# Uboot学习笔记

## 学习目标

uboot基本概念

uboot编译

uboot命令

Uboot 编译流程（顶层makefile分析）

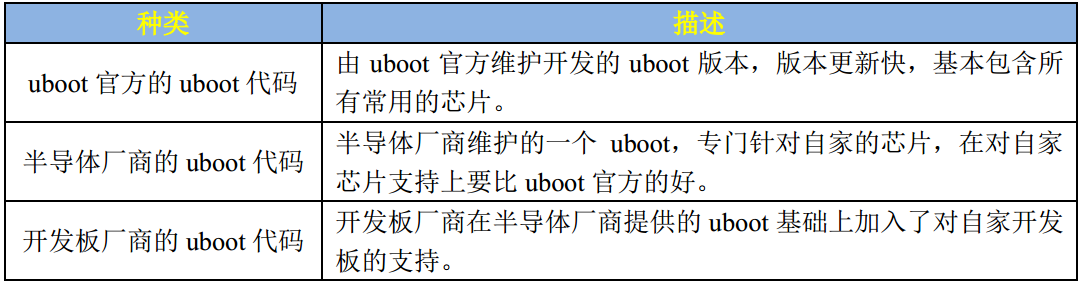
uboot启动流程

uboot移植

## Uboot基本概念

uboot是一个裸机程序，BootLoader，最主要的工作是启动linux内核，这段 bootloader程序会先初始化 DDR等外设，然后将 Linux内核从 flash(NAND，NOR FLASH， SD， MMC 等)拷贝到 DDR 中，最后启动 Linux 内核，bootloader 和 Linux 内核的关系就跟 PC 上的 BIOS 和 Windows 的关系一样， bootloader 就相当于 BIOS。有很多现成的 bootloader 软件可以使用，比如 U-Boot、 vivi、 RedBoot 等等，其中以 U-Boot 使用最为广泛。

uboot 的全称是 Universal Boot Loader， uboot 是一个遵循 GPL 协议的开源软件， uboot 是一个裸机代码，可以看作是一个裸机综合例程。现在的 uboot 已经支持液晶屏、网络、 USB 等高级功能。 uboot 官网为 <http://www.denx.de/wiki/U-Boot/>



## Uboot编译

编译前先安装ncurses 库：apt-get install libncurses5-dev

编译命令：

make ARCH**=**arm CROSS\_COMPILE**=**arm**-**linux**-**gnueabihf**-** distclean

make ARCH**=**arm CROSS\_COMPILE**=**arm**-**linux**-**gnueabihf**-** (加空格)  
mx6ull\_14x14\_ddr512\_emmc\_defconfig

make V**=**1 ARCH**=**arm CROSS\_COMPILE**=**arm**-**linux**-**gnueabihf**- -**j2

几个参数含义：

ARCH**=**arm为指定架构为arm；

CROSS\_COMPILE为指定编译器；

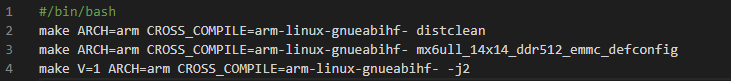
distclean表示清理工程；

mx6ull\_14x14\_ddr512\_emmc\_defconfig指定根据此配置生成配置文件；

V=1用于设置编译过程的信息输出级别；

-j 用于设置主机使用多少线程编译uboot；

**脚本方法编译**



**烧录方法：**

Mfgtool工具烧录：

连接开发板OTG接口，使用相对应得Mfgtool工具进行烧写

烧录原理：

Mfgtool会先向板子DDR里下载一个系统，然后启动系统，最后将需要烧写的系统通过脚本烧写至EMMC或SD卡内

Mfgtool工具文件目录

\mfgtool\Profiles\Linux\OS Firmware\firmware 存放需要烧录的uboot zImage 设备树 文件系统等文件

mfgtool\Profiles\Linux\OS Firmware\files 临时启动的uboot zImage 设备树 文件系统等

## Uboot命令

**信息查询命令**：

bdinfo:查看板子信息，DRAM起始地址和大小、启动参数保存起始地址、波特率、

sp(堆栈指针)起始地址等信息；

printenv:查看环境变量信息；

version:查看uboot的版本号；

**环境变量操作命令**：

**修改环境变量**:

Setenv:用于设置或者修改环境变量的值

用法：setenv 命令 值 或者 setenv 命令 值1 值2

saveenv:将修改后的环境变量保存到flash中

用法：setenv bootdelay 5

saveenv

修改的环境变量值可能会有空格， 比如 bootcmd、 bootargs 等， 这个时候环境变

量值就得用单引号括起来，比如下面修改环境变量 bootcmd 的值：

setenv bootcmd 'console=ttymxc0,115200 root=/dev/mmcblk1p2 rootwait rw'  
 saveenv

**新建环境变量**:

setenv 也可以用于新建环境变量，用法就是修改环境变量一样。

**删除环境变量**:

setenv 也可以用于删除环境变量，用法就是修改环境变量时赋值为空即可。

**内存操作命令**：

md:用于显示内存值

用法：md[.b, .w, .l] address [# of objects]

命令中的[.b .w .l]对应 byte、word和long，[# of objects]表示要查看的数据长度（16进制）；

例如：md.b 80000000 14

查看以0X80000000开始的 20 个字节的内存值

nm:用于修改指定地址的内存值

用法：nm[.b, .w, .l] address

例如：nm.l 80000000

输入要修改的值，输入完成输入回车，让后输入q即可退出

mm:用于修改指定地址的内存值，与nm的区别是，mm修改内存值时会自增

用法：与nm一样

mw:用于使用一个指定的数据填充一段内存

用法：mw[.b, .w, .l] address value [count]; count 是填充的长度

例如：mw.l 80000000 0A0A0A0A 10 使用.l格式将以0X80000000为起始地址的0x10个内存块(0x10 \* 4=64字节)填充为0X0A0A0A0A

cp:数据拷贝

用法：cp[.b, .w, .l] source target count; source为源地址，target为目的地址，count

为拷贝的长度

例如：cp.l 80000000 80000100 10 使用.l格式将0x80000000处的地址拷贝到 0X80000100处，长度为0x10个内存块(0x10 \* 4=64 个字节)

cmp:比较两段内存的数据是否相等

用法：cmp[.b, .w, .l] addr1 addr2 count

**网络操作命令**：

ping:测试网络

dhcp:动态获取IP

nfs: nfs(Network File System)网络文件系统，通过 nfs 可以在计算机之间通过网络来分享资源

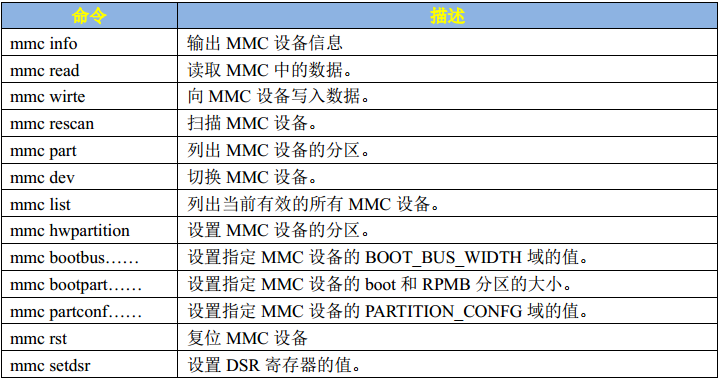
用法：nfs [loadAddress] [[hostIPaddr:]bootfilename]；loadAddress 是要保存的 DRAM 地址， [[hostIPaddr:]bootfilename]是要下载的文件地址

需要在ubuntu主机上开启nfs服务

tftp: 用于通过网络下载东西到DRAM中，tftp命令使用的TFTP协议

用法：tftpboot [loadAddress] [[hostIPaddr:]bootfilename]

**EMMC和SD卡操作命令**：



**FAT格式文件系统操作命令**：

文件操作相关的命令有： fatinfo、 fatls、 fstype、 fatload 和 fatwrite

**FAT格式文件系统操作命令**：

uboot有ext2和ext4这两种格式的文件系统的操作命令，常用的就四个命令，分别为：ext2load、ext2ls、ext4load、ext4ls 和ext4write。这些命令的含义和使用与fatload、fatls 和 fatwrit一样，只是ext2 和ext4 都是针对ext文件系统的。

**BOOT操作命令**：

bootz:用于启动linux内核，用于自动zImage镜像文件

用法：bootz [addr [initrd[:size]] [fdt]]；addr是Linux镜像文件在DRAM中的位置，initrd 是initrd文件在DRAM中的地址，如果不使用initrd的话使用‘-’代替即可，fdt就是设备树文件在DRAM中的地址。

bootm: 和bootz功能类似，但是bootm用于启动uImage镜像文件

用法：不使用设备树bootm addr

使用设备树bootm [addr [initrd[:size]] [fdt]]

boot:也是用来启动Linux内核的，boot会读取环境变量bootcmd来启动Linux系统

例如：setenv bootcmd 'fatload mmc 1:1 80800000 zImage; fatload mmc 1:1 83000000 imx6ullalientek\_emmc.dtb; bootz 80800000 - 83000000'

savenev

boot

**其他常用命令**：

reset:复位重启uboot

go:用于跳到指定的地址处执行应用

用法：go addr [arg ...]

run:用于运行环境变量中定义的命令

mtest:一个简单的内存读写测试命令，以用来测试自己开发板上的DDR

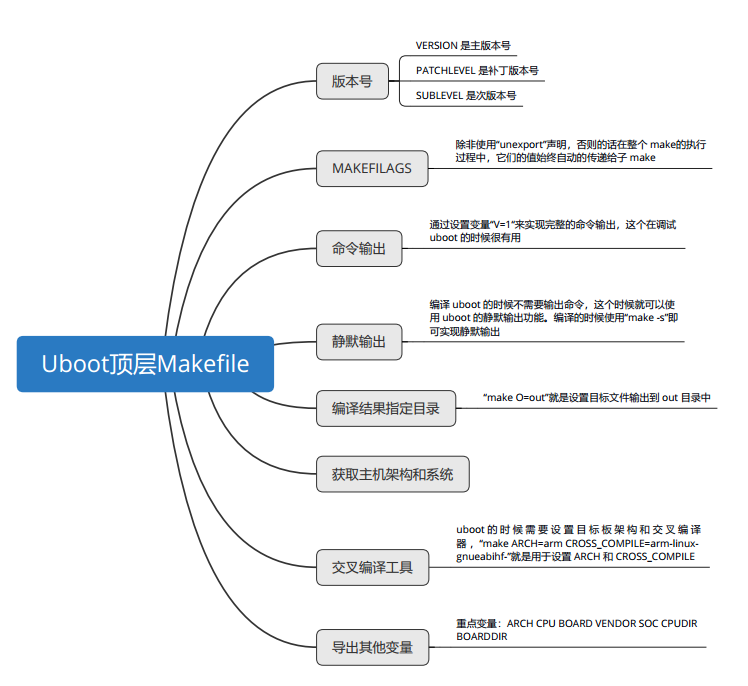
用法：mtest [start [end [pattern [iterations]]]]；start是要测试的DRAM开始地址，end 是结束地址，比如我们测试0X80000000~0X8000100这段内存，输入“mtest 80000000 80001000”

## Uboot 编译流程（顶层makefile分析）

Uboot源码目录



顶层Makefile分析：



ARCH=arm

CPU=armv7

BOARD=mx6ullevk

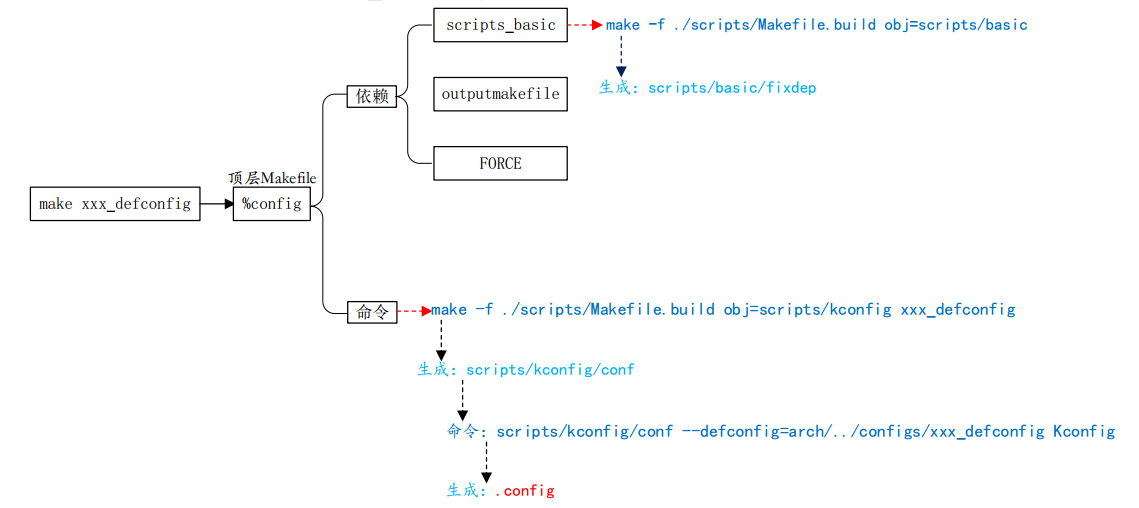
VENDOR=freescale

SOC=mx6

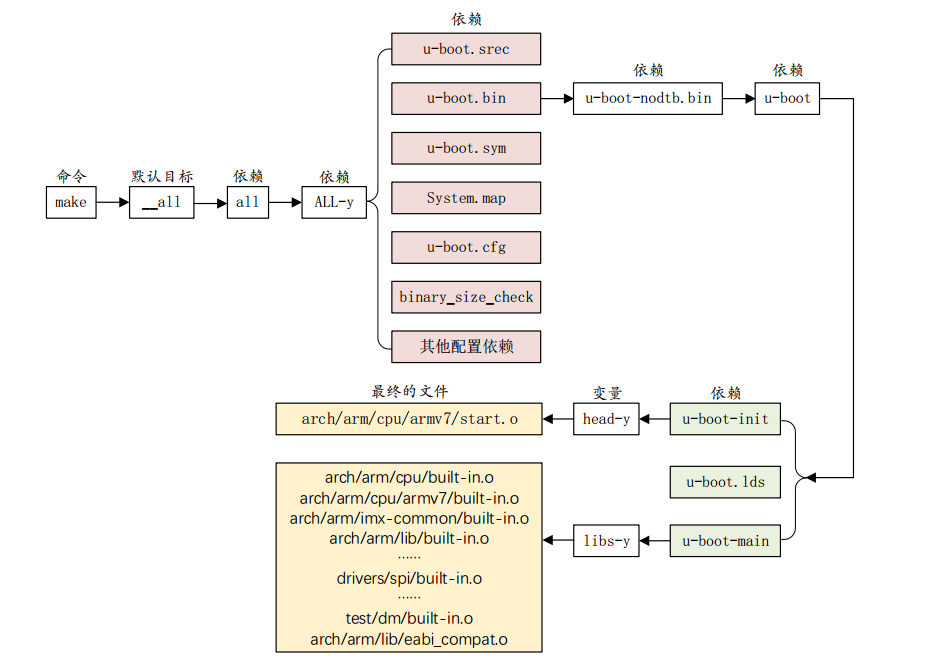
CPUDIR=arch/arm/cpu/armv7

BOARDDIR=freescale/mx6ullevk

mx6ull\_14x14\_ddr512\_emmc\_defconfig过程：

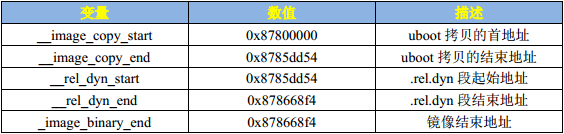


make过程:



## Uboot 启动流程

链接脚本 **u-boot.lds** 详解:

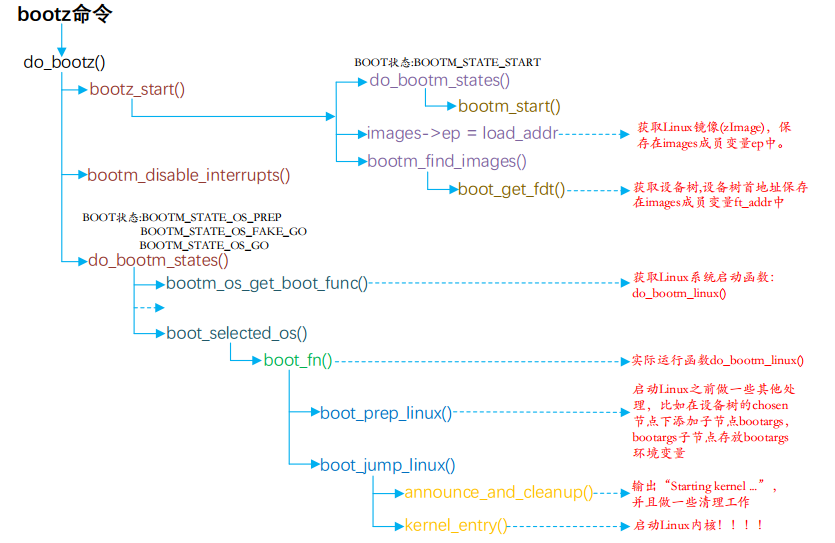




启动流程：



bootz 启动linux内核流程：

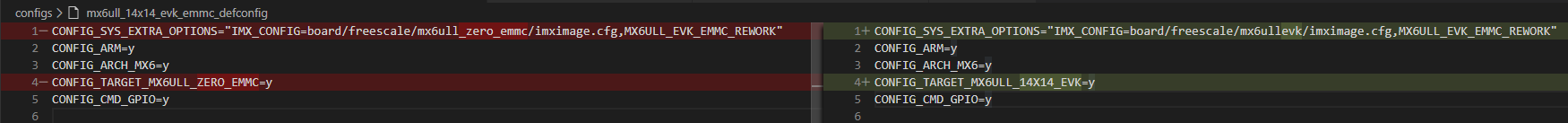


## Uboot移植

1. 添加开发板默认配置文件

Uboot源码configs 目录下创建默认配置文件mx6ull\_zero\_emmc\_defconfig, 复制mx6ull\_14x14\_evk\_emmc\_defconfig内容做下面修改：

红色新配置文件修改的内容

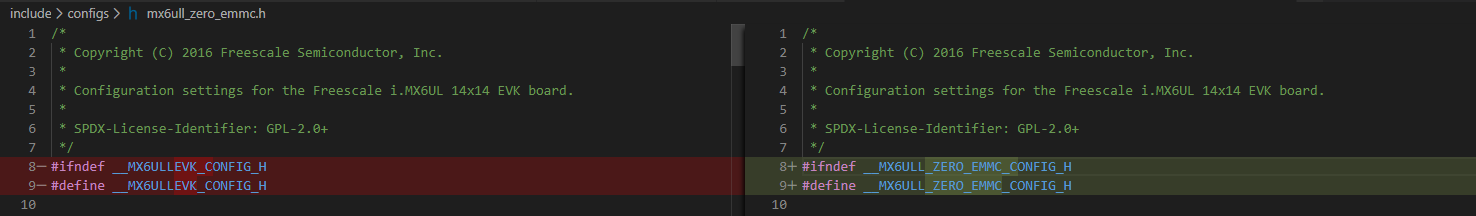


1. 添加开发板对应的头文件

在 目 录 include/configs新建文件mx6ull\_zero\_emmc.h，复制include/configs/mx6

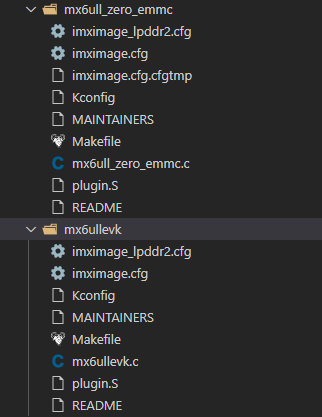
ullevk.h内容到 mx6ull\_zero\_emmc.h，做下面修改：

绿色为新文件修改的内容

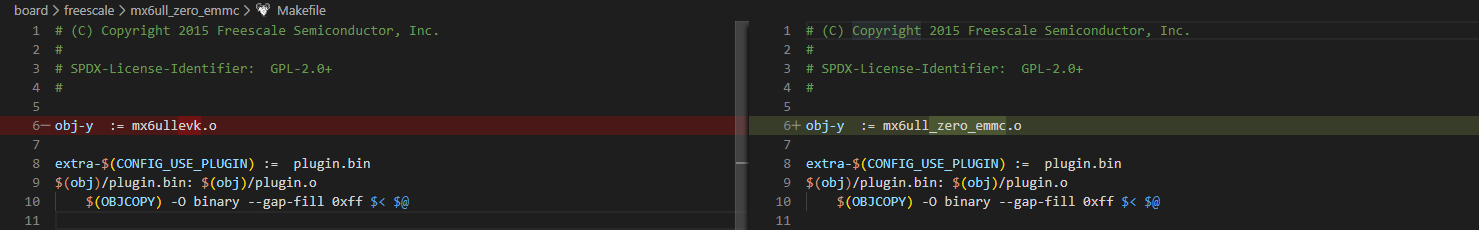


1. 添加开发板对应的板级文件夹

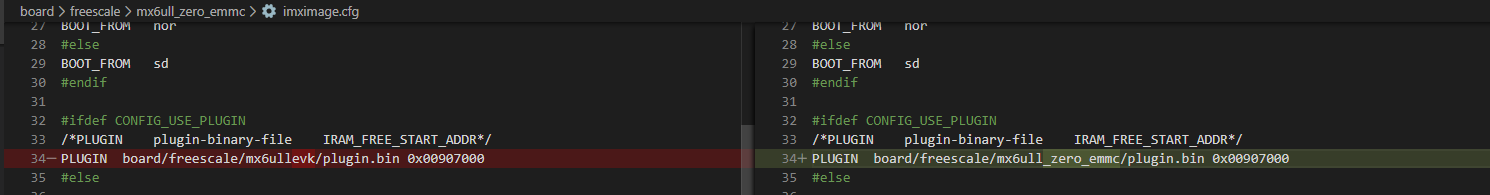
在Uboot源码board/freescale中新建mx6ull\_zero\_emmc文件夹，复制mx6ullevk文件夹所有文件到新建文件夹中，且修改mx6ullevk.c文件名，如下图：



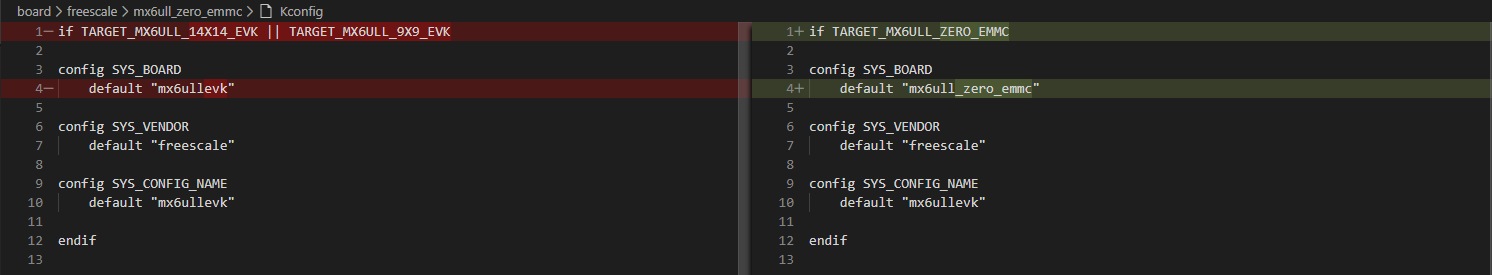
对mx6ull\_zero\_emmc文件夹里Makefile文件做如下修改:



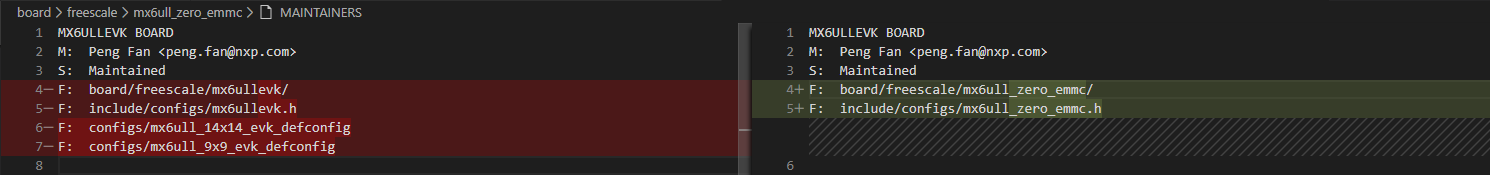
对mx6ull\_zero\_emmc文件夹里imximage.cfg文件做如下修改:



对mx6ull\_zero\_emmc文件夹里Kconfig文件做如下修改:

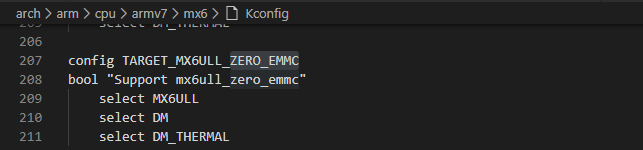


对mx6ull\_zero\_emmc文件夹里MAINTAINERS文件做如下修改:



1. 修改 U-Boot 图形界面配置文件

改文件arch/arm/cpu/armv7/mx6/Kconfig，在 207 行加入如下内容：



1. 编译

创建脚本进行编译

