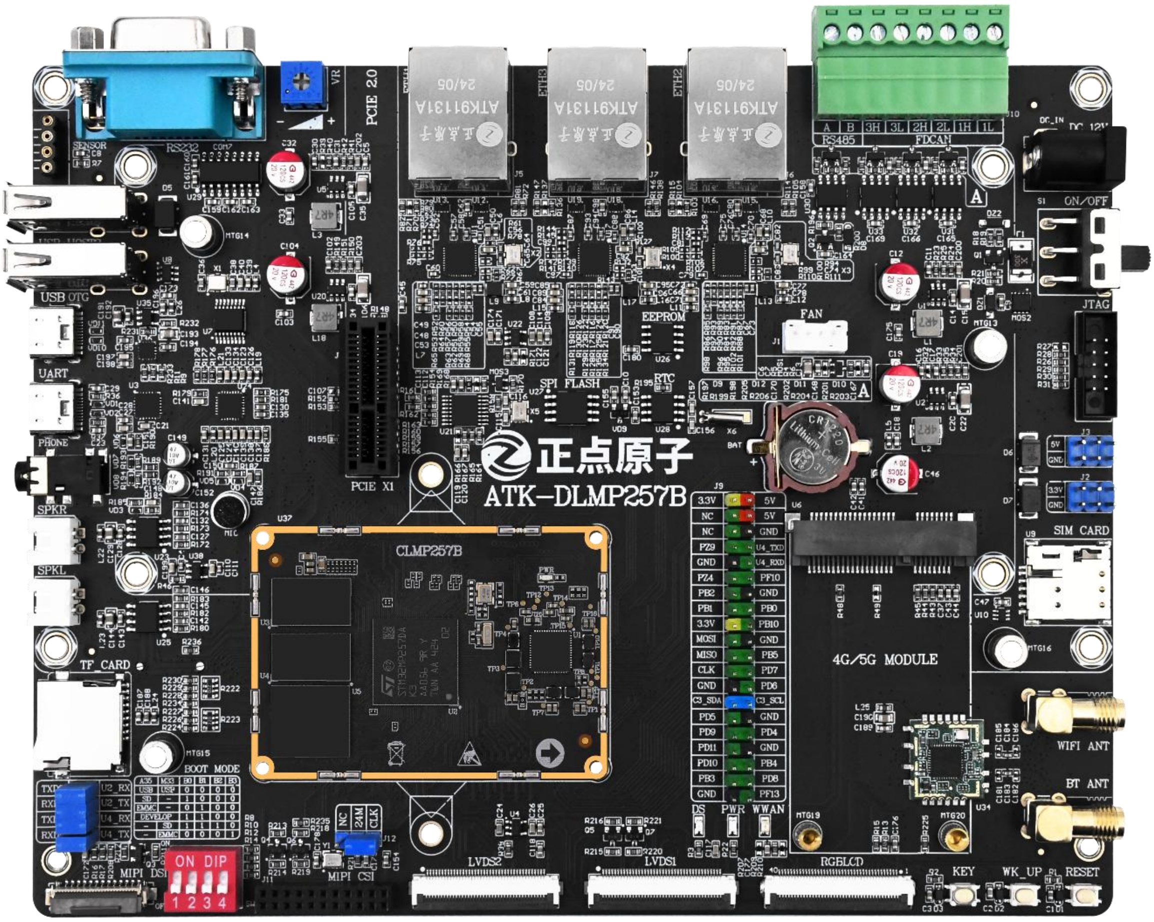
ATK-DLMP257B

Firmware update reference documentation

V1.0



1. Shopping：

TMALL：https://zhengdianyuanzi.tmall.com

TAOBAO：https://openedv.taobao.com

2. Download

Address：http://www.openedv.com/docs/index.html

3. FAE

Website ： www.alientek.com

Forum ： http://www.openedv.com/forum.php

Videos ： www.yuanzige.com

Fax ： +86 - 20 - 36773971

Phone ： +86 - 20 - 38271790

Disclaimer

The product specifications and instructions mentioned in this document are for reference only and subject to update without prior notice; Unless otherwise agreed, this document is intended as a product guide only, and none of the representations made herein constitutes a warranty of any kind. The copyright of this document belongs to Guangzhou Xingyi Electronic Technology Co., LTD. Without the written permission of the company, any unit or individual shall not be used for profit-making purposes in any way of dissemination.

In order to get the latest version of product information, please regularly visit the download center or contact the customer service of Taobao ALIENTEK flagship store. Thank you for your tolerance and support.

Revision History:

Version	Version Update Notes	Responsible person	Proofreading	Date
V1.0	release officially	ALIENTEK	ALIENTEK	2025.04.01

Catalogue

Chapter 1. Basic Operations

1.1 Factory source compilation

In [ALIENTEK] STM32MP257 development board (disk A) - Basic Information \10_user manual \ [ALIENTEK] ATK-DLMP257B Factory System Source Code Use Guide V1.0. There are detailed factory source code compilation methods in the pdf document. Users can modify the source code to add their own functions after compiling the factory source code. Then compile again, generate the required image firmware, files.

It is assumed that the user has already obtained the image firmware and files he needs.

1.2 File transfer between computer and development board

After users get the image firmware they need, the next step is to replace the image firmware to the development board system for updating, preliminary verification or debugging, which requires the operation of file transfer between the computer and the development board. File transfer operations include storage device transmission (U disk or TF card) and network transmission. For specific operations, please refer to [ALIENTEK] STM32MP257 development board (A disk) - Basic information \10_user manual \ [ALIENTEK] Reference manual V1.0 for transferring files between Ubuntu&Windows&Linux development boards.

For different images and files, the paths that need to be copied to the development board are also different. Taking the 2GB version as an example, here are the paths of commonly used images and files on the development board:

Filename	System Path (EMMC)	Notes
tf-a-stm32mp257d-atk-ddr-2GB-optee-emmc.stm32	/dev/mmcblk1boot0	Boot partition 1
tf-a-stm32mp257d-atk-ddr-2GB-optee-emmc.stm32	/dev/mmcblk1boot1	Boot partition 2
metadata.bin	/dev/mmcblk1p1	TF-A firmware update data
metadata.bin	/dev/mmcblk1p2	TF-A firmware update data
fip-stm32mp257d-atk-ddr-2GB-optee-emmc.bin	/dev/mmcblk1p3	FIP, including optee, uboot
	/dev/mmcblk1p4	FIP backup partition
Image.gz	/dev/mmcblk1p5	Kernel image
stm32mp257d-atk-ddr-2GB.dtb	/dev/mmcblk1p5	Other device trees are also present here
6.6.48	/dev/mmcblk1p5	Kernel module
mmc1_extlinux	/dev/mmcblk1p5	Uboot configuration file

If the system is made of TF cards, the corresponding device is named mmcblk0.

Chapter 2. Firmware partition table

2.1 EMMC partitioning

Partition name		Notes
/dev/mmcblk1boot0		Boot area partition 1
/dev/mmcblk1boot1		Boot area partition 2
User data area	/dev/mmcblk1p1	The metadata.bin partition
	/dev/mmcblk1p2	metadata.bin Backup partition
	/dev/mmcblk1p3	fip.bin partition
	/dev/mmcblk1p4	fip.bin Backup partition, not written
	/dev/mmcblk1p5	Bootfs partition
	/dev/mmcblk1p6	Rootfs partition

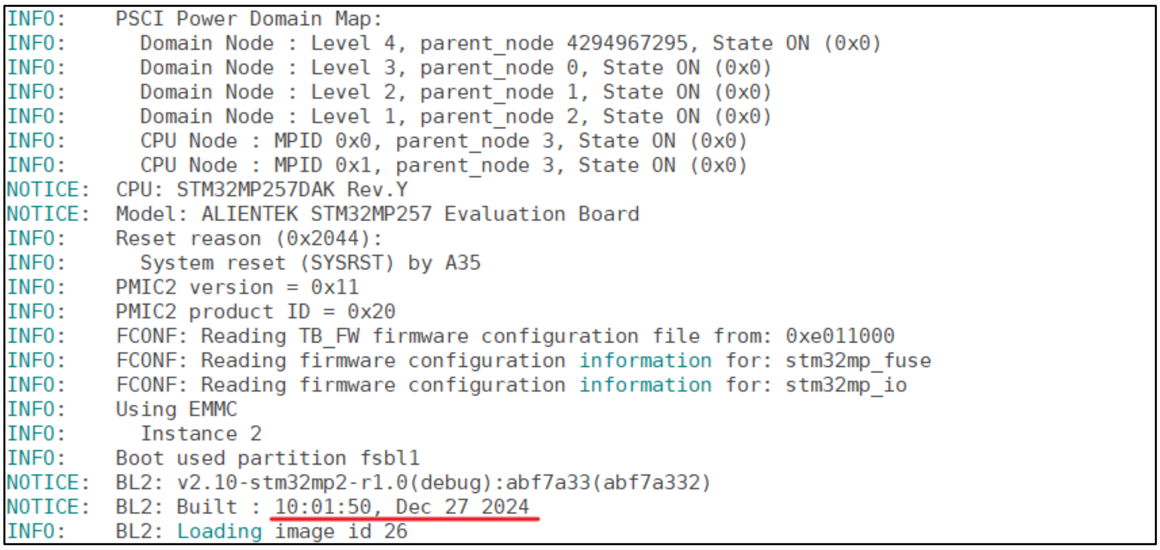
2.2 TF card partition

Partition name		Notes
/dev/mmcblk0boot0		Boot area partition 1
/dev/mmcblk0boot1		Boot area partition 2
User data area	/dev/mmcblk0p1	The metadata.bin partition
	/dev/mmcblk0p2	metadata.bin Backup partition
	/dev/mmcblk0p3	fip.bin partition
	/dev/mmcblk0p4	fip.bin Backup partition, not written
	/dev/mmcblk0p5	Bootfs partition
	/dev/mmcblk0p6	Rootfs partition

Chapter 3. Update firmware TF-A

3.1 Updating with dd instructions (EMMC)

Before updating, you can check the generation time of TF-A of the current development board to facilitate the comparison after subsequent updates and confirm the success of burning.

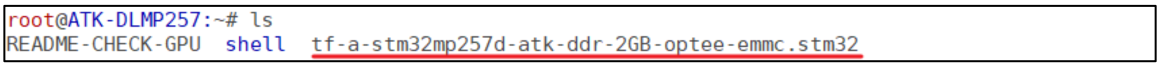


```
INFO:    PSCI Power Domain Map:
INFO:      Domain Node : Level 4, parent_node 4294967295, State ON (0x0)
INFO:      Domain Node : Level 3, parent_node 0, State ON (0x0)
INFO:      Domain Node : Level 2, parent_node 1, State ON (0x0)
INFO:      Domain Node : Level 1, parent_node 2, State ON (0x0)
INFO:      CPU Node : MPID 0x0, parent_node 3, State ON (0x0)
INFO:      CPU Node : MPID 0x1, parent_node 3, State ON (0x0)
NOTICE:  CPU: STM32MP257DAK Rev.Y
NOTICE:  Model: ALIENTEK STM32MP257 Evaluation Board
INFO:    Reset reason (0x2044):
INFO:      System reset (SYSRST) by A35
INFO:    PMIC2 version = 0x11
INFO:    PMIC2 product ID = 0x20
INFO:    FCONF: Reading TB_FW firmware configuration file from: 0xe011000
INFO:    FCONF: Reading firmware configuration information for: stm32mp_fuse
INFO:    FCONF: Reading firmware configuration information for: stm32mp_io
INFO:    Using EMMC
INFO:      Instance 2
INFO:    Boot used partition fsbl1
NOTICE:  BL2: v2.10-stm32mp2-r1.0(debug):abf7a33(abf7a332)
NOTICE:  BL2: Built : 10:01:50, Dec 27 2024
INFO:    BL2: Loading image id 26
```

Figure 3.1-1 TF-A generates temporal examples

Assuming that the user has compiled the tf-a, optee, uboot source code and packaged the tf-a tm32 file, the user can copy the tf-A tm32 file to the development board home directory and prepare for tf-A tm32 burning.

Here the author takes the 2GB version as an example, using tf-a-stm32mp257d-atk-ddr-2GB-optee-emmc.stm32



```
root@ATK-DLMP257:~# ls
README-CHECK-GPU  shell  tf-a-stm32mp257d-atk-ddr-2GB-optee-emmc.stm32
```

Figure 3.1-2 Copy Tf-a.sm32 to the development board

Execute the following command to enable emmc to start the partition before burning.

```
echo 0 > /sys/class/block/mmcblk1boot0/force_ro
```

<send_ack> =1 enables the boot acknowledge bit in the EMMC ext_csd register. The EMMC startup configuration is: 1-wire configuration and 25 MHz, which is done by command:

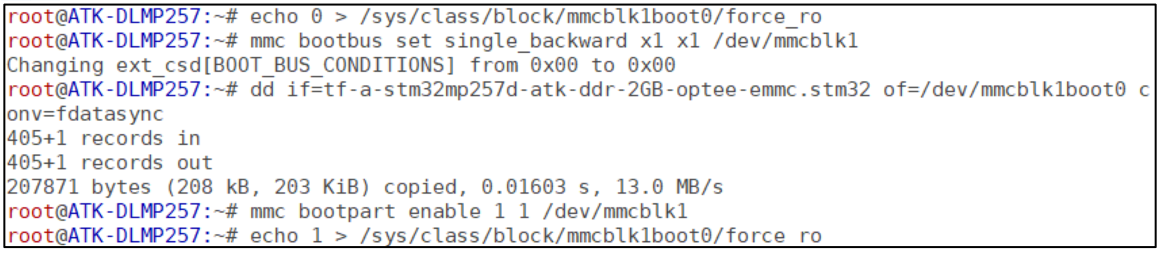
```
mmc bootbus set single_backward x1 x1 /dev/mmcblk1
```

Update TF-A in boot1 and select this boot partition 1 (default) :

```
dd if=tf-a-stm32mp257d-atk-ddr-2GB-optee-emmc.stm32 of=/dev/mmcblk1boot0 conv=fdatasync
mmc bootpart enable 1 1 /dev/mmcblk1
```

When the burn is complete, close the boot partition to be burned.

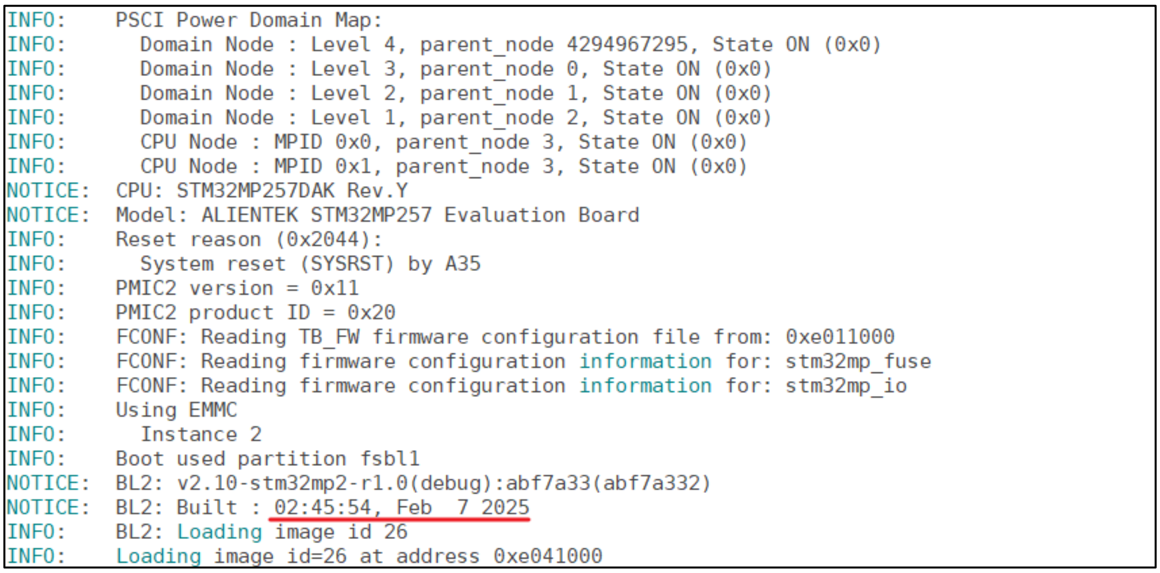
```
echo 1 > /sys/class/block/mmcblk1boot0/force_ro
```



```
root@ATK-DLMP257:~# echo 0 > /sys/class/block/mmcblk1boot0/force_ro
root@ATK-DLMP257:~# mmc bootbus set single_backward x1 x1 /dev/mmcblk1
Changing ext_csd[BOOT_BUS_CONDITIONS] from 0x00 to 0x00
root@ATK-DLMP257:~# dd if=tf-a-stm32mp257d-atk-ddr-2GB-optee-emmc.stm32 of=/dev/mmcblk1boot0 c
onv=fdatasync
405+1 records in
405+1 records out
207871 bytes (208 kB, 203 KiB) copied, 0.01603 s, 13.0 MB/s
root@ATK-DLMP257:~# mmc bootpart enable 1 1 /dev/mmcblk1
root@ATK-DLMP257:~# echo 1 > /sys/class/block/mmcblk1boot0/force_ro
```

Figure 3.1-3 Update TF-A

Restart the board and view the TF-A information:



```
INFO:    PSCI Power Domain Map:
INFO:      Domain Node : Level 4, parent_node 4294967295, State ON (0x0)
INFO:      Domain Node : Level 3, parent_node 0, State ON (0x0)
INFO:      Domain Node : Level 2, parent_node 1, State ON (0x0)
INFO:      Domain Node : Level 1, parent_node 2, State ON (0x0)
INFO:      CPU Node : MPID 0x0, parent_node 3, State ON (0x0)
INFO:      CPU Node : MPID 0x1, parent_node 3, State ON (0x0)
NOTICE:  CPU: STM32MP257DAK Rev.Y
NOTICE:  Model: ALIENTEK STM32MP257 Evaluation Board
INFO:    Reset reason (0x2044):
INFO:      System reset (SYSRST) by A35
INFO:    PMIC2 version = 0x11
INFO:    PMIC2 product ID = 0x20
INFO:    FCONF: Reading TB_FW firmware configuration file from: 0xe011000
INFO:    FCONF: Reading firmware configuration information for: stm32mp_fuse
INFO:    FCONF: Reading firmware configuration information for: stm32mp_io
INFO:    Using EMMC
INFO:      Instance 2
INFO:    Boot used partition fsbl1
NOTICE:  BL2: v2.10-stm32mp2-r1.0(debug):abf7a33(abf7a332)
NOTICE:  BL2: Built : 02:45:54, Feb  7 2025
INFO:    BL2: Loading image id 26
INFO:    Loading image id=26 at address 0xe041000
```

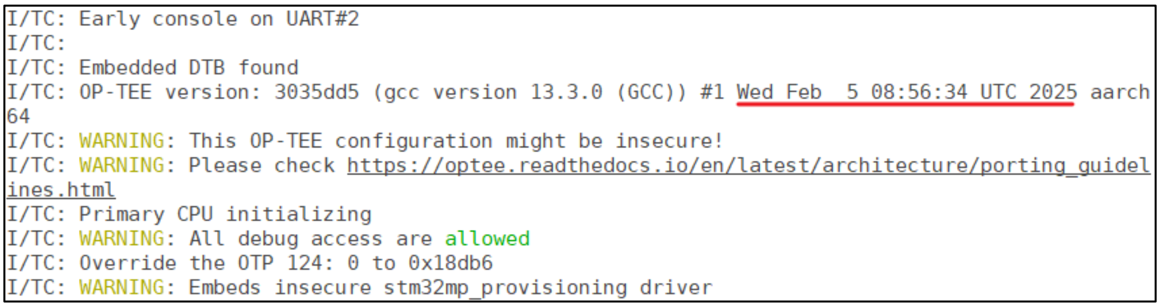
Figure 3.1-4 TF-A update date

You can see that the TF-A date has been updated and the TF-A burn is complete.

Chapter 4. Update firmware uboot, optee

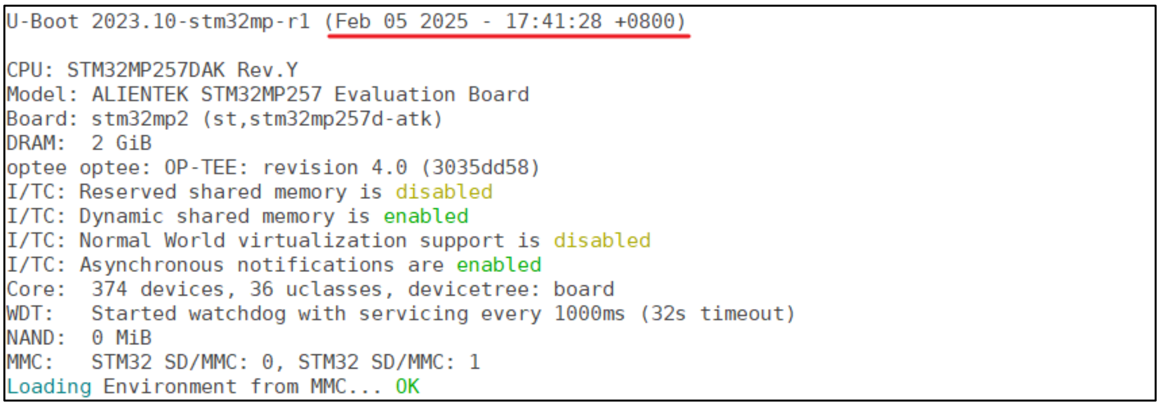
4.1 Updating with dd instructions (EMMC)

Before updating, you can first check the generation time of optee and uboot of the current development board, so as to facilitate the comparison after subsequent updates and confirm the success of burning.



```
I/TC: Early console on UART#2
I/TC:
I/TC: Embedded DTB found
I/TC: OP-TEE version: 3035dd5 (gcc version 13.3.0 (GCC)) #1 Wed Feb  5 08:56:34 UTC 2025 aarch
64
I/TC: WARNING: This OP-TEE configuration might be insecure!
I/TC: WARNING: Please check https://optee.readthedocs.io/en/latest/architecture/porting_guidel
ines.html
I/TC: Primary CPU initializing
I/TC: WARNING: All debug access are allowed
I/TC: Override the OTP 124: 0 to 0x18db6
I/TC: WARNING: Embeds insecure stm32mp_provisioning driver
```

Figure 4.1-1 optee generates time examples

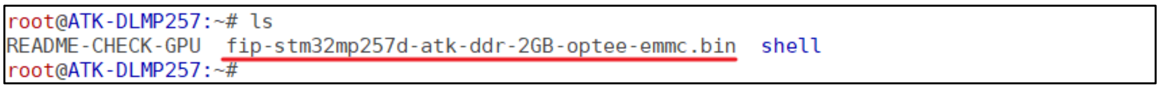


```
U-Boot 2023.10-stm32mp-r1 (Feb 05 2025 - 17:41:28 +0800)

CPU: STM32MP257DAK Rev.Y
Model: ALIENTEK STM32MP257 Evaluation Board
Board: stm32mp2 (st,stm32mp257d-atk)
DRAM:  2 GiB
optee optee: OP-TEE: revision 4.0 (3035dd58)
I/TC: Reserved shared memory is disabled
I/TC: Dynamic shared memory is enabled
I/TC: Normal World virtualization support is disabled
I/TC: Asynchronous notifications are enabled
Core:  374 devices, 36 uclasses, devicetree: board
WDT:   Started watchdog with servicing every 1000ms (32s timeout)
NAND:  0 MiB
MMC:   STM32 SD/MMC: 0, STM32 SD/MMC: 1
Loading Environment from MMC... OK
```

Figure 4.1-2 Example of uboot generation time

Assuming you have compiled the tf-a, optee, uboot source code and packaged the fip.bin file, you can copy the fip.bin file to the development board home directory to prepare for fip.bin burning.

Here the author takes the 2GB version as an example, using fip-stm32mp257d-atk-ddr-2GB-optee-emmc.bin

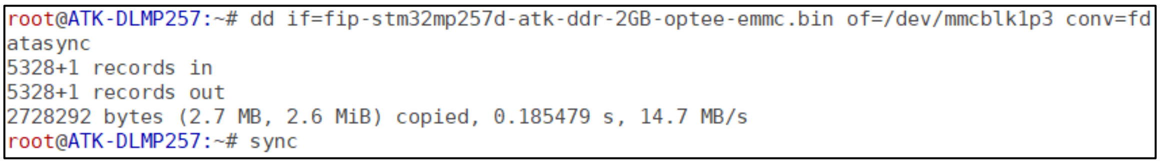


```
root@ATK-DLMP257:~# ls
README-CHECK-GPU  fip-stm32mp257d-atk-ddr-2GB-optee-emmc.bin  shell
root@ATK-DLMP257:~#
```

Figure 4.1-3 Copy fip.bin to the board

Execute the following command to burn.

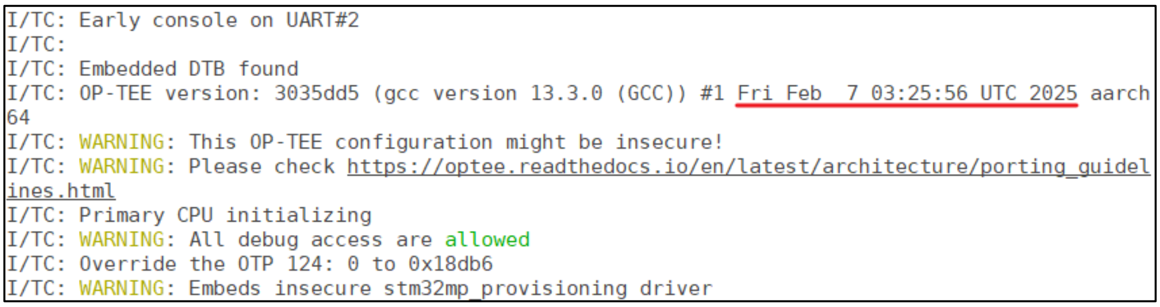
```
dd if=fip-stm32mp257d-atk-ddr-2GB-optee-emmc.bin of=/dev/mmcblk1p3 conv=fdatasync
sync
```



```
root@ATK-DLMP257:~# dd if=fip-stm32mp257d-atk-ddr-2GB-optee-emmc.bin of=/dev/mmcblk1p3 conv=fd
atasync
5328+1 records in
5328+1 records out
2728292 bytes (2.7 MB, 2.6 MiB) copied, 0.185479 s, 14.7 MB/s
root@ATK-DLMP257:~# sync
```

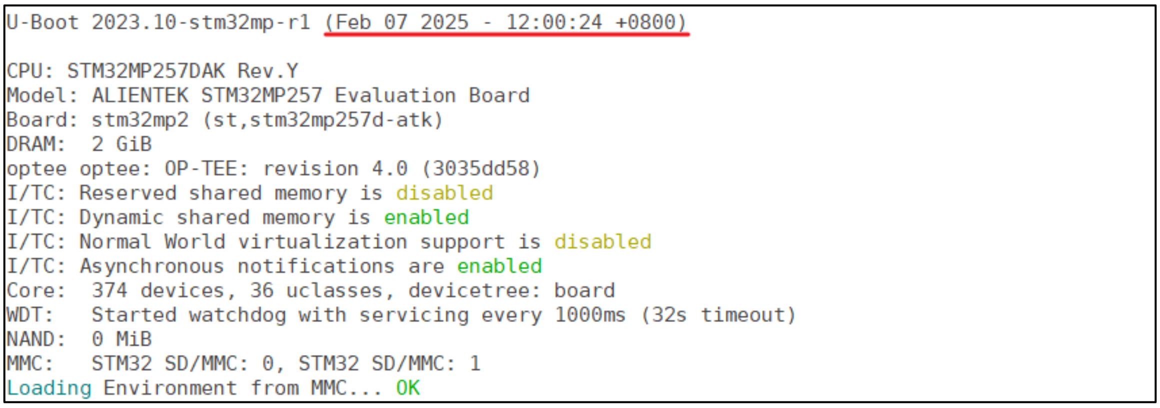
Figure 4.1-4 Burn fip.bin to eMMC

After burning, restart the development board to view the printing information of optee and uboot. The information here is as follows:



```
I/TC: Early console on UART#2
I/TC: 
I/TC: Embedded DTB found
I/TC: OP-TEE version: 3035dd5 (gcc version 13.3.0 (GCC)) #1 Fri Feb  7 03:25:56 UTC 2025 aarch
64
I/TC: WARNING: This OP-TEE configuration might be insecure!
I/TC: WARNING: Please check https://optee.readthedocs.io/en/latest/architecture/porting_guidel
ines.html
I/TC: Primary CPU initializing
I/TC: WARNING: All debug access are allowed
I/TC: Override the OTP 124: 0 to 0x18db6
I/TC: WARNING: Embeds insecure stm32mp_provisioning driver
```

Figure 4.1-5 View the optee information after burning



```
U-Boot 2023.10-stm32mp-r1 (Feb 07 2025 - 12:00:24 +0800)

CPU: STM32MP257DAK Rev.Y
Model: ALIENTEK STM32MP257 Evaluation Board
Board: stm32mp2 (st,stm32mp257d-atk)
DRAM:  2 GiB
optee optee: OP-TEE: revision 4.0 (3035dd58)
I/TC: Reserved shared memory is disabled
I/TC: Dynamic shared memory is enabled
I/TC: Normal World virtualization support is disabled
I/TC: Asynchronous notifications are enabled
Core:  374 devices, 36 uclasses, devicetree: board
WDT:   Started watchdog with servicing every 1000ms (32s timeout)
NAND:  0 MiB
MMC:   STM32 SD/MMC: 0, STM32 SD/MMC: 1
Loading Environment from MMC... OK
```

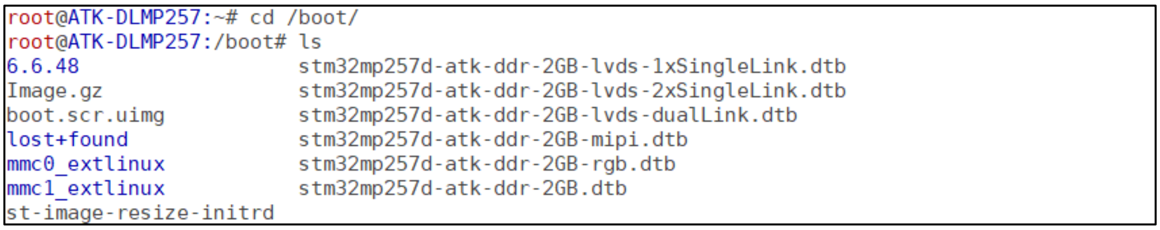
Figure 4.1-6 View the uboot information after burning

Comparing the previous information of optee and uboot, it can be seen that the generation time of optee and uboot has changed, and the time is also the corresponding time in the author's compilation environment, and the burning is completed.

Chapter 5. Updates the kernel, kernel modules, and device tree

5.1 Debugging and validation phase

In the debugging verification phase, we just need to replace the image file to the corresponding path. Kernel and device tree files are in /boot of the system.

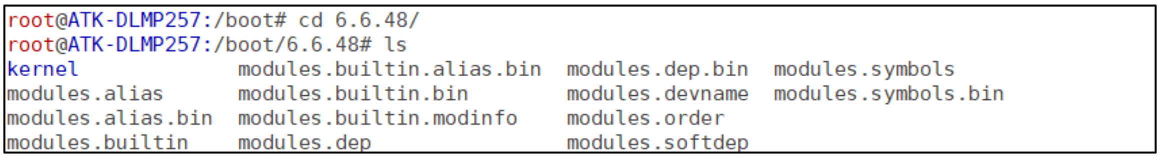


```
root@ATK-DLMP257:~# cd /boot/
root@ATK-DLMP257:/boot# ls
6.6.48                  stm32mp257d-atk-ddr-2GB-lvds-1xSingleLink.dtb
Image.gz                stm32mp257d-atk-ddr-2GB-lvds-2xSingleLink.dtb
boot.scr.uimg           stm32mp257d-atk-ddr-2GB-lvds-dualLink.dtb
lost+found              stm32mp257d-atk-ddr-2GB-mipi.dtb
mmc0_extlinux           stm32mp257d-atk-ddr-2GB-rgb.dtb
mmc1_extlinux           stm32mp257d-atk-ddr-2GB.dtb
st-image-resize-initrd
```

Figure 5.1-1 Kernel, device tree path

The factory system has multiple device trees compatible with multiple screens of ALIENTEK, taking the 2GB version as an example, stm32mp257d-atk-ddr-2GB.dtb is the base tree, and other device tree files contain this device tree. When users need to modify the device tree, they need to modify the device tree file from stm32mp257d-atk-ddr-2GB.dts, and update the device tree file to the development board for use.

The driver module path is located at /boot/6.6.48 on the system (linked to /lib/modules/6.6.48), where you can place the driver file and execute the driver for validation once it has been compiled.



```
root@ATK-DLMP257:/boot# cd 6.6.48/
root@ATK-DLMP257:/boot/6.6.48# ls
kernel             modules.builtin.alias.bin  modules.dep.bin   modules.symbols
modules.alias      modules.builtin.bin        modules.devname   modules.symbols.bin
modules.alias.bin  modules.builtin.modinfo    modules.order
modules.builtin    modules.dep                modules.softdep
```

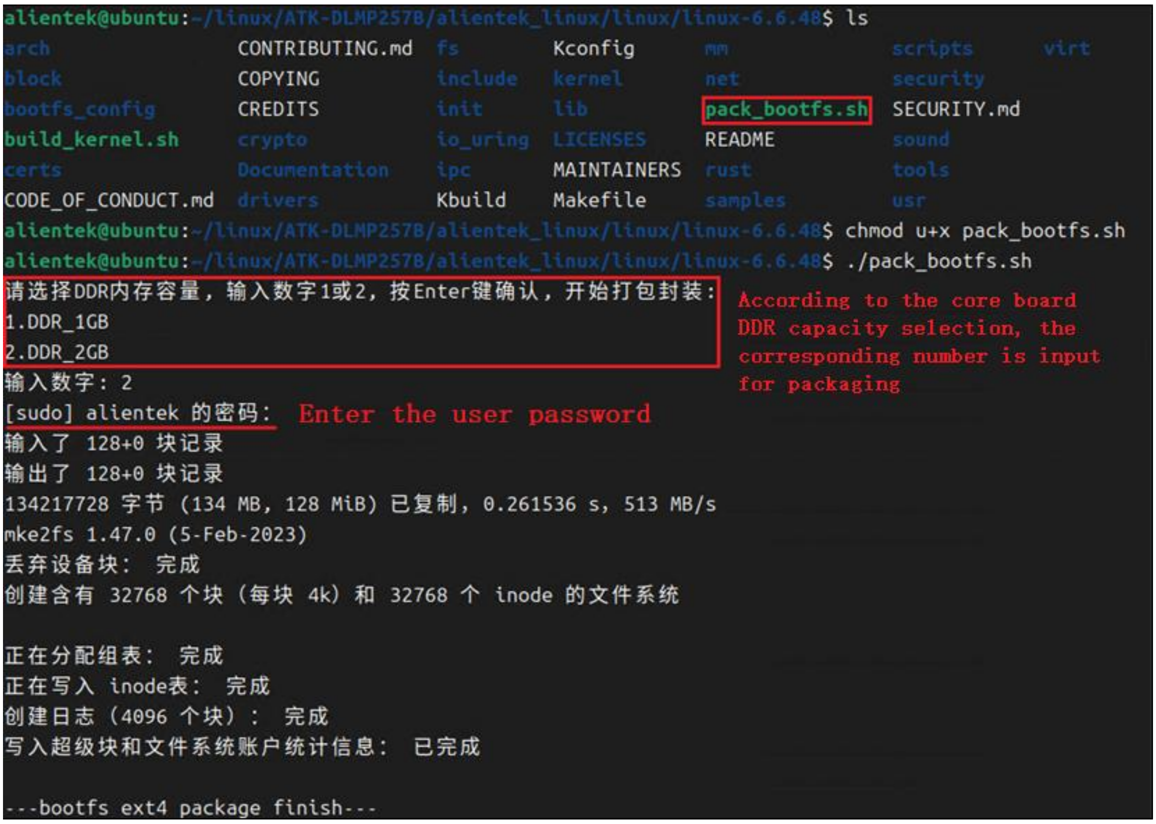
Figure 5.1-2 Driver module path

5.2 Packaging images

The 2GB version, for example, can be packaged as a bootfs-2GB.ext4 image after finally confirming the mirror functionality. There is also a pack_bootfs.sh script reserved in the factory kernel sources to package the image. After executing the build_kernel.sh script, the pack_bootfs.sh script can be used to generate the corresponding bootfs image.

```
Chmod u+x pack_bootfs.sh
./pack_bootfs.sh
```

To run this script, you need to use the sudo permission. Enter the user password.



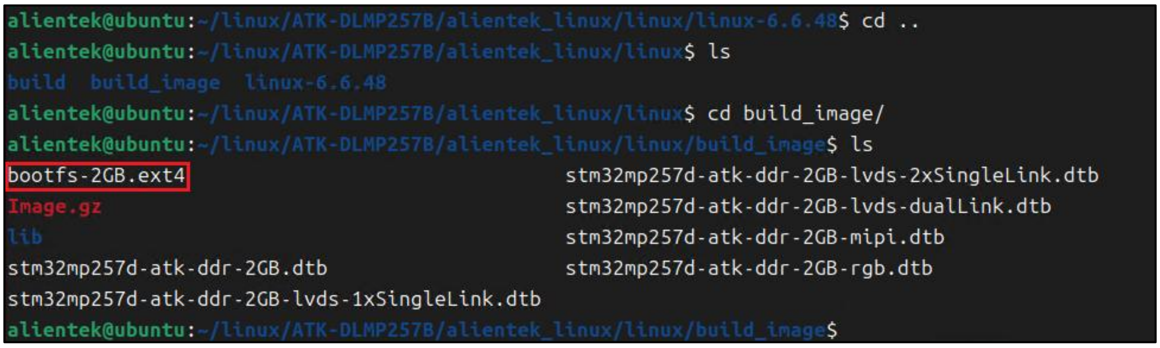
```
alientek@ubuntu:~/linux/ATK-DLMP257B/alientek_linux/linux/linux-6.6.48$ ls
arch                CONTRIBUTING.md  fs         Kconfig      mm              scripts        virt
block               COPYING          include    kernel       net             security
bootfs_config       CREDITS          init       lib          pack_bootfs.sh  SECURITY.md
build_kernel.sh     crypto           io_uring   LICENSES     README          sound
certs               Documentation    ipc        MAINTAINERS  rust            tools
CODE_OF_CONDUCT.md  drivers          Kbuild     Makefile     samples         usr
alientek@ubuntu:~/linux/ATK-DLMP257B/alientek_linux/linux/linux-6.6.48$ chmod u+x pack_bootfs.sh
alientek@ubuntu:~/linux/ATK-DLMP257B/alientek_linux/linux/linux-6.6.48$ ./pack_bootfs.sh
请选择DDR内存容量，输入数字1或2，按Enter键确认，开始打包封装：
1.DDR_1GB
2.DDR_2GB
输入数字：2
[sudo] alientek 的密码：
输入了 128+0 块记录
输出了 128+0 块记录
134217728 字节 (134 MB, 128 MiB) 已复制，0.261536 s，513 MB/s
mke2fs 1.47.0 (5-Feb-2023)
丢弃设备块： 完成
创建含有 32768 个块（每块 4k）和 32768 个 inode 的文件系统

正在分配组表： 完成
正在写入 inode表： 完成
创建日志（4096 个块）： 完成
写入超级块和文件系统账户统计信息： 已完成

---bootfs ext4 package finish---
```

According to the core board DDR capacity selection, the corresponding number is input for packaging

Enter the user password

Figure 5.2-1 Execute the pack_bootfs.sh script

Once the package is complete, a bootfs image is generated under the build_image directory in the parent directory.



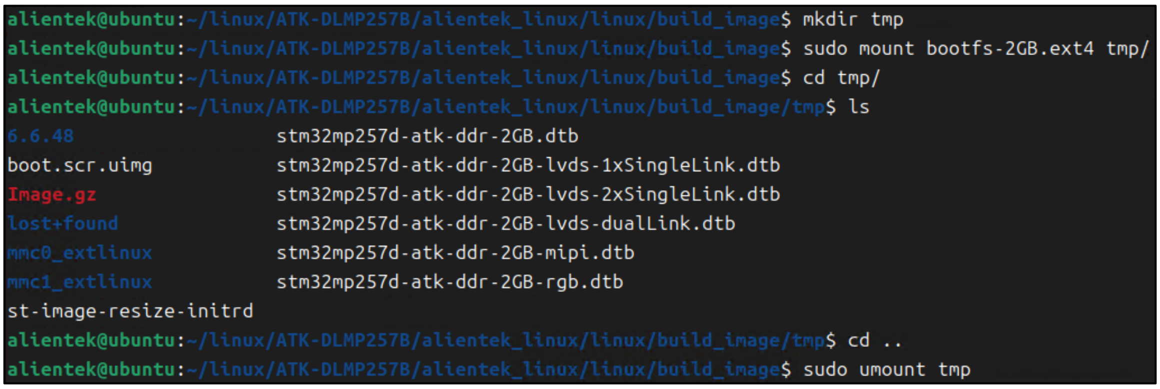
```
alientek@ubuntu:~/linux/ATK-DLMP257B/alientek_linux/linux/linux-6.6.48$ cd ..
alientek@ubuntu:~/linux/ATK-DLMP257B/alientek_linux/linux$ ls
build  build_image  linux-6.6.48
alientek@ubuntu:~/linux/ATK-DLMP257B/alientek_linux/linux$ cd build_image/
alientek@ubuntu:~/linux/ATK-DLMP257B/alientek_linux/linux/build_image$ ls
bootfs-2GB.ext4                                stm32mp257d-atk-ddr-2GB-lvds-2xSingleLink.dtb
Image.gz                                       stm32mp257d-atk-ddr-2GB-lvds-dualLink.dtb
lib                                            stm32mp257d-atk-ddr-2GB-mipi.dtb
stm32mp257d-atk-ddr-2GB.dtb                    stm32mp257d-atk-ddr-2GB-rgb.dtb
stm32mp257d-atk-ddr-2GB-lvds-1xSingleLink.dtb
alientek@ubuntu:~/linux/ATK-DLMP257B/alientek_linux/linux/build_image$
```

Figure 5.2-2 The corresponding bootfs image is generated under build_image

This Image is a bootfs file that contains the kernel image image.gz, kernel module lib, device tree dtb file, and added boot related files such as mmc_extlinux, boot.scr.uimg, st-image-resize-initrd. It can be used for STM32CubeProgrammer host computer burning or mass production burning.

If changes are needed, the bootfs image can be mounted using the mount command to replace the files. The reference instructions are as follows:

```
mkdir tmp // Create a temporary mount directory
sudo mount bootfs-2GB.ext4 tmp // Mount the ext4 image to the tmp directory
cd tmp // enters the mount directory, which can be modified, copied, and so on
cd .. // Exit the mount directory
sudo umont tmp // Unmount image
```



```
alientek@ubuntu:~/linux/ATK-DLMP257B/alientek_linux/linux/build_image$ mkdir tmp
alientek@ubuntu:~/linux/ATK-DLMP257B/alientek_linux/linux/build_image$ sudo mount bootfs-2GB.ext4 tmp/
alientek@ubuntu:~/linux/ATK-DLMP257B/alientek_linux/linux/build_image$ cd tmp/
alientek@ubuntu:~/linux/ATK-DLMP257B/alientek_linux/linux/build_image/tmp$ ls
6.6.48                  stm32mp257d-atk-ddr-2GB.dtb
boot.scr.uimg           stm32mp257d-atk-ddr-2GB-lvds-1xSingleLink.dtb
Image.gz                stm32mp257d-atk-ddr-2GB-lvds-2xSingleLink.dtb
lost+found              stm32mp257d-atk-ddr-2GB-lvds-dualLink.dtb
mmc0_extlinux           stm32mp257d-atk-ddr-2GB-mipi.dtb
mmc1_extlinux           stm32mp257d-atk-ddr-2GB-rgb.dtb
st-image-resize-initrd
alientek@ubuntu:~/linux/ATK-DLMP257B/alientek_linux/linux/build_image/tmp$ cd ..
alientek@ubuntu:~/linux/ATK-DLMP257B/alientek_linux/linux/build_image$ sudo umount tmp
```

Figure 5.2-3 Image mount \ unmount

Chapter 6. New filesystem

6.1 .Packaging the filesystem

After file transfer and functional verification on the development board, the file system needs to be packaged for burning before final production. In general, users perform functional verification on the development board. When it is necessary to package the file system running on the EMMC of the development board, many problems may be encountered when operating directly on the running system, such as the consistency of the file system and the risk of data corruption. To safely and efficiently package an entire filesystem, several approaches can be taken:

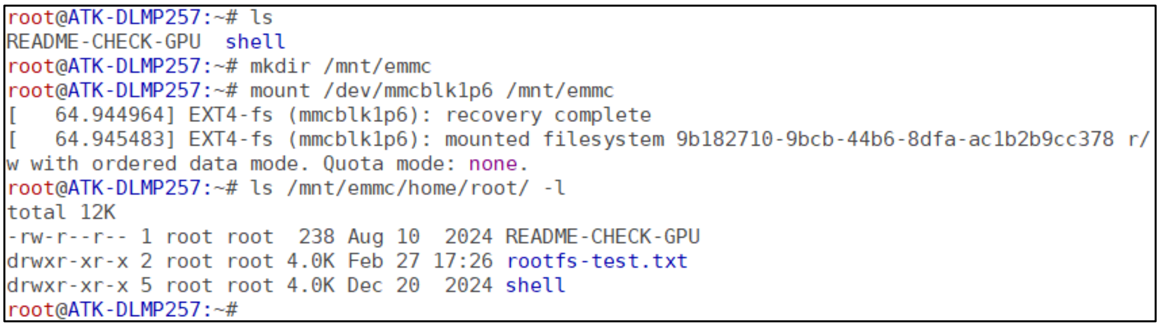
6.1.1 Launching and packaging the EMMC filesystem from a TF card

Make A system boot card of the ATK-DLMP257B development board according to the method of "Making TF system boot Card" related chapter of the STM32MP257 development board (Disk A) - Basic Information \10_ user_manual \ [ALIENTEK] ATK-DLMP257B Quick Test Manual V1.0. Note that the TF card size should be greater than 16G, otherwise the lack of space will lead to packing failure.

Connect the startup card into the TF card slot of the development board, and set the dial switch to 1000 (SD card startup mode).

After booting the system with the TF card, mount the EMMC device /dev/mmcblk1p6

```
mkdir /mnt/emmc
mount /dev/mmcblk1p6 /mnt/emmc
ls /mnt/emmc/home/root/ -l
```



```
root@ATK-DLMP257:~# ls
README-CHECK-GPU  shell
root@ATK-DLMP257:~# mkdir /mnt/emmc
root@ATK-DLMP257:~# mount /dev/mmcblk1p6 /mnt/emmc
[   64.944964] EXT4-fs (mmcblk1p6): recovery complete
[   64.945483] EXT4-fs (mmcblk1p6): mounted filesystem 9b182710-9bcb-44b6-8dfa-ac1b2b9cc378 r/
w with ordered data mode. Quota mode: none.
root@ATK-DLMP257:~# ls /mnt/emmc/home/root/ -l
total 12K
-rw-r--r-- 1 root root  238 Aug 10  2024 README-CHECK-GPU
drwxr-xr-x 2 root root 4.0K Feb 27 17:26 rootfs-test.txt
drwxr-xr-x 5 root root 4.0K Dec 20  2024 shell
root@ATK-DLMP257:~#
```

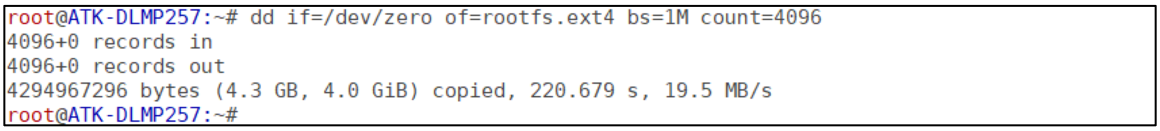
Figure 6.1-1 Mount the EMMC device

Package as rootfs.ext4 image: Use the dd, mkfs.ext4, and mount commands to create a 4GB rootfs.ext4 image, depending on the size of your project, using 4GB as an example:

1. Create an empty 4GB file

```
dd if=/dev/zero of=rootfs.ext4 bs=1M count=4096
```

- of=rootfs.ext4: Output filename, which is the generated rootfs image file.
- bs=1M: The block size is 1MB.
- count=4096: Write a total of 4096MB (4GB).



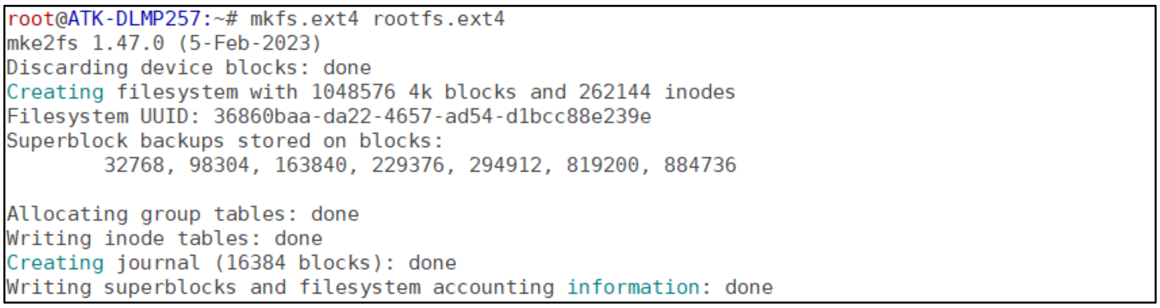
```
root@ATK-DLMP257:~# dd if=/dev/zero of=rootfs.ext4 bs=1M count=4096
4096+0 records in
4096+0 records out
4294967296 bytes (4.3 GB, 4.0 GiB) copied, 220.679 s, 19.5 MB/s
root@ATK-DLMP257:~#
```

Figure 6.1-2 Create rootfs.ext4

2.Formatting to the ext4 filesystem

```
mkfs.ext4 rootfs.ext4
```

This creates an ext4 filesystem on the rootfs.ext4 file.



```
root@ATK-DLMP257:~# mkfs.ext4 rootfs.ext4
mke2fs 1.47.0 (5-Feb-2023)
Discarding device blocks: done
Creating filesystem with 1048576 4k blocks and 262144 inodes
Filesystem UUID: 36860baa-da22-4657-ad54-d1bcc88e239e
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done
```

Figure 6.1-3 Format rootfs.ext4 in ext4 format

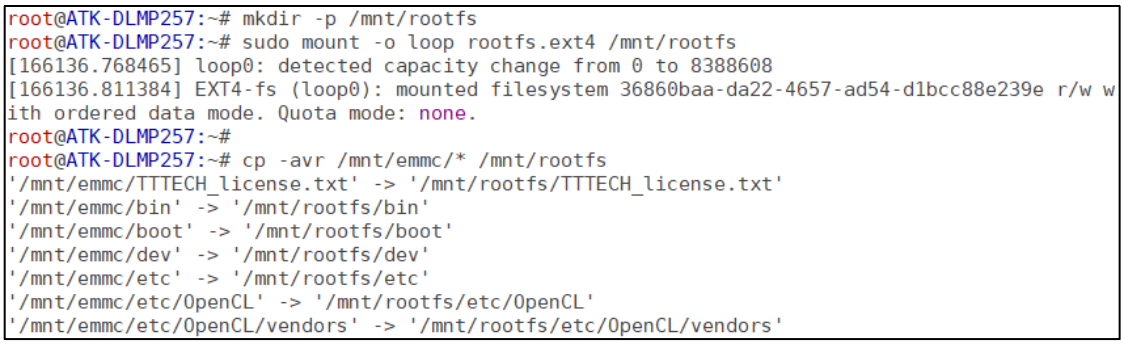
3.Mount and add the file

Now we need to add the \ copy content to the filesystem by creating a mount point:

```
mkdir -p /mnt/rootfs
sudo mount -o loop rootfs.ext4 /mnt/rootfs
```

Copy the contents of /mnt/emmc to /mnt/rootfs

```
cp -avr /mnt/emmc/* /mnt/rootfs
```

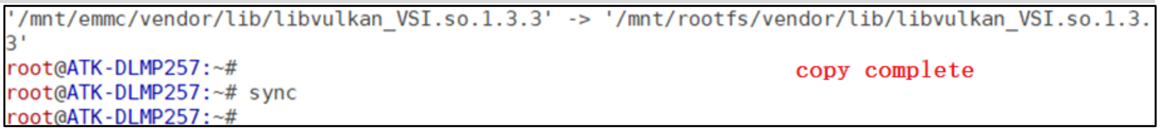


```
root@ATK-DLMP257:~# mkdir -p /mnt/rootfs
root@ATK-DLMP257:~# sudo mount -o loop rootfs.ext4 /mnt/rootfs
[166136.768465] loop0: detected capacity change from 0 to 8388608
[166136.811384] EXT4-fs (loop0): mounted filesystem 36860baa-da22-4657-ad54-d1bcc88e239e r/w w
ith ordered data mode. Quota mode: none.
root@ATK-DLMP257:~#
root@ATK-DLMP257:~# cp -avr /mnt/emmc/* /mnt/rootfs
'/mnt/emmc/TTTECH_license.txt' -> '/mnt/rootfs/TTTECH_license.txt'
'/mnt/emmc/bin' -> '/mnt/rootfs/bin'
'/mnt/emmc/boot' -> '/mnt/rootfs/boot'
'/mnt/emmc/dev' -> '/mnt/rootfs/dev'
'/mnt/emmc/etc' -> '/mnt/rootfs/etc'
'/mnt/emmc/etc/OpenCL' -> '/mnt/rootfs/etc/OpenCL'
'/mnt/emmc/etc/OpenCL/vendors' -> '/mnt/rootfs/etc/OpenCL/vendors'
```

Figure 6.1-4 Copy the emmc contents to the rootfs image

Once the copy is complete, the sync command is used to synchronize the cache to ensure that all files have been copied

```
sync
```



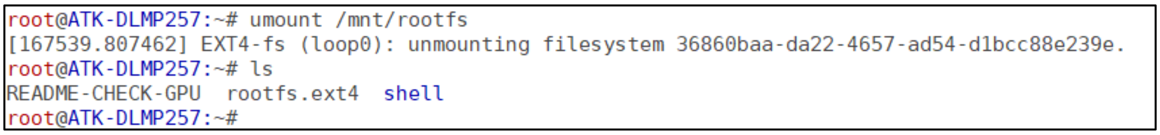
```
'/mnt/emmc/vendor/lib/libvulkan_VSI.so.1.3.3' -> '/mnt/rootfs/vendor/lib/libvulkan_VSI.so.1.3.
3'
root@ATK-DLMP257:~#                                          copy complete
root@ATK-DLMP257:~# sync
root@ATK-DLMP257:~#
```

Figure 6.1-5 The copy is complete and the cache is synchronized

4. Uninstall the image

Once the copy is complete, uninstall the image:

```
umount /mnt/rootfs
```



```
root@ATK-DLMP257:~# umount /mnt/rootfs
[167539.807462] EXT4-fs (loop0): unmounting filesystem 36860baa-da22-4657-ad54-d1bcc88e239e.
root@ATK-DLMP257:~# ls
README-CHECK-GPU  rootfs.ext4  shell
root@ATK-DLMP257:~#
```

Figure 6.1-6 Uninstall an image

Then we can remove the TF card, connect the TF card to the ubuntu system, copy the rootfs.ext4 image, and use STM32CubeProgrammer to burn.