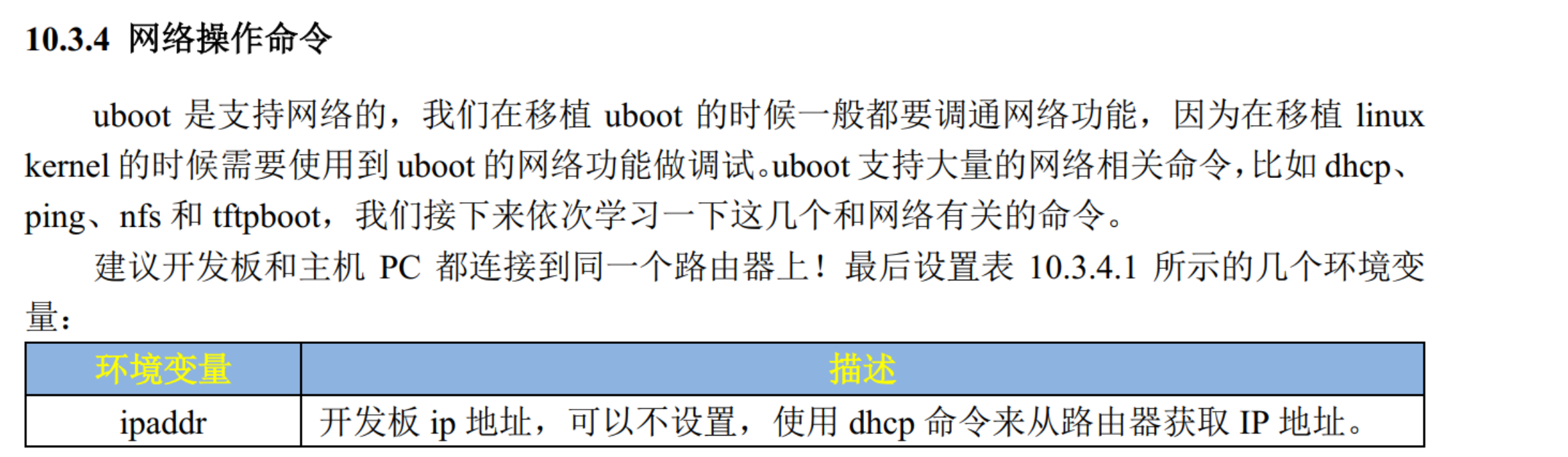
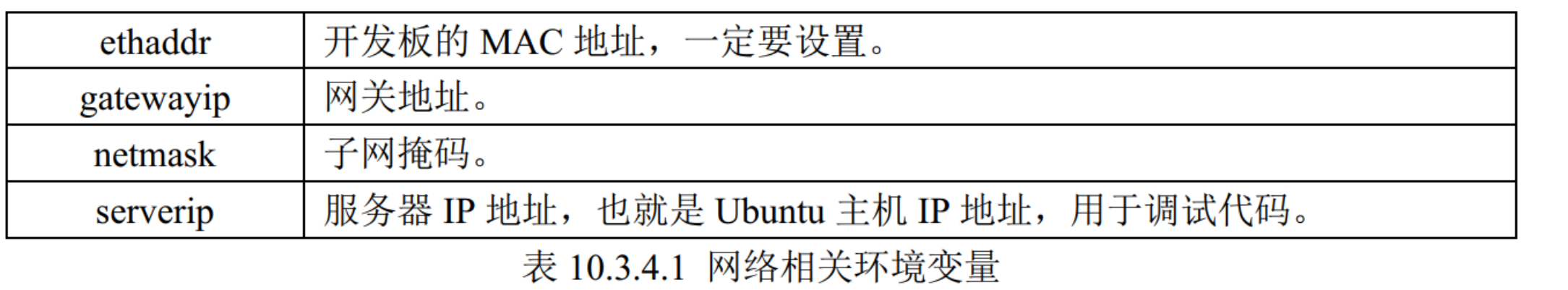
# Linux驱动开发笔记

Ubuntu 虚拟机 IP： 192.168.10.100  
电脑网口的 IP： 192.168.10.200  
开发板 IP 地址： 192.168.10.10

### Uboot ip设置：





setenv ipaddr 192.168.10.10  
setenv gatewayip 192.168.10.1  
setenv netmask 255.255.255.0  
setenv serverip 192.168.10.100

setenv ethaddr b8:ae:1d:01:01:00  
saveenv

### 编译

Makefile 路径 ：

KERNELDIR := /home/book/teach\_linux/

CURRENT\_PATH := $(shell pwd)

make 编译驱动程序

arm-none-linux-gnueabihf-gcc ledApp.c -o ledApp 编译测试程序

### 开发板挂载nfs

ifconfig eth0 192.168.10.10

#### //共享目录

mount -t nfs -o nolock,nfsvers=3 192.168.10.100:/home/book/alientek\_linux/nfs /lib/modules/5.4.31

#### 挂载Linux 内核

##### 1.TFTP挂载

//运行之前确保赋予文件权限 chmod 777 uImage stm32mp157d-atk.dtb

setenv bootcmd 'tftp c2000000 uImage;tftp c4000000 stm32mp157d**-**atk.dtb;bootm c2000000 - c4000000'

saveenv  
boot

##### 2.EMMC 挂载

setenv bootcmd 'ext4load mmc 1:2 c2000000 uImage;ext4load mmc 1:2 c4000000 **stm32mp157d-atk.dtb**;bootm c2000000 - c4000000'  
saveenv  
boot

#### //挂载文件系统

##### 1.NFS挂载

**setenv bootargs 'root=/dev/nfs nfsroot=192.168.10.100:/home/book/alientek\_linux/nfs/rootfs,proto=tcp rw ip=192.168.10.10:192.168.10.100:192.168.10.1:255.255.255.0::eth0:off'**

saveenv  
boot

**setenv bootargs '** console=tty1 console=ttySTM0,115200 **root=/dev/nfs nfsroot=192.168.10.100:/home/book/alientek\_linux/nfs/rootfs,proto=tcp rw ip=192.168.10.10:192.168.10.100:192.168.10.1:255.255.255.0::eth0:off'**

saveenv  
boot

**setenv android\_mmc\_kernel 'if part start mmc ${devnum} boot\_${suffix} boot\_start &&part size mmc ${devnum} boot\_${suffix} boot\_size;then mmc read ${kernel\_addr\_r} ${boot\_start} ${boot\_size};part nb mmc ${devnum} system\_${suffix} rootpart\_nb;env set bootargsroot=/dev/nfs androidboot.serialno=${serial#} androidboot.slot\_suffix=\_${suffix};fi'**

##### 2.EMMC 挂载

setenv bootargs 'console=ttySTM0,115200 root=/dev/mmcblk1p3 rootwait rw'

saveenv  
boot

### uImage .dtb 镜像打包 .ext4

**1**、新建 **ext4** 格式磁盘  
首先新建一个 ext4 格式的磁盘，然后挂载这个 ext4 格式的磁盘，将 stm32mp157d-atk.dtb和 uImage 拷贝到这个 ext4 磁盘即可。

cd bootfs

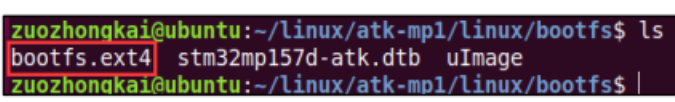
//内核移植（内存大小不一样）  
dd if=/dev/zero of**=**bootfs**.**ext4 bs**=**1M count**=**10

mkfs**.**ext4 **-**L bootfs bootfs**.**ext4

sudo mkdir ./mnt

sudo mount bootfs.ext4 ./mnt

sudo cp uImage stm32mp157d-atk.dtb ./mnt



//文件系统移植

dd if=/dev/zero of**=**rootfs**.**ext4 bs**=**1M count**=**1024

mkfs**.**ext4 **-**L rootfs rootfs**.**ext4  
sudo mkdir ./mnt

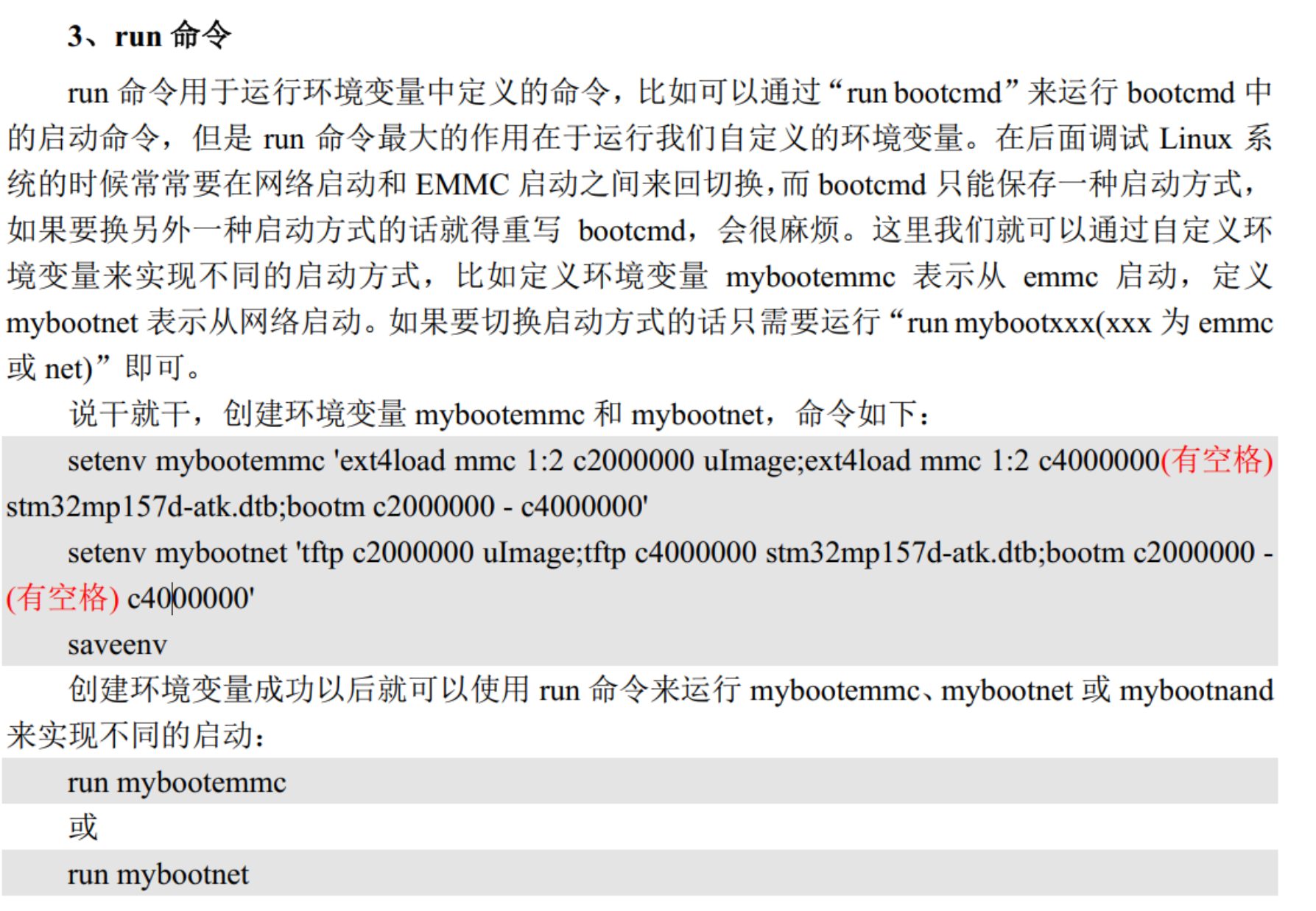
sudo mount rootfs.ext4 ./mnt

sudo cp /home/book/alientek\_linux/nfs/rootfs/\* ./mnt -drf

卸载./mnt

sudo umount ./mnt

### boot模式下从emmc / 网络 加载内核和设备树



setenv mybootemmc 'ext4load mmc 1:2 c2000000 uImage;ext4load mmc 1:2 c4000000 stm32mp157d**-**atk.dtb;bootm c2000000 - c4000000'

setenv mybootnet 'tftp c2000000 uImage;tftp c4000000 stm32mp157d**-**atk.dtb;bootm c2000000 - c4000000'

saveenv

boot

### 驱动模块的安装

modprobe drv.ko

lsmod 查看安装的模块

**dmesg | grep** chrdevbase**检查驱动程序加载是否成功**

mknod /dev/chrdevbase c 200 0 创建设备 200 主设备号 0次设备号

cat /proc/devices 查看设备列表

ls /dev/chrdevbase -l 查看/dev/chrdevbase这个设备信息

./chrdevbaseApp /dev/chrdevbase 1 运行测试程序

rm /dev/ chrdevbase 卸载设备节点

rmmod drv.ko 卸载驱动

设备没反应可能是权限不够： chmod 666 /dev/led

### 查看系统LED驱动

ls /sys/class/leds

cd /sys/class/leds/user-led

cat brightness # 查看当前亮度值

cat trigger # 查看当前触发器

echo none > /sys/class/leds/sys-led/trigger

echo 0 > /sys/class/leds/sys-led/brightness

### 移动文件到另一个文件夹

mv chrdevbaseApp chrdevbase.ko /lib/modules/5.4.31-g886e225be/

输入“lsmod”命令即可查看当前系统中存在的模块

### **查看当前已经被使用掉的设备号**

输入命令“cat /proc/devices”

### 查看模块是否存在

lsmod

### 创建设备节点

驱动加载成功需要在/dev 目录下创建一个与之对应的设备节点文件，应用程序就是通过操作这个设备节点文件来完成对具体设备的操作。输入如下命令创建/dev/chrdevbase 这个设备节点文件：

mknod /dev/chrdevbase c 200 0

使用“ls /dev/chrdevbase -l”命令查看设备节点

### 动态设备号申请：

int alloc\_chrdev\_region(dev\_t \*dev, unsigned baseminor, unsigned count, const char \*name)

**dev**：保存申请到的设备号。  
**baseminor**： 次设备号起始地址， alloc\_chrdev\_region 可以申请一段连续的多个设备号，这些设备号的主设备号一样，但是次设备号不同，次设备号以 baseminor 为起始地址地址开始递增。一般 baseminor 为 0，也就是说次设备号从 0 开始。  
**count**： 要申请的设备号数量。  
**name**：设备名字。

释放掉设备号  
void unregister\_chrdev\_region(dev\_t from, unsigned count)

|  |
| --- |
| **from**：要释放的设备号。 |

**count**： 表示从 from 开始，要释放的设备号数量。

单独编译 uImage  
make uImage LOADADDR=0XC2000040  
单独编译设备树

make dtbs

### 赋予文件权限

chmod 777 stm32mp157d-atk.dtb uImag

### 查看文件权限

ls -l

### 自动创建设备

先创建创建设备号 （有无提前定义主设备号，没有则自动分配 ）获得主和次设备号

初始化cdev 添加一个cdev 创建类 创建设备

删除cdev类 cdev 设备号

### 复制设备树文件

cp stm32mp157d-atk.dtb /home/book/alientek\_linux/nfs

几个常用复制文件

cp stm32mp157d-atk.dtb /home/book/alientek\_linux/nfs

cp stm32mp157d-atk-hdmi.dtb /home/book/alientek\_linux/nfs

cp stm32mp157d-atk-mipi.dtb /home/book/alientek\_linux/nfs

cp -f stm32mp157d-atk.dtb /boot

cp -f stm32mp157d-atk-hdmi.dtb /boot

cp -f stm32mp157d-atk-mipi.dtb /boot

### 查看GPIO使用情况

cat /sys/kernel/debug/gpio

### 查找当前目录相关文件

ls \*stm32mp\*.dts

### 查找某个文件是否有相应关键词

**grep -A 10 -B 10 "led" stm32mp157d-ed1.dts**

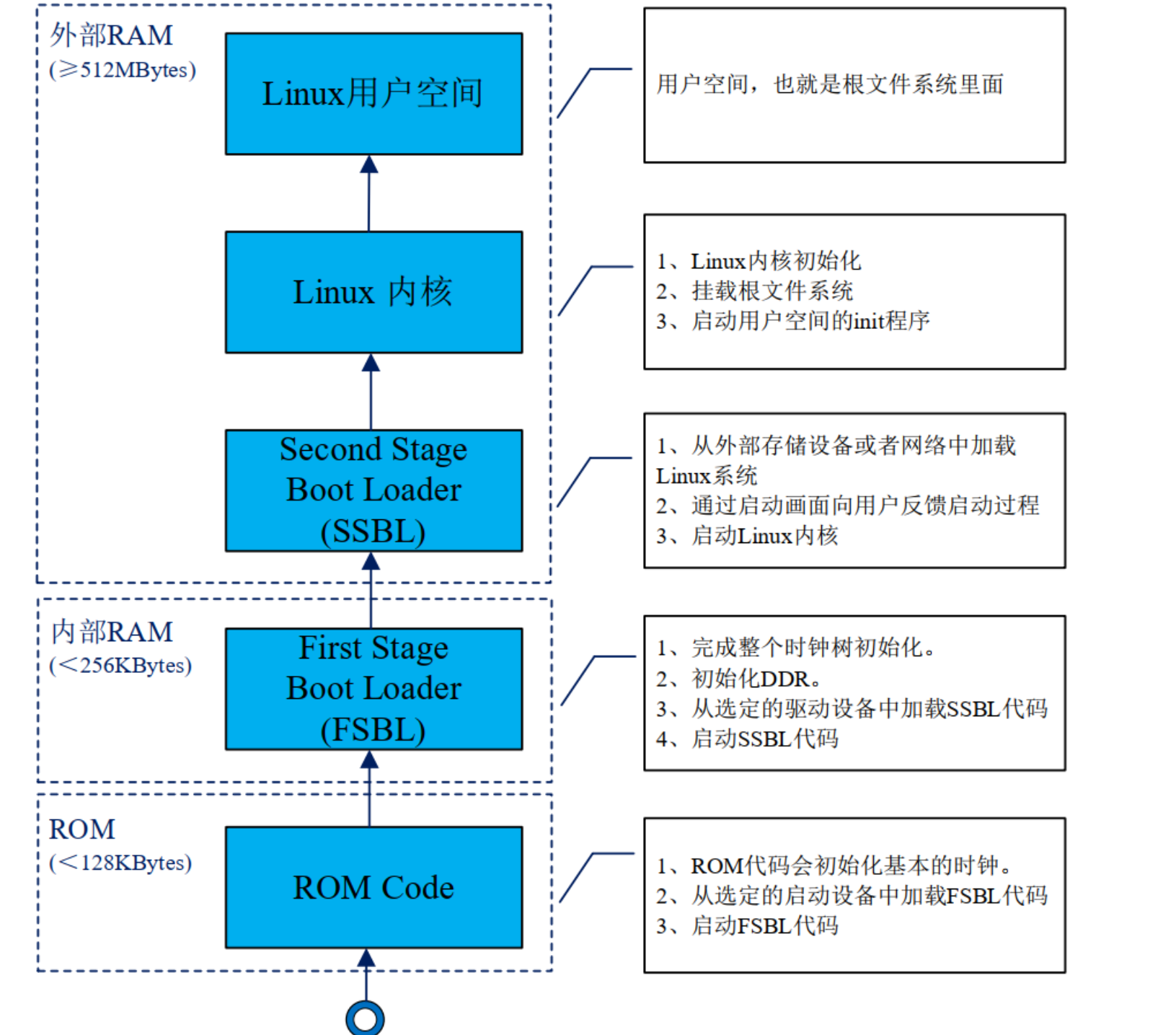
这将在文件中查找 "led" 关键字，并显示每个匹配的前后 10 行，以便更容易找到相关的配置。如果找到了与 **sys-led** 相关的信息，请提供相应的部分内容。

**sys-led在**stm32mp157d-atk.dtsi文件中定义

status = "okay";-> status = "disabled"; 禁用该设备节点

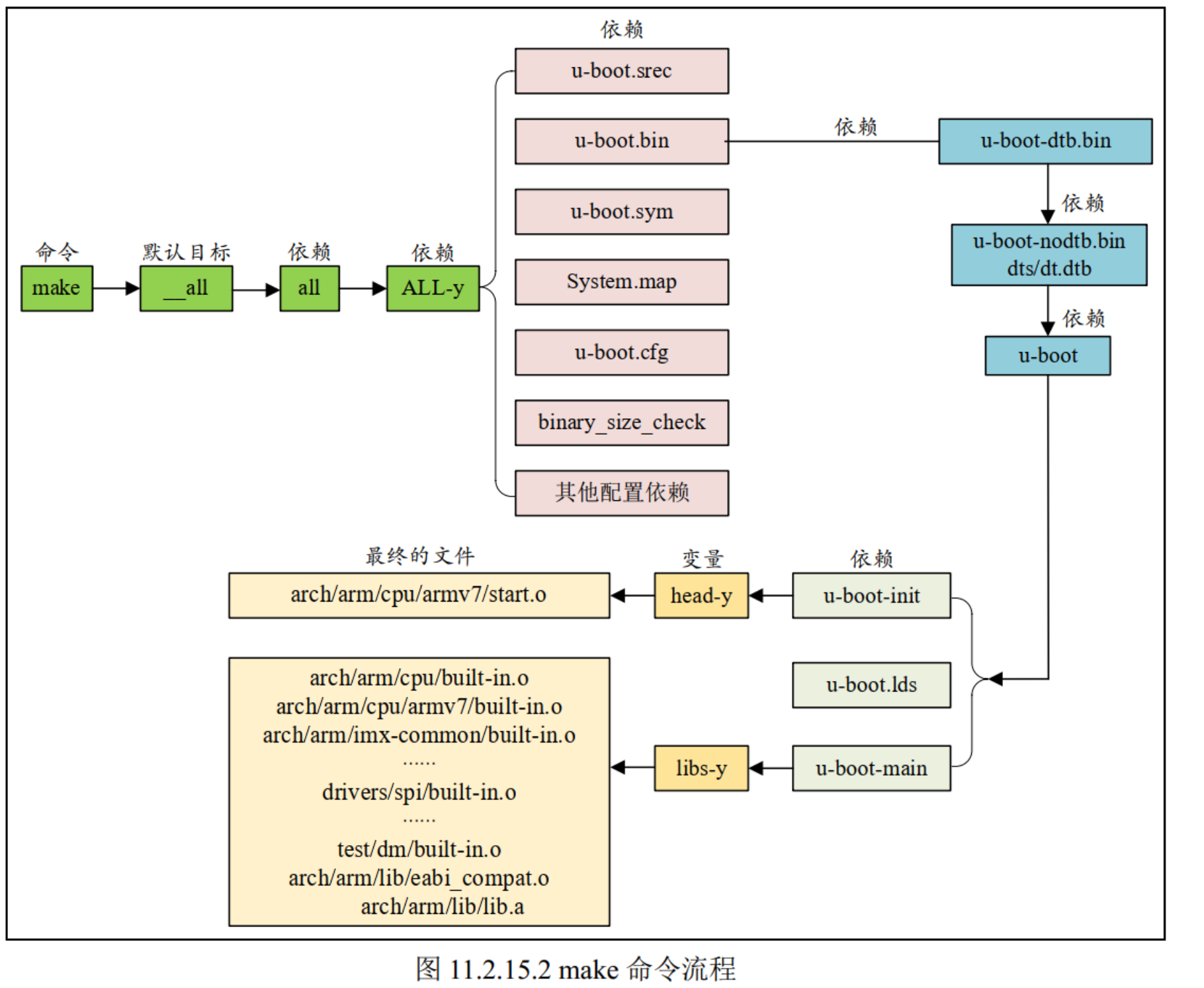
sys\_led uer\_led节点在stm32mp157d-atk.dtsi中

### 启动流程

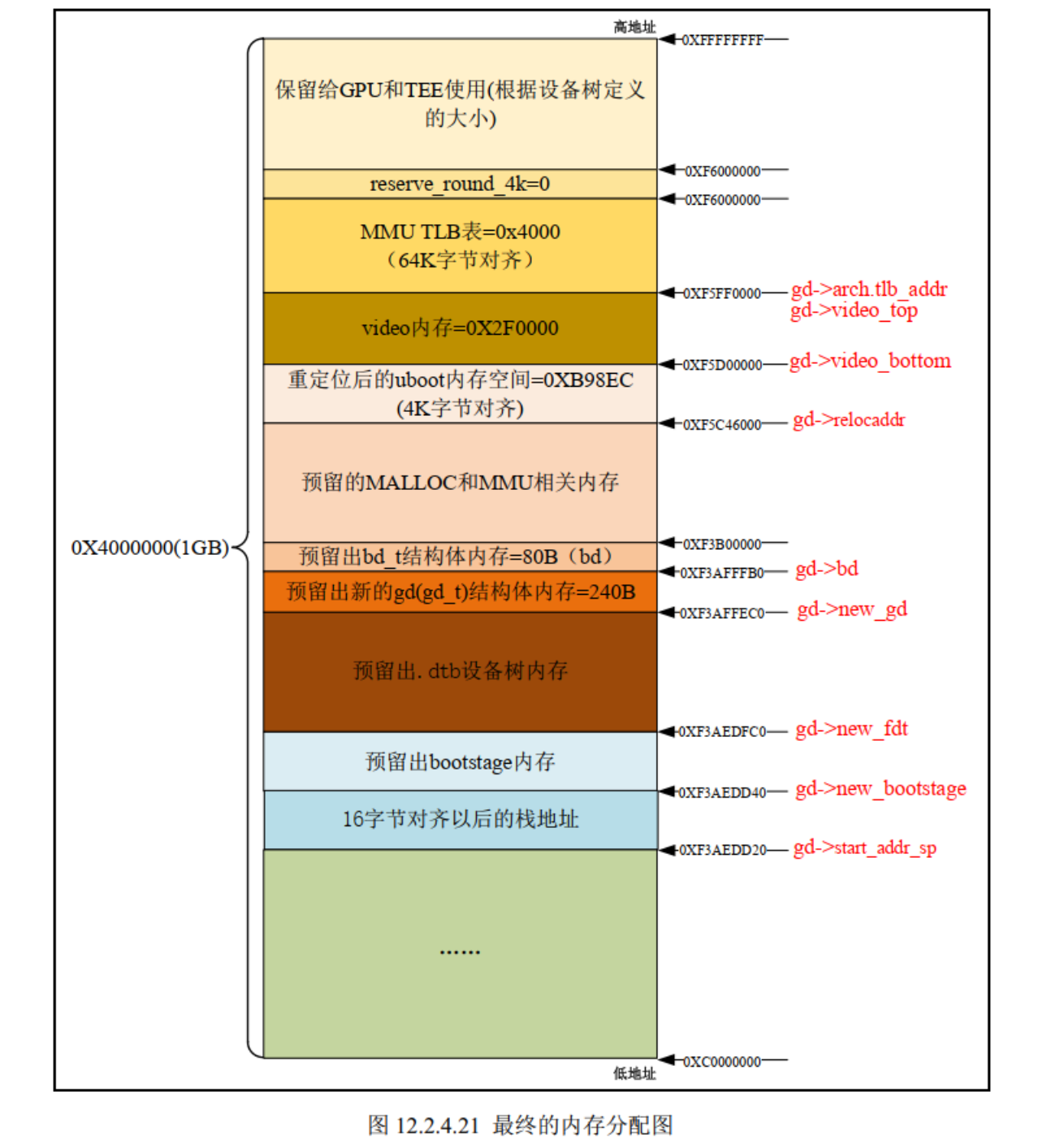
****

### Uboot Makefile 流程

****

****

### EMMC 内存分配

****

### vmlinux、 Image， zImage、 uImage 的区别



### 交叉编译器目录

/usr/local/arm/

### 图形配置命令

make menuconfig

**只要通过图形化界面修改了 Linux 内核配置，最好及时将其保存到  
stm32mp1\_atk\_defconfig 文件。因为图形化界面修改的配置只是暂时保存到.config 文件里面，一旦使用“make clean”清理工程，那么.config 文件就会被删除掉，所有的配置也就丢失了！**

### 软件后台运行

“./hello &”就是让 hello 在后台运行。

使用“ps”命令查看要关闭的软件 PID 是多少

在后台运行的软件可以使用“kill -9 pid(进程 ID)”命令来关闭掉，

### 开机自启动运行软件

进入根文件系统的时候会运行/etc/init.d/rcS 这个 shell 脚本，因此我们可以在这个脚本里面添加自启动相关内容。



### 文件解压

tar -vxjf buildroot-2020.02.6.tar.bz2

### 文件打包

#### -1. 使用 `tar` 命令创建 tar 压缩文件：

`tar` 命令可以创建 tar 压缩文件，通常使用 ".tar" 扩展名。例如，要将名为 "myfolder" 的目录打包成 "myfolder.tar" 文件，可以运行：

```

tar -cvf myfolder.tar myfolder

```

- `-c`: 创建新的 tar 文件。

- `-v`: 显示详细的操作信息（可选）。

- `-f`: 指定要创建的 tar 文件的名称。

#### 2. 使用 `zip` 命令创建 zip 压缩文件：

如果您想要创建 zip 压缩文件，可以使用 `zip` 命令。要将名为 "myfolder" 的目录打包成 "myfolder.zip" 文件，可以运行：

```

zip -r myfolder.zip myfolder

```

- `-r`: 递归地将目录及其内容添加到压缩文件中。

#### 3. 使用 `7z` 命令创建 7z 压缩文件：

如果您希望创建 7z 压缩文件，可以使用 `7z` 命令（需要安装 7-Zip 软件）。要将名为 "myfolder" 的目录打包成 "myfolder.7z" 文件，可以运行：

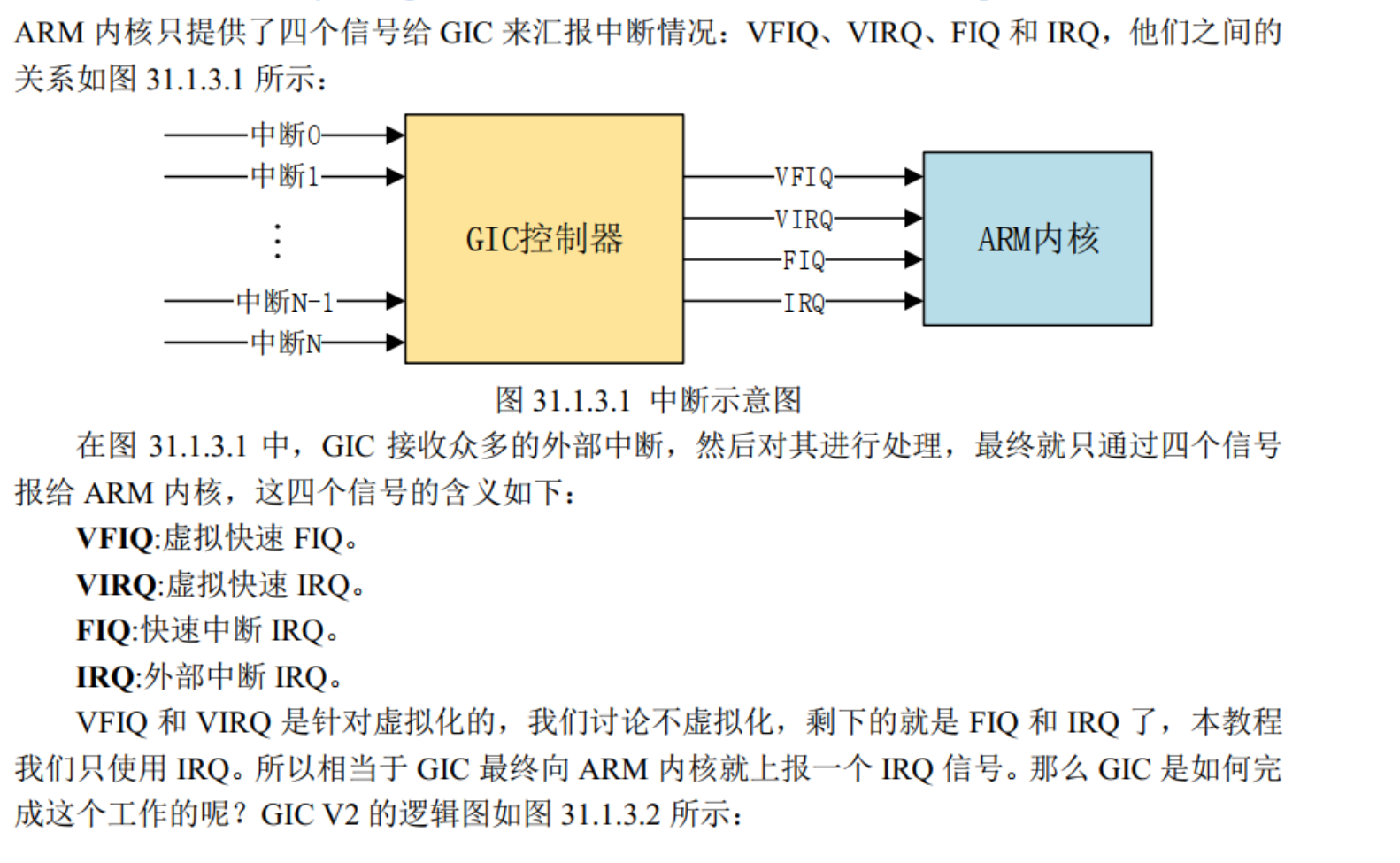
```

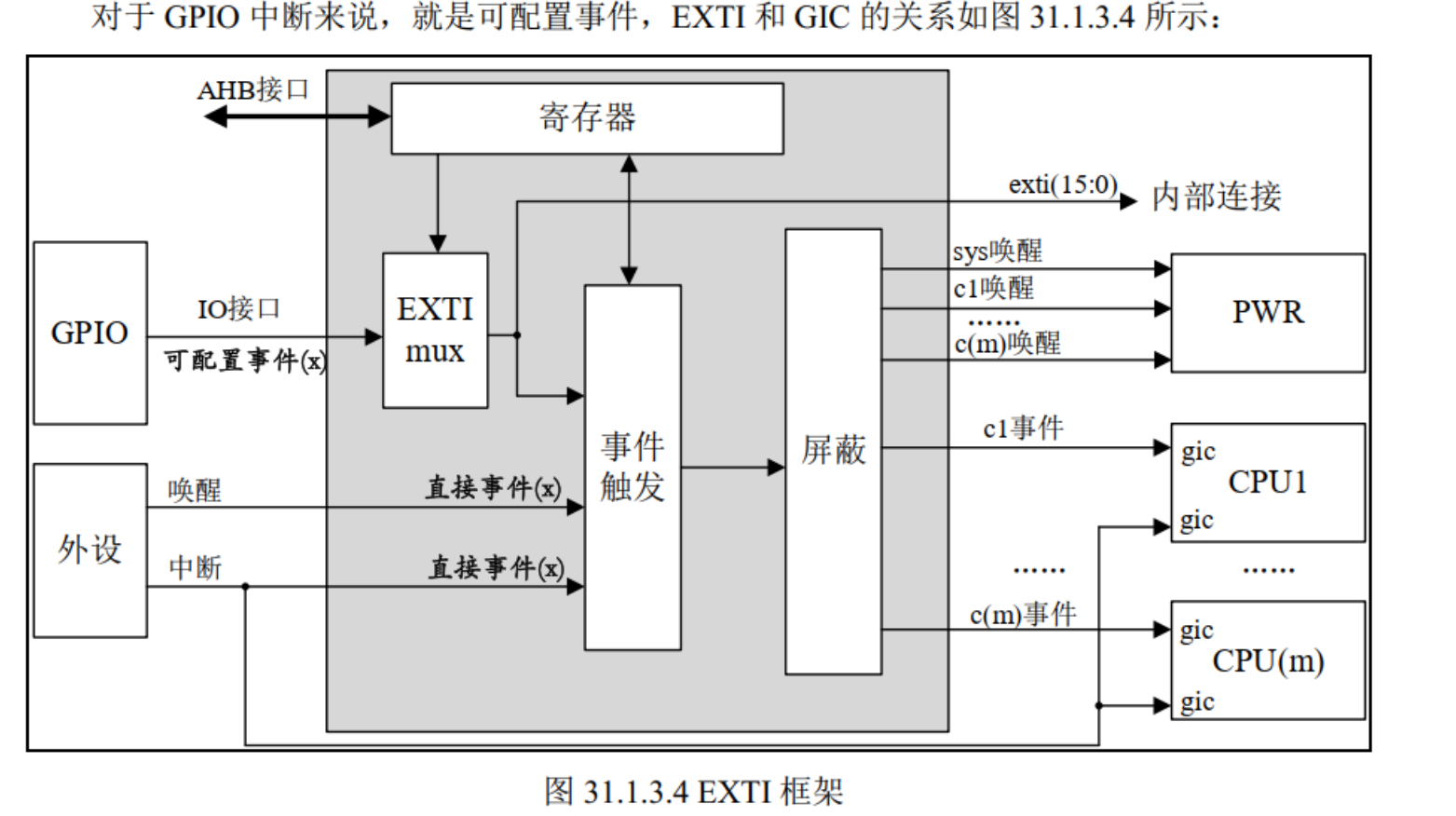
7z a myfolder.7z myfolder

```

- `a`: 添加文件/目录到压缩文件中。

### 中断





### Qt程序运行

|  |  |
| --- | --- |
| systemctl stop atk-qtapp-start.service  chmod 777 test  ./test -platform eglfs | //暂时关闭出厂系统桌面 //赋予测试文件权限 //以 eglfs 方式运行测试程序 |

./test -platform linuxfb //以 linuxfb 方式运行测试程序

Ls

|  |  |
| --- | --- |
| systemctl disable atk-qtapp-start.service  systemctl stop atk-qtapp-start.service  systemctl enable atk-qtapp-start.service | // 执行此项关闭 Qt 桌面服务，下次启动不生效 // 暂时停止 Qt 桌面服务，重启服务用 restart // 若前面 disable 了，想再开启，用 enable |
|  |  |

altbootcmd=run bootcmd

android\_mmc\_boot=mmc dev ${devnum};run android\_mmc\_splash;run android\_mmc\_fdt;run android\_mmc\_kernel;bootm ${kernel\_addr\_r} - ${fdt\_addr\_r};

android\_mmc\_fdt=if part start mmc ${devnum} dt\_${suffix} dt\_start &&part size mmc ${devnum} dt\_${suffix} dt\_size;then mmc read ${dtimg\_addr} ${dt\_start} ${dt\_size};dtimg getindex ${dtimg\_addr} ${board\_id} ${board\_rev} dt\_index;dtimg start ${dtimg\_addr} ${dt\_index} fdt\_addr\_r;fi

android\_mmc\_kernel=if part start mmc ${devnum} boot\_${suffix} boot\_start &&part size mmc ${devnum} boot\_${suffix} boot\_size;then mmc read ${kernel\_addr\_r} ${boot\_start} ${boot\_size};part nb mmc ${devnum} system\_${suffix} rootpart\_nb;env set bootargsroot=/dev/mmcblk${devnum}p${rootpart\_nb} androidboot.serialno=${serial#} androidboot.slot\_suffix=\_${suffix};fi

android\_mmc\_splash=if part start mmc ${devnum} splash splash\_start && part size mmc ${devnum} splash splash\_size;then mmc read ${splashimage} ${splash\_start} ${splash\_size};cls; bmp display ${splashimage} m m;fi

arch=arm

autoload=no

baudrate=115200

board=stm32mp1

board\_name=stm32mp157d-ed1

boot\_a\_script=load ${devtype} ${devnum}:${distro\_bootpart} ${scriptaddr} ${prefix}${script}; source ${scriptaddr}

boot\_device=mmc

boot\_efi\_binary=if fdt addr ${fdt\_addr\_r}; then bootefi bootmgr ${fdt\_addr\_r};else bootefi bootmgr ${fdtcontroladdr};fi;load ${devtype} ${devnum}:${distro\_bootpart} ${kernel\_addr\_r} efi/boot/bootarm.efi; if fdt addr ${fdt\_addr\_r}; then bootefi ${kernel\_addr\_r} ${fdt\_addr\_r};else bootefi ${kernel\_addr\_r} ${fdtcontroladdr};fi

boot\_extlinux=run scan\_m4fw;sysboot ${devtype} ${devnum}:${distro\_bootpart} any ${scriptaddr} ${prefix}${boot\_syslinux\_conf}

boot\_instance=1

boot\_m4fw=rproc init; rproc load 0 ${m4fw\_addr} ${filesize}; rproc start 0

boot\_net\_usb\_start=true

boot\_prefixes=/mmc1\_

boot\_script\_dhcp=boot.scr.uimg

boot\_scripts=boot.scr.uimg boot.scr

boot\_syslinux\_conf=extlinux/stm32mp157d-atk\_extlinux.conf

boot\_targets=mmc1

bootargs=root=/dev/nfs nfsroot=192.168.10.100:/home/book/alientek\_linux/nfs/rootfs,proto=tcp rw ip=192.168.10.10:192.168.10.100:192.168.10.1:255.255.255.0::eth0:off

bootcmd=ext4load mmc 1:2 c2000000 uImage;ext4load mmc 1:2 c4000000 stm32mp157d-atk.dtb;bootm c2000000 - c4000000

bootcmd\_android=env set mmc\_boot run android\_mmc\_boot;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} = serial || test ${boot\_device} = usb;then stm32prog ${boot\_device} ${boot\_instance}; else run env\_check;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; fi;if test ${boot\_device} = nor;then env set boot\_targets mmc0; fi;run distro\_bootcmd;fi;

bootcmd\_ubifs0=devnum=0; run ubifs\_boot

bootcount=1947

bootdelay=1

bootfstype=ext4

cpu=armv7

devplist=2

distro\_bootcmd=for target in ${boot\_targets}; do run bootcmd\_${target}; done

dtimg\_addr=0xc4500000

efi\_dtb\_prefixes=/ /dtb/ /dtb/current/

env\_check=env exists env\_ver || env set env\_ver ${ver};if env info -p -d -q; then env save; fi;if test "$env\_ver" != "$ver"; then echo "\*\*\* Warning: old environment ${env\_ver}"; echo '\* set default: env default -a; env save; reset'; echo '\* update current: env set env\_ver ${ver}; env save';fi;

env\_ver=U-Boot 2020.01-stm32mp-r1 (Dec 30 2022 - 12:24:48 +0800)

ethact=ethernet@5800a000

ethaddr=b8:ae:1d:01:01:00

fdt\_addr\_r=0xc4000000

fdtcontroladdr=f3aee1a0

fdtfile=stm32mp157d-ed1.dtb

fileaddr=c4100000

filesize=b7f

gatewayip=192.168.10.1

ipaddr=192.168.10.10

kernel\_addr\_r=0xc2000000

lcd\_bmp=alientek\_1024x600.bmp

lcd\_id=2

load\_efi\_dtb=load ${devtype} ${devnum}:${distro\_bootpart} ${fdt\_addr\_r} ${prefix}${efi\_fdtfile}

loadaddr=0xc2000000

m4fw\_addr=0xc2000000

m4fw\_name=rproc-m4-fw.elf

mmc\_boot=if mmc dev ${devnum}; then devtype=mmc; run scan\_dev\_for\_boot\_part; fi

netmask=255.255.255.0

partitions=name=ssbl,size=2M;name=bootfs,size=64MB,bootable;name=vendorfs,size=16M;name=rootfs,size=746M;name=userfs,size=-

pxefile\_addr\_r=0xc4200000

ramdisk\_addr\_r=0xc4400000

scan\_dev\_for\_boot=run select\_lcd\_id;echo Scanning ${devtype} ${devnum}:${distro\_bootpart}...; for prefix in ${boot\_prefixes}; do run scan\_dev\_for\_extlinux; run scan\_dev\_for\_scripts; done;run scan\_dev\_for\_efi;

scan\_dev\_for\_boot\_part=part list ${devtype} ${devnum} -bootable devplist; env exists devplist || setenv devplist 1; for distro\_bootpart in ${devplist}; do if fstype ${devtype} ${devnum}:${distro\_bootpart} bootfstype; then run scan\_dev\_for\_boot; fi; done; setenv devplist

scan\_dev\_for\_efi=setenv efi\_fdtfile ${fdtfile}; if test -z "${fdtfile}" -a -n "${soc}"; then setenv efi\_fdtfile ${soc}-${board}${boardver}.dtb; fi; for prefix in ${efi\_dtb\_prefixes}; do if test -e ${devtype} ${devnum}:${distro\_bootpart} ${prefix}${efi\_fdtfile}; then run load\_efi\_dtb; fi;done;if test -e ${devtype} ${devnum}:${distro\_bootpart} efi/boot/bootarm.efi; then echo Found EFI removable media binary efi/boot/bootarm.efi; run boot\_efi\_binary; echo EFI LOAD FAILED: continuing...; fi; setenv efi\_fdtfile

scan\_dev\_for\_extlinux=if test -e ${devtype} ${devnum}:${distro\_bootpart} ${prefix}${boot\_syslinux\_conf}; then echo Found ${prefix}${boot\_syslinux\_conf}; run boot\_extlinux; echo SCRIPT FAILED: continuing...; fi

scan\_dev\_for\_scripts=for script in ${boot\_scripts}; do if test -e ${devtype} ${devnum}:${distro\_bootpart} ${prefix}${script}; then echo Found U-Boot script ${prefix}${script}; run boot\_a\_script; echo SCRIPT FAILED: continuing...; fi; done

scan\_m4fw=if test -e ${devtype} ${devnum}:${distro\_bootpart} ${m4fw\_name};then echo Found M4 FW $m4fw\_name; if load ${devtype} ${devnum}:${distro\_bootpart} ${m4fw\_addr} ${m4fw\_name}; then run boot\_m4fw; fi; fi;

scriptaddr=0xc4100000

select\_lcd\_id=ext4load mmc ${devnum}:${distro\_bootpart} c4300000 ${lcd\_bmp};bmp display c4300000;

serial#=0036002A3232511937343637

serverip=192.168.10.100

soc=stm32mp

splashimage=0xc4300000

suffix=a

ubifs\_boot=env exists bootubipart || env set bootubipart UBI; env exists bootubivol || env set bootubivol boot; if ubi part ${bootubipart} && ubifsmount ubi${devnum}:${bootubivol}; then devtype=ubi; run scan\_dev\_for\_boot; fi

usb\_boot=usb start; if usb dev ${devnum}; then devtype=usb; run scan\_dev\_for\_boot\_part; fi

vendor=st

ver=U-Boot 2020.01-stm32mp-r1 (Jan 29 2024 - 02:48:13 -0500)

Environment size: 6517/8187 bytes

STM32MP> printenv

altbootcmd=run bootcmd

android\_mmc\_boot=mmc dev ${devnum};run android\_mmc\_splash;run android\_mmc\_fdt;run android\_mmc\_kernel;bootm ${kernel\_addr\_r} - ${fdt\_addr\_r};

android\_mmc\_fdt=if part start mmc ${devnum} dt\_${suffix} dt\_start &&part size mmc ${devnum} dt\_${suffix} dt\_size;then mmc read ${dtimg\_addr} ${dt\_start} ${dt\_size};dtimg getindex ${dtimg\_addr} ${board\_id} ${board\_rev} dt\_index;dtimg start ${dtimg\_addr} ${dt\_index} fdt\_addr\_r;fi

android\_mmc\_kernel=if part start mmc ${devnum} boot\_${suffix} boot\_start &&part size mmc ${devnum} boot\_${suffix} boot\_size;then mmc read ${kernel\_addr\_r} ${boot\_start} ${boot\_size};part nb mmc ${devnum} system\_${suffix} rootpart\_nb;env set bootargsroot=/dev/mmcblk${devnum}p${rootpart\_nb} androidboot.serialno=${serial#} androidboot.slot\_suffix=\_${suffix};fi

android\_mmc\_splash=if part start mmc ${devnum} splash splash\_start && part size mmc ${devnum} splash splash\_size;then mmc read ${splashimage} ${splash\_start} ${splash\_size};cls; bmp display ${splashimage} m m;fi

arch=arm

autoload=no

baudrate=115200

board=stm32mp1

board\_name=stm32mp157d-atk

boot\_a\_script=load ${devtype} ${devnum}:${distro\_bootpart} ${scriptaddr} ${prefix}${script}; source ${scriptaddr}

boot\_device=mmc

boot\_efi\_binary=if fdt addr ${fdt\_addr\_r}; then bootefi bootmgr ${fdt\_addr\_r};else bootefi bootmgr ${fdtcontroladdr};fi;load ${devtype} ${devnum}:${distro\_bootpart} ${kernel\_addr\_r} efi/boot/bootarm.efi; if fdt addr ${fdt\_addr\_r}; then bootefi ${kernel\_addr\_r} ${fdt\_addr\_r};else bootefi ${kernel\_addr\_r} ${fdtcontroladdr};fi

boot\_extlinux=run scan\_m4fw;sysboot ${devtype} ${devnum}:${distro\_bootpart} any ${scriptaddr} ${prefix}${boot\_syslinux\_conf}

boot\_instance=1

boot\_m4fw=rproc init; rproc load 0 ${m4fw\_addr} ${filesize}; rproc start 0

boot\_net\_usb\_start=true

boot\_prefixes=/mmc1\_

boot\_script\_dhcp=boot.scr.uimg

boot\_scripts=boot.scr.uimg boot.scr

boot\_syslinux\_conf=extlinux/stm32mp157d-atk\_extlinux.conf

boot\_targets=mmc1

bootcmd=run bootcmd\_stm32mp

bootcmd\_android=env set mmc\_boot run android\_mmc\_boot;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} = serial || test ${boot\_device} = usb;then stm32prog ${boot\_device} ${boot\_instance}; else run env\_check;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; fi;if test ${boot\_device} = nor;then env set boot\_targets mmc0; fi;run distro\_bootcmd;fi;

bootcmd\_ubifs0=devnum=0; run ubifs\_boot

bootcount=1950

bootdelay=1

bootfstype=ext4

cpu=armv7

devplist=2

distro\_bootcmd=for target in ${boot\_targets}; do run bootcmd\_${target}; done

dtimg\_addr=0xc4500000

efi\_dtb\_prefixes=/ /dtb/ /dtb/current/

env\_check=env exists env\_ver || env set env\_ver ${ver};if env info -p -d -q; then env save; fi;if test "$env\_ver" != "$ver"; then echo "\*\*\* Warning: old environment ${env\_ver}"; echo '\* set default: env default -a; env save; reset'; echo '\* update current: env set env\_ver ${ver}; env save';fi;

env\_ver=U-Boot 2020.01-stm32mp-r1 (Dec 30 2022 - 12:24:48 +0800)

fdt\_addr\_r=0xc4000000

fdtcontroladdr=f3aedfc0

fdtfile=stm32mp157d-atk.dtb

fileaddr=c4100000

filesize=b7f

kernel\_addr\_r=0xc2000000

lcd\_bmp=alientek\_1024x600.bmp

lcd\_id=2

load\_efi\_dtb=load ${devtype} ${devnum}:${distro\_bootpart} ${fdt\_addr\_r} ${prefix}${efi\_fdtfile}

loadaddr=0xc2000000

m4fw\_addr=0xc2000000

m4fw\_name=rproc-m4-fw.elf

mmc\_boot=if mmc dev ${devnum}; then devtype=mmc; run scan\_dev\_for\_boot\_part; fi

partitions=name=ssbl,size=2M;name=bootfs,size=64MB,bootable;name=vendorfs,size=16M;name=rootfs,size=746M;name=userfs,size=-

pxefile\_addr\_r=0xc4200000

ramdisk\_addr\_r=0xc4400000

scan\_dev\_for\_boot=run select\_lcd\_id;echo Scanning ${devtype} ${devnum}:${distro\_bootpart}...; for prefix in ${boot\_prefixes}; do run scan\_dev\_for\_extlinux; run scan\_dev\_for\_scripts; done;run scan\_dev\_for\_efi;

scan\_dev\_for\_boot\_part=part list ${devtype} ${devnum} -bootable devplist; env exists devplist || setenv devplist 1; for distro\_bootpart in ${devplist}; do if fstype ${devtype} ${devnum}:${distro\_bootpart} bootfstype; then run scan\_dev\_for\_boot; fi; done; setenv devplist

scan\_dev\_for\_efi=setenv efi\_fdtfile ${fdtfile}; if test -z "${fdtfile}" -a -n "${soc}"; then setenv efi\_fdtfile ${soc}-${board}${boardver}.dtb; fi; for prefix in ${efi\_dtb\_prefixes}; do if test -e ${devtype} ${devnum}:${distro\_bootpart} ${prefix}${efi\_fdtfile}; then run load\_efi\_dtb; fi;done;if test -e ${devtype} ${devnum}:${distro\_bootpart} efi/boot/bootarm.efi; then echo Found EFI removable media binary efi/boot/bootarm.efi; run boot\_efi\_binary; echo EFI LOAD FAILED: continuing...; fi; setenv efi\_fdtfile

scan\_dev\_for\_extlinux=if test -e ${devtype} ${devnum}:${distro\_bootpart} ${prefix}${boot\_syslinux\_conf}; then echo Found ${prefix}${boot\_syslinux\_conf}; run boot\_extlinux; echo SCRIPT FAILED: continuing...; fi

scan\_dev\_for\_scripts=for script in ${boot\_scripts}; do if test -e ${devtype} ${devnum}:${distro\_bootpart} ${prefix}${script}; then echo Found U-Boot script ${prefix}${script}; run boot\_a\_script; echo SCRIPT FAILED: continuing...; fi; done

scan\_m4fw=if test -e ${devtype} ${devnum}:${distro\_bootpart} ${m4fw\_name};then echo Found M4 FW $m4fw\_name; if load ${devtype} ${devnum}:${distro\_bootpart} ${m4fw\_addr} ${m4fw\_name}; then run boot\_m4fw; fi; fi;

scriptaddr=0xc4100000

select\_lcd\_id=ext4load mmc ${devnum}:${distro\_bootpart} c4300000 ${lcd\_bmp};bmp display c4300000;

serial#=0036002A3232511937343637

serverip=192.168.1.1

soc=stm32mp

splashimage=0xc4300000

suffix=a

ubifs\_boot=env exists bootubipart || env set bootubipart UBI; env exists bootubivol || env set bootubivol boot; if ubi part ${bootubipart} && ubifsmount ubi${devnum}:${bootubivol}; then devtype=ubi; run scan\_dev\_for\_boot; fi

usb\_boot=usb start; if usb dev ${devnum}; then devtype=usb; run scan\_dev\_for\_boot\_part; fi

vendor=st

ver=U-Boot 2020.01-stm32mp-r1 (Dec 30 2022 - 12:24:48 +0800)

\