA35D1 Evaluation Board Quick Start 。

于更新作业前，首要准备预更新的档案(如 kernel、device-tree 或 root filesystem)，移入至 MA35D1 装置上，用户可自行选择移入的方式，如 TFTP、NFS 等方式都是可以下载至 MA35D1 装置上。本应用使用 network file system(NFS)将欲更新的 kernel 移入 MA35D1 装置开发平台，并于装置上进入 uboot 程序，覆写旧的 kernel 软件包，再重新执行开机程序，加载新的 kernel。以下说明 buildroot 及 yocto project 的操作说明:

- **Buildroot**

将ma35d1_ota_update.c 放到MA35D1 Buildroot uboot-custom/cmd文件夹下:

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

修改 cmd/Makefile，并添加下列指令:

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

储存并退出。接着安装 NFS (network file system)，NFS 是一个 RPC service，能够共享档案，本范例使用 NFS 将预更新的软件包放置到 MA35D1 开发平台上，若您使用别的方式挂载更新软件包，则可以略过此步骤。

buildroot u-boot 下勾选 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
```

储存并退出，执行 make 指令更新 image:

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

编译完成后请更新并透过 NuWriter 烧录及启动 MA35D1 开发平台。

- **Yocto Project**

将ma35d1_ota_update.c 放到MA35D1 yocto u-boot/git/cmd文件夹下：

```
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，并添加下列指令:

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

储存并退出，接着安装NFS (network file system)，NFS是一个RPC service，能够共享档案，本范例使用NFS将预更新的软件包放置到MA35D1开发平台上，若您使用别的方式挂载更新软件包，则可以略过此步骤。

MA35D1支持 NAND Flash/SDcard/SPI-NAND Flash …等 ，yocto project 提供对应的u-boot配置文件，用户可以针对各自的开机方式选择至对应的配置文件:

- ma35d1_nand_defconfig
- ma35d1_sdcard0_defconfig
- ma35d1_spinand_defconfig

前往 yocto project u-boot file ，选择对应的 u-boot 文件夹下，(本范例展示以 sdcard0 开机方式):

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

~$ cd ma35d1_sdcard0_defconfig
```

并执行下方指令，进入 u-boot-menuconfig 勾选 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: 编译时会使用到 toolchain: source /usr/local/oecore-x86_64/environment-setup-aarch64-poky-linux。請参考新唐 MA35D1 Evaluation Board Quick Start 。

储存并退出，执行 bitbake 指令编译 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
```

编译完成后请更新并透过NuWriter烧录及启动MA35D1 开发平台。

1.2 OTA 命令说明

启动MA35D1开发板，在进入kernel之前，按任意键进入 uboot 环境，输入下列指令查看OTA 指令:

```
MA35D1> ota help
```

OTA cmd 结构如下图 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.
------------------------------------------------
```

图 1-3 OTA 指令结构

此命令结构固定从RAM addr:0x80800000写入对应区块，size大小，则采用partition设置:

CMD Format	Description
ota nand [part] [size]	抹除NAND中对应的位置，并将RAM(0x80800000)写入NAND中对应区域。 part ： kernel、dts、rootfs size：更新的档案大小(hex)，default: kernel=24M、dts=256K、rootfs=100M
ota spinand [part] [size]	抹除SPI-NAND中对应的位置，并将RAM (0x80800000)写入SPI-NAND中对应区域。 part ： kernel、dts、rootfs

	size：更新的档案大小(hex)，default: kernel=24M、dts=256K、rootfs=100M
ota sdcard [part] [size]	抹除 SDcard 中对应的位置，并将 RAM (0x80800000) 写入 SDcard中对应区域。 part ： kernel、dts size : 更新的档案大小(hex)，default: kernel=16M、dts=64K
ota emmc [part] [size]	抹除eMMC中对应的位置，并将RAM(0x80800000)写入eMMC中对应区域。 part ： kernel、dts size : 更新的档案大小(hex)，default: kernel=16M、dts=64K

表 1-1 OTA 命令表

1.3 执行结果

步驟一: 更新軟體包透過 NFS 方式下載至開發平台。

本应用使用 network file system(NFS)将欲更新的 kernel 移入 MA35D1 装置开发平台，于 PC host 安装 nfs service:

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

设定共享文件夹:

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

新增 ${NFS_FOLDER_PATH}路径为欲分享的文件夹:

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

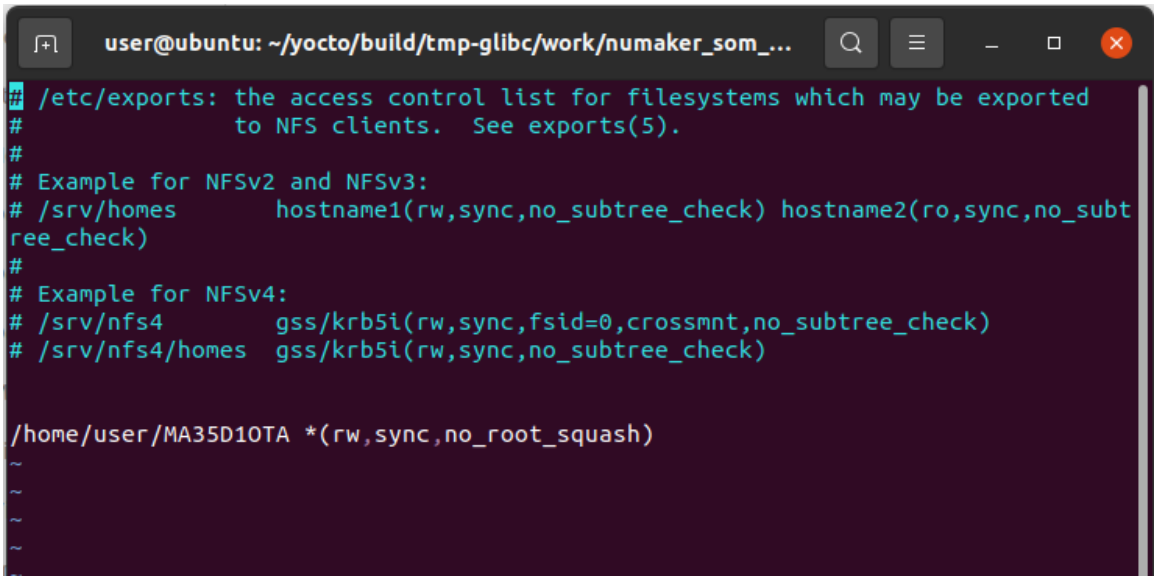

图 1-4 NFS 设定

重新启动 nfs service:

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

回到 MA35D1 开发平台，于进入 kernel 之前，按任意键进入 uboot 环境:

并接上网络线取得 IP 位置:

```
MA35D1> dhcp
```

使用 nfs cmd 下载欲更新的档案:

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

loadAddress: 0x80800000 MA35D1 RAM 的暂存空间

host IP addr: PC host 的网络 ip 位置

filename: 更新的档案路径

下载 device-tree:

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

下载 kernel:

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

下载 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>
```

图 1-5 执行 NFS 执行页面

步骤二: 清除对应区域及写入档案

依 NAND 为例并更新 kernel:

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

注意，输入$size 以 page 为单位(0x800)。程式将透过下列指令覆盖:

1. 抹除 kernel (NAND) :

   MA35D1> *mtd erase kernel*

2. 从 0x80800000 写入 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>
```

图 1-6 执行 OTA 画面

2. 代码介绍

设定 uboot 命令行名称及所对应的函式，说明如批注所示。

```
/* 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:对 NAND 处理执行命令处理

```
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_nand_ota:对 NAND 处理执行命令处理

```
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. 软件与硬件需求

3.1 软件需求

- BSP 版本
  - Linux-5.10.x

3.2 硬件需求

- 电路组件
  - NuMaker-HMI-MA35D1-S1

4. 目录信息

EC_MA35D1_OTA_Update_by_Uboot_Stage_V1.00

SampleCode uboot cmd code

图 4-1 目录信息

5. 范例程序执行

1. 将更新软体包透过 NFS 方式下载至开发平台。
2. 清除对应区域及写入档案

6. 修订纪录

Date	Revision	Description
2023.03.13	1.00	初始发布。

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.