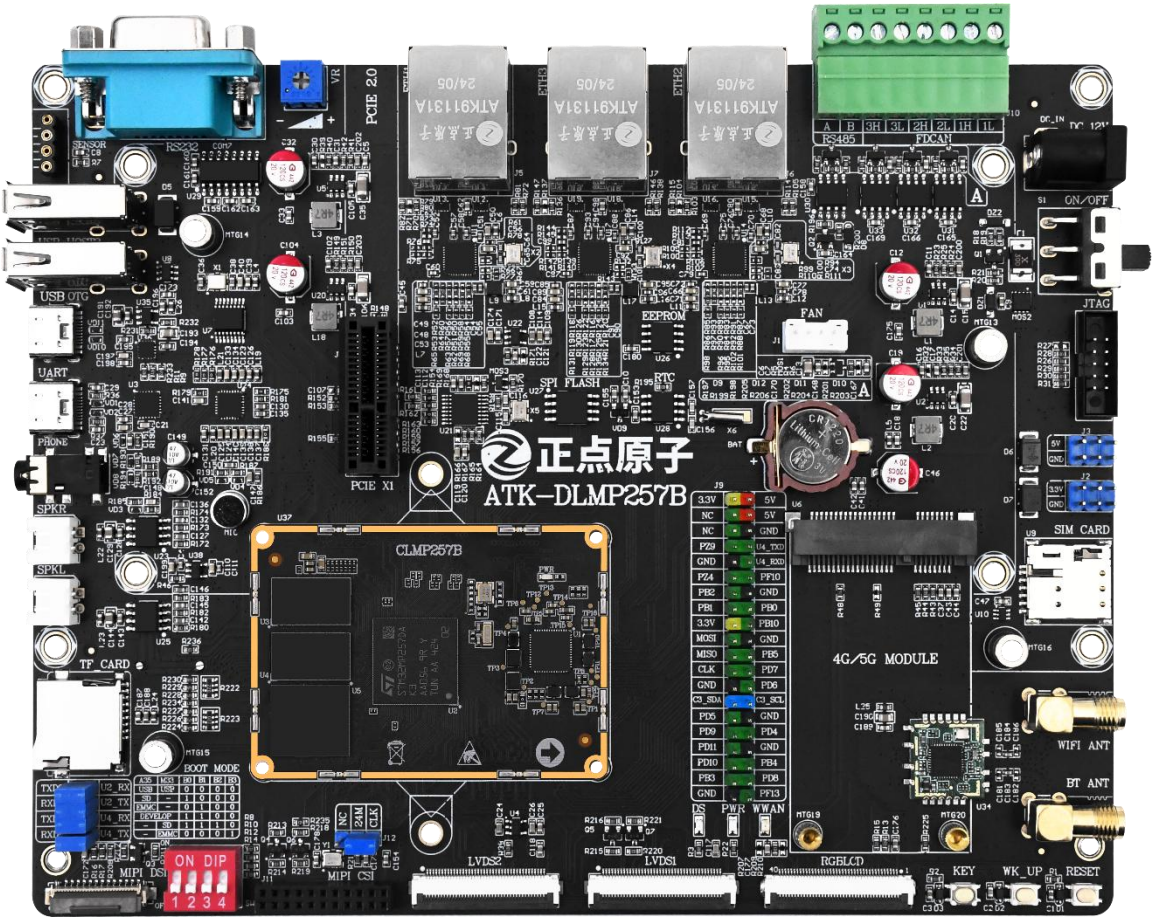
ATK-DLMP257B

U-Boot command Reference manual

V1.0



ALIENTEK

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.

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. The use of U-boot

ATK-DLMP257B development board network disk data has provided a configured factory system U-Boot, the development board has written U-boot by default, we can learn to use the U-Boot command after starting the development board.

1.1 Introduction to U-Boot

Linux system needs to boot through the bootloader program boot, that is to say after the chip is powered on, run a bootloader program. This bootloader program will first initialize the DDR and other peripherals, and then copy the Linux kernel from flash(NAND, NOR FLASH, SD, EMMC, etc.) to the DDR, and finally boot the Linux kernel. Of course, the actual job of bootloader is more complicated, but its main job is to boot the Linux kernel. bootloader is related to the Linux kernel in the same way as BIOS is related to Windows on a PC. bootloader is the BIOS. Fortunately, there are many bootloaders available, such as U-Boot, vivi, RedBoot, etc., among which U-Boot is the most widely used. For the convenience of writing, this book will write U-Boot as uboot.

The full name of uboot is Universal Boot Loader, uboot is an open source software under GPL license, uboot is a bare-metal code, can be seen as a bare-metal synthesis routine. Now uboot already supports LCD, network, USB and other advanced functions.

In this chapter, we mainly use the factory system uboot to use the uboot command, which can directly start the development board for learning. uboot compilation and burning written in the network disk data **10_user_manual \[ALIENTEK]ATK-DLMP257B Factory System Source Code Use Guide V1.0** is introduced, here is not detailed expansion.

1.2 U-Boot command usage

Start the development board, and you can see the information related to uboot startup in the serial port print information interface. We need to press the Enter key and stay in the uboot command line before the uboot countdown is over, so that we can operate the uboot command in the uboot command line.

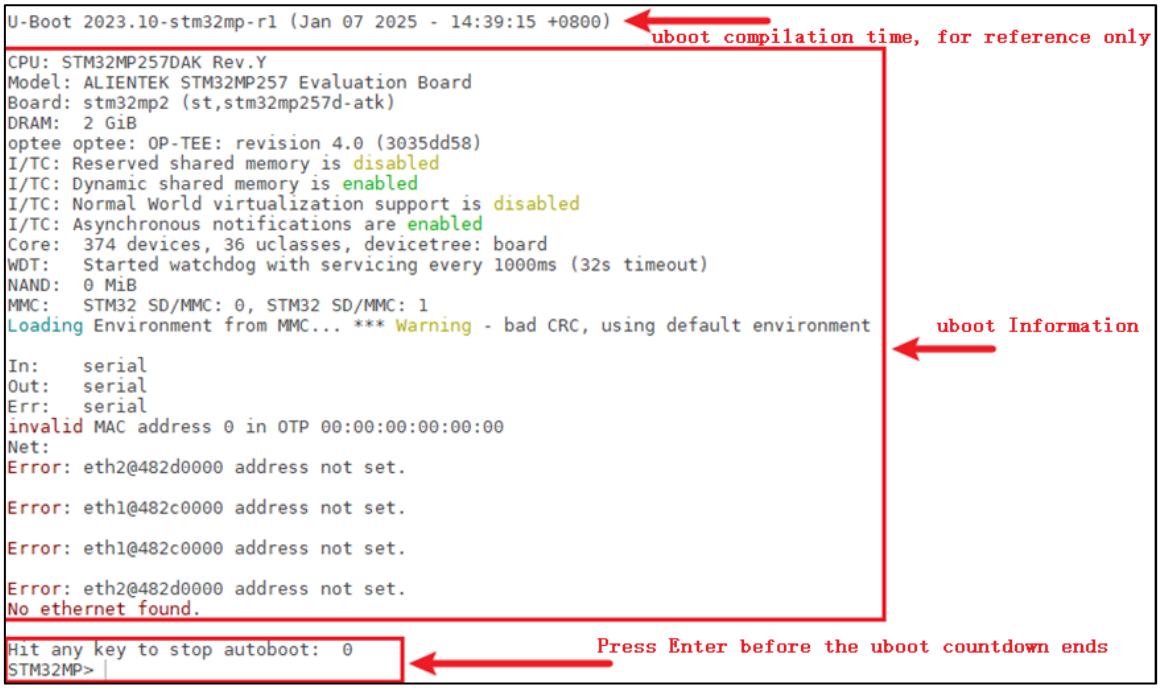


Figure 1.2-1 uboot startup information

Enter command-line mode in uboot and type "help" or "?" Then press Enter to see the commands supported by uboot, as shown in the following screenshot:

```
Hit any key to stop autoboot:  0
STM32MP> ?
?          - alias for 'help'
adtimg     - manipulate dtb/dtbo Android image
base       - print or set address offset
bdinfo     - print Board Info structure
blkcache   - block cache diagnostics and control
bmp        - manipulate BMP image data
boot       - boot default, i.e., run 'bootcmd'
bootd      - boot default, i.e., run 'bootcmd'
bootflow   - Boot flows
booti      - boot Linux kernel 'Image' format from memory
```

Figure 1.2-2 uboot command list (partial)

This is just a subset of the commands in uboot. The list of commands is subject to the actual situation. The commands in the figure are not all supported by uboot. It is said that uboot is configurable and can enable whatever command is needed. So the commands in the figure are the ones enabled by uboot provided by ALIENTEK. There are many more commands supported by uboot, and you can also customize commands in uboot. Each of these commands is followed by a command description that describes what the command does, but how does it work? We type "help(or? For example, to see the usage of the command "bootm", we can enter the following command to see the usage of the command "bootm" :

```
? bootm or help bootm
```

The result is shown in the figure:

```
STM32MP> ? bootm
bootm - boot application image from memory

Usage:
bootm [addr [arg ...]]
    - boot application image stored in memory
        passing arguments 'arg ...'; when booting a Linux kernel,
        'arg' can be the address of an initrd image
        When booting a Linux kernel which requires a flat device-tree
        a third argument is required which is the address of the
        device-tree blob. To boot that kernel without an initrd image,
        use a '-' for the second argument. If you do not pass a third
        a bd_info struct will be passed instead

For the new multi component uImage format (FIT) addresses
        must be extended to include component or configuration unit name:
        addr:<subimg_uname> - direct component image specification
        addr#<conf_uname>   - configuration specification
        Use iminfo command to get the list of existing component
        images and configurations.

Sub-commands to do part of the bootm sequence.  The sub-commands must be
issued in the order below (it's ok to not issue all sub-commands):
        start [addr [arg ...]]
        loados  - load OS image
        ramdisk - relocate initrd, set env initrd_start/initrd_end
        fdt     - relocate flat device tree
        cmdline - OS specific command line processing/setup
        bdt     - OS specific bd_info processing
        prep    - OS specific prep before relocation or go
        go      - start OS
STM32MP>
```

Figure 1.2-3 Instructions for using the boot command

The figure lists the details of the "bootm" command. Other commands can also be used to query the specific usage method. Let's look at some common uboot commands.

1.2.1 Querying commands

Three commands are commonly used to query information: bdinfo, printenv, and version. First, let's take a look at the bdinfo command.This command is used to view the board information, simply enter "bdinfo", the result is shown in the figure:

```
STM32MP> bdinfo
boot_params = 0x0000000000000000
DRAM bank   = 0x0000000000000000
-> start    = 0x0000000080000000
-> size     = 0x0000000080000000
flashstart  = 0x0000000000000000
flashsize   = 0x0000000000000000
flashoffset = 0x0000000000000000
baudrate    = 115200 bps
relocaddr   = 0x00000000fa67b000
reloc off   = 0x000000007667b000
Build       = 64-bit

Error: eth1@482c0000 address not set.

Error: eth2@482d0000 address not set.

Error: eth1@482c0000 address not set.

Error: eth1@482c0000 address not set.

Error: eth2@482d0000 address not set.

Error: eth1@482c0000 address not set.
current eth = unknown
eth-1addr   = (not set)
IP addr     = <NULL>
fdt_blob    = 0x00000000f8661030
new_fdt     = 0x00000000f8661030
fdt_size    = 0x0000000000017d20
```

Figure 1.2-4 bdinfo command

From the figure, we can see the information such as the start address and size of DRAM, BOOT parameter saving start address, baud rate, and sp(stack pointer) start address. The command "printenv" is used to output the information of environment variables. uboot also supports the TAB key autocompletion function. Type "print" and then press TAB to autocomplete the command. Just typing "print" works, because in the whole uboot command, only printenv is prefixed with "print", so when you type print, it's just printenv. Type "print" and press Returt.The environment variable looks like this:

```
STM32MP> printenv
api_address=f86bb630
arch=arm
autoload=0
baudrate=115200
board=stm32mp2
board_name=stm32mp257d-atk
boot_a_script=load ${devtype} ${devnum}:${distro_bootpart} ${scriptaddr} ${prefix}${script}; s
ource ${scriptaddr}
boot_device=mmc
boot_extlinux=sysboot ${devtype} ${devnum}:${distro_bootpart} any ${scriptaddr} ${prefix}${boo
t_syslinux_conf}
boot_instance=1
boot_net_usb_start=true
boot_prefixes=/ /boot/
boot_script_dhcp=boot.scr.uimg
boot_scripts=boot.scr.uimg boot.scr
boot_syslinux_conf=extlinux/extlinux.conf
boot_targets=mmc1 ubifs0 mmc0 mmc2 usb0 pxe
bootcmd=run bootcmd_stm32mp
bootcmd_mmc0=devnum=0; run mmc_boot
bootcmd_mmc1=devnum=1; run mmc_boot
bootcmd_mmc2=devnum=2; run mmc_boot
bootcmd_pxe=run boot_net_usb_start; dhcp; if pxe get; then pxe boot; fi
bootcmd_stm32mp=echo "Boot over ${boot_device}${boot_instance}!";if test ${boot_device} = seri
al || test ${boot_device} = usb;then stm32prog ${boot_device} ${boot_instance}; else run env_c
heck;if test ${boot_device} = mmc;then env set boot_targets "mmc${boot_instance}"; fi;if test
${boot_device} = nand || test ${boot_device} = spi-nand ;then env set boot_targets ubifs0 mmc0
; fi;if test ${boot_device} = nor;then env set boot_targets mmc0; fi;run distro_bootcmd;fi;
bootcmd_ubifs0=bootubipart=UBI; bootubivol=boot; bootubioff=; run ubifs_boot
bootcmd_usb0=devnum=0; run usb_boot
```

Figure 1.2-5 printenv commands partial results

The preceding figure is just a sampling of the printenv command.The MP257 family has many environment variables, such as baudrate, board, board_name, bootcmd, bootdelay, and so on. All environment variables in uboot are strings, and since they're called environment variables, they act like "variables." For example, the bootdelay environment variable represents the uboot bootdelay time, and the default bootdelay=1 is 1 second. The 1-second countdown is defined by bootdelay; changing bootdelay to 5 will result in a countdown of 5 seconds. The environment variables in uboot can be mutated.There are commands to change the values of environment variables, which we'll cover in a moment.

The version command is used to check the version number of uboot, type "version", and the version number of uboot is shown in the figure (the author here is the test version, the specific version is subject to the actual source code and image, and the version number is only for reference) :

```
STM32MP> version
U-Boot 2023.10-stm32mp-r1 (Jan 07 2025 - 14:39:15 +0800)

aarch64-ostl-linux-gcc (GCC) 13.3.0
GNU ld (GNU Binutils) 2.42.0.20240716
```

Figure 1.2-6 version command results

1.2.2 Environment variable manipulation commands

1. Modify environment variables

There are two commands to manipulate environment variables: setenv and saveenv.The setenv command sets or changes the value of an environment variable. The saveenv command is used to save the modified environment variables. Generally, environment variables are stored in external flash. When uboot starts, environment variables will be read from flash to DRAM. Therefore, the command

setenv is used to modify the environment variable value in DRAM. After modification, the saveenv command should be used to save the modified environment variable to flash, otherwise uboot will continue to use the previous environment variable value in the next restart.

The saveenv command is simple to use.The format is:

```
Saveenv
```

For example, if we want to change the bootdelay environment variable to 5, we can use the following command:

```
setenv bootdelay 5
saveenv
```

The preceding command executes as shown in the following figure:

```
STM32MP> setenv bootdelay 5
STM32MP> saveenv
Saving Environment to MMC... Writing to MMC(1)... OK
STM32MP>
```

Figure 1.2-7 Environment variable modification

When we save the changed environment variables using the saveenv command, we see that the environment variables are saved to MMC(1), which is also called EMMC. Since I started it with EMMC, it will be saved to MMC(1). After modifying bootdelay, restart the development board, uboot is changed to 5 seconds countdown, as shown in the following figure:

```
U-Boot 2023.10-stm32mp-r1 (Jan 07 2025 - 14:39:15 +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
In:    serial
Out:   serial
Err:   serial
invalid MAC address 0 in OTP 00:00:00:00:00:00
Net:
Error: eth2@482d0000 address not set.

Error: eth1@482c0000 address not set.

Error: eth1@482c0000 address not set.

Error: eth2@482d0000 address not set.
No ethernet found.

Hit any key to stop autoboot:  5
```

Figure 1.2-8 Five seconds countdown

As can be seen from the figure, at this point the countdown to uboot becomes 5 seconds.

2. Create a new environment variable

The setenv command can also be used to create a new command, in the same way as the environment variable.For example, if we create an environment variable author with the value 'console=ttySTM0,115200 root=/dev/mmcblk1p1 rootwait rw', You can use the following command:

```
setenv author 'console=ttySTM0,115200 root=/dev/mmcblk1p1 rootwait rw '
saveenv
```

The preceding command sets author to "console=ttySTM0,115200 root=/dev/mmcblk2p1 rootwait rw", Where "console=ttySTM0,115200", "root=/dev/mmcblk1p1", "rootwait", and "rw" are equivalent to four sets of "values", which are separated by Spaces, so they need to be enclosed in single quotes. Indicates that all four "values" belong to the environment variable author.

After the author command has been created, restart uboot, and then use the command printenv to see the current environment variables, as shown in the figure:

```
STM32MP> setenv author 'console=ttySTM0,115200 root=/dev/mmcblk1p1 rootwait rw '
STM32MP> saveenv
Saving Environment to MMC... Writing to MMC(1)... OK
STM32MP>
```

Figure 1.2-9 The value of the new author environment variable

3.Remove environment variables

Now that we can create new environment variables, we can remove them, again using the setenv command,

To delete an environment variable, simply give it an empty value.For example, to delete the author environment variable we created above, use the following command:

```
setenv author
saveenv
```

In the preceding command, the author environment variable is deleted by assigning an empty value to author via setenv, which means that nothing is written. Restarting uboot will show that the author environment variable is gone.

```
STM32MP> setenv author
STM32MP> saveenv
Saving Environment to MMC... Writing to redundant MMC(1)... OK
STM32MP> printenv author
## Error: "author" not defined
```

Figure 1.2-10 Delete the author environment variable

1.2.3 Network operation commands

uboot supports network, we generally need to tune the network function when using uboot, because when using the linux kernel, we need to use the network function of uboot for debugging. uboot supports a number of networking commands, such as dhcp, ping, nfs, and tftpboot, which we'll look at in turn.

It is recommended that the development board network cable and the host PC are connected to the same router! Here the author uses the ENET2 development board. Finally, set the environment variables shown in the following table:

environment variable	Description
ethact	The network card address should be configured according to the specific address in the chip manual.
ipaddr	Develop the ip address of the board, can not set, use the dhcp command to get the IP address from the router.
ethaddr	Development board ENET MAC address, if you want to use ENET must be set.
gatewayip	The gateway address.

netmask	Subnet mask.
serverip	The server IP address, which is the Ubuntu host IP address, is used to debug the code.

Table 1.2 1 Network-related environment variables

The command to set the environment variables in the preceding table is as follows:

```
setenv ethact eth2@482d0000
setenv ipaddr 192.168.6.55
setenv ethaddr 36:26:70:88:e4:7a
setenv gatewayip 192.168.6.1
setenv netmask 255.255.255.0
setenv serverip 192.168.6.170
saveenv
```

Note that the setting of the network address environment variable should be based on your actual situation to ensure that the IP address of the Ubuntu host and the development board are in the same network segment. For example, my current development board and computer are in the 192.168.6.0 network segment, so the IP address of the development board can be set to 192.168.6.55. My Ubuntu host address is 192.168.6.170, so the serverip is 192.168.6.170. ethaddr is a network MAC address, which is a 48bit address. If there are multiple development boards in the same network segment, make sure that the ethaddr of each development board is different, otherwise there will be problems in communication! Now that you have your network-related environment variables set, you can start using network-related commands.

The ethact environment variable holds the register address of the specific network card-in the case of MP257, ENET2 is 482d0000.

```
STM32MP> setenv ethact eth2@482d0000
STM32MP> setenv ipaddr 192.168.6.55
STM32MP> setenv ethaddr 36:26:70:88:e4:7a
STM32MP> setenv gatewayip 192.168.6.1
STM32MP> setenv netmask 255.255.255.0
STM32MP> setenv serverip 192.168.6.170
STM32MP> saveenv
Saving Environment to MMC... Writing to MMC(1)... OK
```

Figure 1.2 11 Setting uboot's ENET2 information

1.The ping command

Whether the network of the development board can be used, whether it can communicate with the server (Ubuntu host), you can verify through the ping command, directly ping the IP address of the server, for example, the IP address of the author's server is 192.168.6.170, the command is as follows: ping 192.168.6.170

The result is shown in the following figure:

```
STM32MP> ping 192.168.6.170

Error: eth1@482c0000 address not set.
Using eth2@482d0000 device
host 192.168.6.170 is alive
```

Figure 1.2-12 The ping command

Attention! You can only ping other machines in uboot, other machines cannot ping uboot, because uboot does not handle the ping command, and if you ping uboot from another machine, it will fail!

2. dhcp command

dhcp is used to get the IP address from the router, the premise is that the development board has to be connected to the router, if the development board is directly connected to the computer, then the dhcp command will fail. DHCP doesn't just get an IP address, it also starts the linux kernel via TFTP by typing "? dhcp "to view the details of the dhcp command, as shown in the figure:

```
STM32MP> ? dhcp
dhcp - boot image via network using DHCP/TFTP protocol

Usage:
dhcp [loadAddress] [[hostIPaddr:]bootfilename]
STM32MP>
```

Figure 1.2-13 dhcp commands use queries

Here, we only need to use DHCP to get network parameters, and we don't need TFTP yet. You can tell uboot not to start TFTP after DHCP by setting the environment variable autoload, and then you can get the IP address through the router by typing the dhcp command as follows:

```
setenv autoload no
saveenv
dhcp
```

```
STM32MP> setenv autoload no
STM32MP> saveenv
Saving Environment to MMC... Writing to MMC(1)... OK
STM32MP> dhcp
BOOTP broadcast 1
BOOTP broadcast 2
DHCP client bound to address 192.168.6.231 (281 ms)
STM32MP>
```

Figure 1.2-14 Use the dhcp command

By default, the dhcp command of uboot will use the corresponding network port according to ethact. Here, the author's ethact environment variable is eth2@482d0000, and the ENET2 interface of the development board is used.

3. The tftpboot command

The tftpboot command uses the TFTP protocol, and the Ubuntu host is used as the TFTP server. This command is usually used for network boot or system upgrade of embedded systems. Therefore, we need to set up a tftp server on Ubuntu, and we need to install tftp-hpa and tftpd-hpa. In Ubuntu, tftpboot cannot run on its own and requires network management server support, which requires xinetd. The command is as follows:

```
sudo apt-get install tftp-hpa tftpd-hpa
sudo apt-get install xinetd
```

TFTP also needs a folder to hold your files.Create a new directory in your user directory with the following command:

```
mkdir /home/alientek/linux/tftpboot
chmod 777 /home/alientek/linux/tftpboot
```

So I'll create it on my computer called a tftpboot directory (folder), path is/home/alientek/Linux/tftpboot. Attention! We need to give tftpboot folder permissions, otherwise uboot can't download files from tftpboot folder.

Finally, configure tftp and create a new /etc/xinetd.d/tftp file, or create a new /etc/xinetd.d directory if you don't already have one (use sudo permission). Then enter the following inside:

Example code 1.2.3.1 /etc/xinetd.d/tftp file contents

```
server tftpboot
{
    socket_type = dgram
    wait = yes
    disable = no
    user = root
    protocol = udp
    server = /usr/sbin/in.tftpd
    server_args = -s /home/alientek/linux/tftpboot -c
    #log_on_success += PID HOST DURATION
    #log_on_failure += HOST
    per_source = 11
    cps =100 2
    flags =IPv4
}
```

After that, start the tftp server with the following command:

```
sudo service tftpd-hpa start
```

Open the /etc/default/tftpd-hpa file and change it to the following:

Example code 1.2.3.2 /etc/default/tftpd-hpa file contents

```
# /etc/default/tftpd-hpa

TFTP_USERNAME="tftp"
TFTP_DIRECTORY="/home/alientek/linux/tftpboot"
TFTP_ADDRESS=":69"
TFTP_OPTIONS="-l -c -s"
```

TFTP_DIRECTORY is the tftp folder directory that we created above. From now on, we will put all the files that need to be transferred via TFTP into this folder, and give these files the appropriate permissions.

Finally, restart the tftp server with the following command:

```
sudo service tftpd-hpa restart
```

Now that the tftp server is set up, it's time to use it. Copy the image.gz Image file into the tftpboot folder and give image.gz the appropriate permissions with the following command:

```
cp Image.gz /home/alientek/linux/tftpboot/
cd /home/alientek/linux/tftpboot/
chmod 777 Image.gz
```

We have everything we need to validate, and the tftp command in uboot has the following format:

```
tftp [loadAddress] [[hostIPaddr:]bootfilename]
```

This looks like the same format as the nfs command, with loadAddress being the DDR address of the file and [[hostIPaddr:]bootfilename] being the file to download from Ubuntu. However, unlike the nfs command, the tftp command does not require the full path to the file in Ubuntu, only the filename. For example, we now download the Image.gz file in the tftpboot folder to the 0x90000000 address of the development board DDR, and the device tree downloads to the 0x90000000 command of the DDR as follows:

tftpboot 90000000 Image.gz

The download process looks like this:

```
STM32MP> tftpboot 90000000 Image.gz

Error: eth1@482c0000 address not set.
Using eth2@482d0000 device
TFTP from server 192.168.6.170; our IP address is 192.168.6.55
Filename 'Image.gz'.
Load address: 0x90000000
Loading: #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #######################
         3.2 MiB/s
done
Bytes transferred = 12736099 (c25663 hex)
STM32MP>
```

Figure 1.2-15 tftpboot test

1.2.4 EMMC and SD card operation commands

In this section, the micro SD card should be connected to the TF card slot of the development board in advance and the development board should be started in EMMC mode.

uboot supports EMMC and SD cards, so it is also necessary to provide EMMC and SD card operation commands. EMMC and SD card are generally considered to be the same thing, so in this tutorial we will use MMC to refer to EMMC and SD card. The command commonly used to operate MMC devices in uboot is "mmc".

mmc is a series of commands that can be followed with different parameters by typing "? mmc "to see the commands related to mmc, as shown in the figure:

```
STM32MP> ? mmc
mmc - MMC sub system

Usage:
mmc info - display info of the current MMC device
mmc read addr blk# cnt
mmc write addr blk# cnt
mmc erase blk# cnt
mmc rescan [mode]
mmc part - lists available partition on current mmc device
mmc dev [dev] [part] [mode] - show or set current mmc device [partition] and set mode
  - the required speed mode is passed as the index from the following list
    [MMC_LEGACY, MMC_HS, SD_HS, MMC_HS_52, MMC_DDR_52, UHS_SDR12, UHS_SDR25,
    UHS_SDR50, UHS_DDR50, UHS_SDR104, MMC_HS_200, MMC_HS_400, MMC_HS_400_ES]
mmc list - lists available devices
mmc wp [PART] - power on write protect boot partitions
  arguments:
   PART - [0|1]
        : 0 - first boot partition, 1 - second boot partition
          if not assigned, write protect all boot partitions
mmc hwpartition <USER> <GP> <MODE> - does hardware partitioning
  arguments (sizes in 512-byte blocks):
   USER - <user> <enh> <start> <cnt> <wrrel> <{on|off}>
        : sets user data area attributes
   GP - <{gp1|gp2|gp3|gp4}> <cnt> <enh> <wrrel> <{on|off}>
        : general purpose partition
   MODE - <{check|set|complete}>
        : mode, complete set partitioning completed
  WARNING: Partitioning is a write-once setting once it is set to complete.
  Power cycling is required to initialize partitions after set to complete.
mmc bootbus <dev> <boot_bus_width> <reset_boot_bus_width> <boot_mode>
 - Set the BOOT_BUS_WIDTH field of the specified device
mmc bootpart-resize <dev> <boot part size MB> <RPMB part size MB>
 - Change sizes of boot and RPMB partitions of specified device
mmc partconf <dev> [[varname] | [<boot_ack> <boot_partition> <partition_access>]]
 - Show or change the bits of the PARTITION_CONFIG field of the specified device
   If showing the bits, optionally store the boot_partition field into varname
mmc rst-function <dev> <value>
 - Change the RST_n_FUNCTION field of the specified device
   WARNING: This is a write-once field and 0 / 1 / 2 are the only valid values.
mmc setdsr <value> - set DSR register value
```

Figure 1.2-16 mmc command

As can be seen from the figure, mmc followed by different parameters can achieve different functions, as shown in the table below:

command	Description
mmc info	Output MMC device information
mmc read	Read the data in the MMC.
mmc wirte	Writes data to the MMC device.
mmc rescan	Scan the MMC device.
mmc part	Lists the partitions of the MMC device.
mmc dev	Switch MMC devices.
mmc list	List all MMC devices currently in force.
mmc hwpartition	Sets the partition of the MMC device.
mmc bootbus……	Sets the value of the BOOT_BUS_WIDTH field for the specified MMC device.
mmc bootpart……	Sets the size of the boot and RPMB partitions for the specified MMC device.
mmc partconf……	Sets the value of the PARTITION_CONFG field of the specified MMC device.

mmc rst	Reset MMC device
mmc setdsr	Set the value of the DSR register.

Table 1.2 2 mmc commands

1, mmc list command

The mmc list command is used to see how many MMC devices are on the current development board by typing "mmc list". The result is shown in the figure:

```
STM32MP> mmc list
STM32 SD/MMC: 0
STM32 SD/MMC: 1 (eMMC)
```

Figure 1.2-17 Scanning MMC devices

It can be seen that the current development board has two MMC devices: SD/MMC: 0 and SD/MMC: 1 (eMMC), this is because the author is now using the EMMC version of the core board, plus SD card a total of two MMC devices, SD/MMC: 0 is SD card, SD/MMC: 1 is EMMC. To see the EMMC device information, use the command "mmc dev" to set EMMC as the current MMC device. Similarly, to see the SD card device information, use the command "mmc dev" to set the SD card as the current MMC device.

2. mmc dev command

The mmc dev command is used to switch the current MMC device, and the format is as follows:

```
mmc dev [dev] [part]
```

[dev] is used to set the MMC device number to be switched, and [part] is the partition number. Switch to EMMC with the following command:

```
mmc dev 1 // Switch to EMMC, 1 for EMMC, 0 for SD card
```

The result is shown in the figure:

```
STM32MP> mmc dev 1
switch to partitions #0, OK
mmc1(part 0) is current device
STM32MP>
```

Figure 1.2-18 Switch to EMMC

3, mmc info command

The mmc info command outputs information about the currently selected mmc info device by entering the command "mmc info", as shown in the figure:

```
STM32MP> mmc info
Device: STM32 SD/MMC
Manufacturer ID: d6
OEM: 3
Name: 88A43A
Bus Speed: 52000000
Mode: MMC DDR52 (52MHz)
Rd Block Len: 512
MMC version 5.1
High Capacity: Yes
Capacity: 14.6 GiB
Bus Width: 8-bit DDR
Erase Group Size: 512 KiB
HC WP Group Size: 4 MiB
User Capacity: 14.6 GiB WRREL
Boot Capacity: 4 MiB ENH
RPMB Capacity: 4 MiB ENH
Boot area 0 is not write protected
Boot area 1 is not write protected
```

Figure 1.2-19 mmc info command

As can be seen from the figure, the currently selected MMC device is EMMC, EMMC chip version 5.1, capacity 14.6GiB(16GB for EMMC), speed 52000000Hz=52MHz, 8-bit wide bus. There is also a command with the same functionality as the mmcinfo command: mmcinfo, with no space between "mmc" and "info".

4, mmc part command

Sometimes the SD card or EMMC has more than one partition, you can use the command "mmc part" to see the partition, for example, to see the partition of EMMC, enter the following command:

```
mmc dev 1 // Switch to EMMC
mmc part // View the EMMC partition
```

The result is shown in the figure:

```
STM32MP> mmc dev 1
switch to partitions #0, OK
mmc1(part 0) is current device
STM32MP> mmc part

Partition Map for MMC device 1  --   Partition Type: EFI

Part    Start LBA       End LBA         Name
        Attributes
        Type GUID
        Partition GUID
  1     0x00000400      0x000007ff      "metadata1"
        attrs:  0x0000000000000000
        type:   8a7a84a0-8387-40f6-ab41-a8b9a5a60d23
                (8a7a84a0-8387-40f6-ab41-a8b9a5a60d23)
        guid:   460dd9dd-6999-4b0d-b8b4-926dfef05a7e
  2     0x00000800      0x00000bff      "metadata2"
        attrs:  0x0000000000000000
        type:   8a7a84a0-8387-40f6-ab41-a8b9a5a60d23
                (8a7a84a0-8387-40f6-ab41-a8b9a5a60d23)
        guid:   7515a3c9-33c5-4045-b7f9-4e0d7251a01a
  3     0x00000c00      0x00002bff      "fip-a"
        attrs:  0x0000000000000000
        type:   19d5df83-11b0-457b-be2c-7559c13142a5
                (19d5df83-11b0-457b-be2c-7559c13142a5)
        guid:   4fd84c93-54ef-463f-a7ef-ae25ff887087
  4     0x00002c00      0x00004fff      "fip-b"
        attrs:  0x0000000000000000
        type:   19d5df83-11b0-457b-be2c-7559c13142a5
                (19d5df83-11b0-457b-be2c-7559c13142a5)
        guid:   09c54952-d5bf-45af-acee-335303766fb3
  5     0x00005000      0x00044fff      "bootfs"
        attrs:  0x0000000000000004
        type:   0fc63daf-8483-4772-8e79-3d69d8477de4
                (linux)
        guid:   ac6ce43f-c07e-4ad7-ab4a-10f6a3ac5c57
  6     0x00045000      0x01d28fde      "rootfs"
        attrs:  0x0000000000000000
        type:   0fc63daf-8483-4772-8e79-3d69d8477de4
                (linux)
        guid:   491f6117-415d-4f53-88c9-6e0de54deac6
```

Figure 1.2-20 Look at the EMMC partition

If you want to set partition 2 of EMMC to the current MMC setting partition, you can use the following command:

```
mmc dev 1 2
```

The result is shown in the figure:

```
STM32MP> mmc dev 1 2
switch to partitions #2, OK
mmc1(part 2) is current device
STM32MP>
```

Figure 1.2-21 Set EMMC partition 2 as the current device

5, mmc read command

The mmc read command is used to read data from mmc devices in the following format:

```
mmc read addr blk# cnt
```

addr is the address where the data will be read into DRAM, blk is the start address of the block to be read (hexadecimal), and a block is 512 bytes, where block and sector are the same meaning; in MMC devices, we usually say sector, and cnt is the number of blocks to be read (hexadecimal). For example, starting from the 1024th (0x400) block of EMMC partition 1 and reading 16(0x10) blocks of data to the DRAM address 0X90000000, the command is as follows:

```
mmc dev 1 0 // Switch to EMMC partition 1
mmc read 90000000 400 10 // Read data
```

The result is shown in the figure:

```
STM32MP> mmc dev 1 0
switch to partitions #0, OK
mmc1(part 0) is current device
STM32MP> mmc read 90000000 400 10

MMC read: dev # 1, block # 1024, count 16 ... 16 blocks read: OK
```

Figure 1.2-22 mmc read command

We can't tell if we read it correctly, so simply run md.b to look at 0x90000000, which is 16*512=8192(0x2000) bytes:

```
md.b 90000000 2000
```

The result is shown in the figure:

```
STM32MP> md.b 90000000 2000
90000000: b4 08 4b d1 02 00 00 00 00 00 00 01 00 00 00  ..K.............
90000010: 78 00 00 00 20 00 00 00 fc fc ff ff 00 00 00 00  x... ............
90000020: 02 00 01 00 50 00 18 00 83 df d5 19 b0 11 7b 45  ....P.........{E
90000030: be 2c 75 59 c1 31 42 a5 a0 84 7a 8a 87 83 f6 40  .,uY.1B...z....@
90000040: ab 41 a8 b9 a5 a6 0d 23 93 4c d8 4f ef 54 3f 46  .A.....#.L.O.T?F
90000050: a7 ef ae 25 ff 88 70 87 01 00 00 00 00 00 00 00  ...%..p.........
90000060: 52 49 c5 09 bf d5 af 45 ac ee 33 53 03 76 6f b3  RI.....E..3S.vo.
90000070: 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

Figure 1.2-23 Read data (partial screenshot)

1.2.5 FAT filesystem Manipulation commands

Sometimes you need to operate files stored in the SD card or EMMC in uboot. The file manipulation commands are: fatinfo, fatls, fstype, fatload, and fatwrite, but these file manipulation commands only support FAT file systems!! Since the MP257 system does not use partitions in FAT format, no examples are given in this subsection.

1. fatinfo command

The fatinfo command is used to query the filesystem information for a partition of a given MMC device in the following format:

```
fatinfo <interface> [<dev[:part]>]
```

interface denotes the interface, for example mmc, dev is the device number to query, and part is the partition to query.

2. fatls command

The fatls command is used to query the directory and file information of the FAT format device, and the command format is as follows:

```
fatls <interface> [<dev[:part]>] [directory]
```

interface is the interface to query, such as mmc, dev is the device number to query, part is the partition to query, and directory is the directory to query.

3, fstype command

fstype is used to view the file system format of a partition of an MMC device, and the command format is as follows:

```
fstype <interface> <dev>:<part>
```

4,fatload command

The fatload command is used to read a specified file into DRAM in the following format:

```
fatload <interface> [<dev[:part]> [<addr> [<filename> [bytes [pos]]]]]
```

interface is the interface, such as mmc, dev is the device number, part is the partition, addr is the start address to be stored in DRAM, and filename is the name of the file to be read. bytes indicates how many bytes of data to read, and if bytes is 0 or omitted, the entire file is read. pos is the offset of the file to be read from the beginning of the file.A pos value of 0 or an omitted value indicates that the file should be read from the beginning of the file.

5,fatwrite command

The fatwirte command is used to write data from DRAM to the MMC device. The format of the command is as follows:

```
fatwrite <interface> <dev[:part]> <addr> <filename> <bytes>
```

interface is the interface, such as mmc, dev is the device number, part is the partition, addr is the starting address of the data to be written in DRAM, filename is the name of the data file to be written, and bytes indicates how many bytes of data to be written.

1.2.6 EXT Format Filesystem Manipulation commands

uboot has two file system operation commands in ext2 and ext4 format, and the commonly used four commands are:

ext2load, ext2ls, ext4load, ext4ls, and ext4write. These commands have the same meaning and use as fatload, fatls, and fatwrite, except that both ext2 and ext4 are specific to additional filesystems. For example, the ext4ls command, EMMC partition 5 is ext4 format.Using ext4ls, you can query the files and directories in EMMC partition 5 by typing:

```
ext4ls mmc 1:5
```

The result is shown in the figure:

```
STM32MP> ext4ls mmc 1:5
<DIR>       4096 .
<DIR>       4096 ..
<DIR>      16384 lost+found
        11774753 Image.gz
          107657 stm32mp257d-atk-ddr-2GB.dtb
          110574 stm32mp257d-atk-ddr-2GB-lvds-1xSingleLink.dtb
          110956 stm32mp257d-atk-ddr-2GB-lvds-2xSingleLink.dtb
          111139 stm32mp257d-atk-ddr-2GB-lvds-dualLink.dtb
          110430 stm32mp257d-atk-ddr-2GB-mipi.dtb
          109913 stm32mp257d-atk-ddr-2GB-rgb.dtb
<DIR>       4096 6.6.48
            4187 boot.scr.uimg
         8376477 st-image-resize-initrd
<DIR>       4096 mmc0_extlinux
<DIR>       4096 mmc1_extlinux
STM32MP>
```

Figure 1.2-24 EMMC partition 5 file query

1.2.7 BOOT action commands

uboot's job is essentially to boot Linux, so uboot must have an associated boot command to boot Linux. The common boot-related commands are bootm, bootz, and boot.

1.The bootm command

To boot Linux, you need to copy the Linux image file into DRAM, and if you are using the device tree, you also need to copy the device tree into DRAM. Linux images and device tree files can be copied from storage devices such as EMMC or NAND to DRAM, and Linux images and device tree files can be downloaded to DRAM via nfs or tftp. Either way, as long as you can save the Linux Image and the device tree file to DRAM, you can use the booti command to boot the image image.The format of the booti command is as follows:

```
bootm [addr [arg ...]]
```

The bootm command has three main parameters, addr is the location of the Linux image file in DRAM, followed by "arg...". Indicates additional optional parameters, such as the address of the initrd in DRAM. The Linux kernel requires a third parameter to specify the address of the device tree in DRAM if it uses a device tree, and a second parameter to replace it with '-' if initrd is not needed.

Now we use both network and EMMC methods to boot the Linux system, first send the Linux image and device tree of the ATK-DLMP257B development board to the tftpboot folder in the Ubuntu host. The Linux image file has been put into the tftpboot folder, now put the compiled device tree file into the tftpboot folder and give the permissions, and the finished tftpboot folder will look like this:

1.2.8 The ENV environment variable command

If we want to restore the default factory uboot environment variable, we need to clear out the environment variables we configured earlier, which is where we need to import the env command. Run the command to see the env description:

```
? env
```

```
STM32MP> ? env
env - environment handling commands

Usage:
env default [-f] -a - [forcibly] reset default environment
env default [-f] var [...] - [forcibly] reset variable(s) to their default values
env delete [-f] var [...] - [forcibly] delete variable(s)
env edit name - edit environment variable
env exists name - tests for existence of variable
env export [-t | -b | -c] [-s size] addr [var ...] - export environment
env import [-d] [-t [-r] | -b | -c] addr [size] [var ...] - import environment
env info - display environment information
env info [-d] [-p] [-q] - evaluate environment information
      "-d": default environment is used
      "-p": environment can be persisted
      "-q": quiet output
env print [-a | name ...] - print environment
env run var [...] - run commands in an environment variable
env save - save environment
env erase - erase environment
env set [-f] name [arg ...]
```

Figure 1.2-25 See the env command instructions

You can see that the env directive is mainly around the uboot environment variable to operate, such as clear, print, write, and so on. Here, we'll focus on the most common directives.

1. env print

Print ENVIRONMENT VARIABLES. This is similar to the print directive we used earlier.

Usage:

env print -a

env print name

env print -a prints all the changed variables. env print name prints the specified environment variable.name is the specific environment variable name. For example, if I print the ver environment variable, the output is as follows:

```
STM32MP> env print ver
ver=U-Boot 2023.10-stm32mp-r1 (Jan 07 2025 - 14:39:15 +0800)
```

Figure 1.2-26 Print the ver environment variable

2, env default

This directive resets the current environment variable contents to the default environment variable contents

Usage:

env default -a // Reset all current environment variables to default ones

env default -f -a // Force a reset of all environment variables

env default name // Resets the concrete environment variable where name is the concrete environment variable name

saveenv // Save environment variables

```
STM32MP> env default -a
## Resetting to default environment
STM32MP> env default -f -a
## Resetting to default environment
STM32MP> env default name
STM32MP> saveenv
Saving Environment to MMC... Writing to redundant MMC(1)... OK
STM32MP>
```

Figure 1.2-27 env default example

1.2.9 UMS command

Under uboot, we can virtual the development board into a U disk, and we can choose which Flash to use as the memory of the U disk. For example, the EMMC or SD card on the ALIENTEK ATK-DLMP257B development board can be virtual into a U disk. When we virtual EMMC into a U disk, we can directly copy files to the development board on the computer. For example, we can directly copy a file under uboot to the root file system of the development board in the product development stage, so that there is no need to enter the system or replace the file through the network.

The ums command provided by uboot does this:

```
ums <USB_controller> [<devtype>] <dev[:part]>
```

Where USB_controller is the usb interface index, some development boards have more than one USB SLAVE interface, the specific which to use can be specified through the USB_controller parameter. Only one USB_OTG port of the ATK-DLMP257B development board can be used as a USB SLAVE, and the corresponding index is 0. Devtype is the device to mount, which defaults to mmc, dev[:part] is the Flash device to mount, and part is the partition to mount.

Here we mount the EMMC of the development board to the computer, first use the USB Type-C cable to connect the USB_OTG port of the development board with the computer, and then use the following command to start.

```
ums 0 mmc 1
```

The result is shown in the figure:

```
STM32MP> ums 0 mmc 1
UMS: LUN 0, dev mmc 1, hwpart 0, sector 0x0, count 0x1d29000
dwc3-generic-wrapper usb@48300000: configured in usb2 mode
/
```

Figure 1.2-28 Result of ums command

After successful mounting, there will be multiple U disks on the computer. The number of U disks depends on the number of EMMC partitions on your current development board. If you have burned the factory system, the default is 4 U disks, as shown in the figure:

Figure 1.2-29 U disk

Note that under Windows, the second USB key is not operable because it is in ext4 format, which is not supported by Windows! So we found that Windows reported a problem with U disk recognition when operating, so that the format, do not format! Otherwise, the whole Linux system will be formatted!

Since Windows doesn't recognize ext4, we can mount it to Ubuntu, and we'll be able to work with both USB sticks.

To end the mount, run it in the uboot terminal and press CTRL-C.

1.2.10 Other common commands

There are other common commands in uboot, such as reset, go, run, and mtest.

1, reset command

Reset command as the name suggests is reset, type "reset" to reset restart, as shown in the figure:

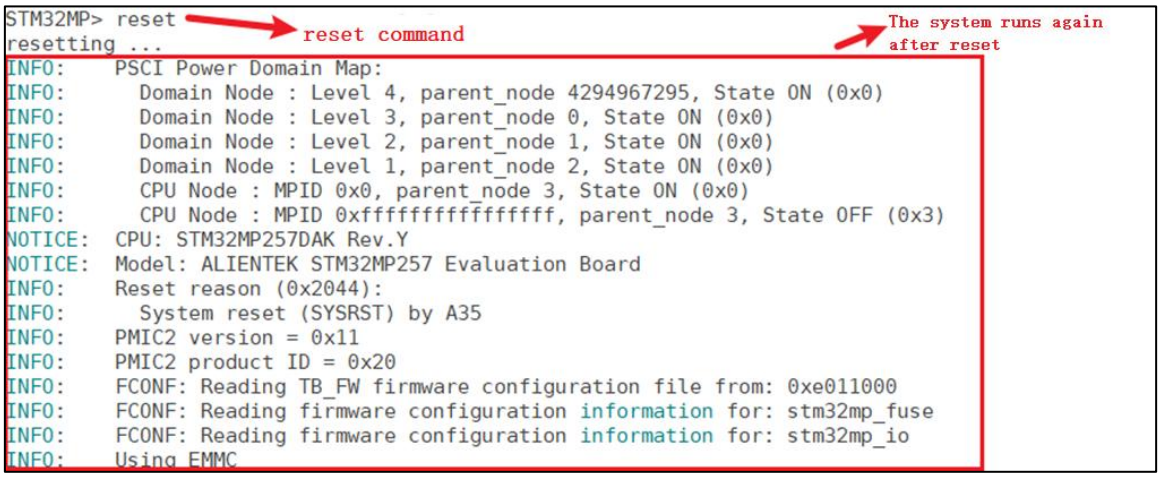


Figure 1.2-30 reset command results

2. go command

The go command jumps the application to a specific location and executes it.This command has the following format:

```
go addr [arg ...]
```

addr is the first address used in DRAM.

3.The run command

The run command is used to run the command defined in the environment variable. For example, we can run the boot command from bootcmd by "run bootcmd", but the most useful of the run command is to run our custom environment variable. The environment variable mybootemmc is used to start from emmc, and mybootnet is used to start from the network. If you want to switch the boot mode, just run "run mybootxxx(xxx is emmc or net)".

4.The mtest command

The mtest command is a simple memory read and write test command, which can be used to test the DDR on the development board. The command format is as follows:

```
mtest [start [end [pattern [iterations]]]]
```

start is the start address of the DRAM to be tested, and end is the end address. For example, if we test the memory from 0X90000000 to 0X90001000, enter "mtest 90000000 90001000" to start the test, and press CTRL+C to end the test, the results are as shown in the figure:

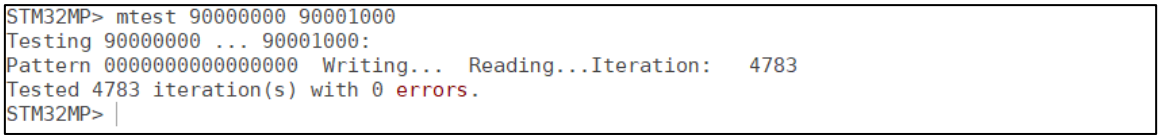


Figure 1.2-31 The results of the mtest command are for reference only

At this point, the commonly used commands of uboot are explained. If you want to use uboot's other commands, you can check the help information in uboot, or check the corresponding information online.