# E9V2 系统烧写

## 1.1 使用官方工具烧写

烧写工具：Mfgtools-NEW-EMMC\_V6

烧写文件：uboot.bin;uImage;roorfs.tgz。

烧写配置：

硬件：拨码开关设为usb启动，值为1001。网口方向为上，向上拨码为1。

软件：镜像文件改名为uboot.bin;uImage;roorfs.tgz。

存放目录Mfgtools-NEW-EMMC\_V6\Profiles\MX6Q Linux Update\OS Firmware\files\linux

cfg配置如下：不用改动。

[profiles]

chip = MX6Q Linux Update

[platform]

board = SabreSD

[LIST]

name =Linux-SabreSD-eMMC

烧写：

打开mfgtool ，检测到硬件设备，点击start,进度条变为绿色成功。先点击down,后点击exit。烧写成功

## 配置内核支持nfs

1、配置网络部分，主要是使能CONFIG\_IP\_PNP以在2中能够看到Root file system on NFS选项  
Networking support   
Networking options   
TCP/IP networking  
[\*]IP: kernel level autoconfiguration  
[\*] IP: DHCP support  
[\*] IP: BOOTP support  
  
2、配置开启nfs服务  
File systems --->   
Network File Systems --->  
<\*> NFS client support   
[\*] NFS client support for NFS version 3

[\*] NFS client support for the NFSv3 ACL protocol extension   
[\*] NFS client support for NFS version 4 (EXPERIMENTAL)   
[\*] NFS client support for NFSv4.1 (DEVELOPER ONLY)   
[\*] Root file system on NFS

<\*> NFS server support

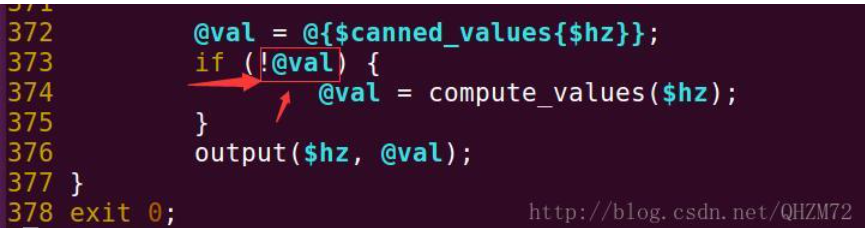
## 1.3 重新编译内核

编译器使用gcc4.6-linaro

修改内核build.sh编译器路径为编译器实际安装路径，注意大小写。采用四核编译make -j4.

编译内核错误提示2：recipe for target 'kernel/timeconst.h' failed

查找资料后发现：因为在新版本的(Linux) 上编译老版本的内核版本出现的错误（新版主机linux内核是4.10，被编译版本的内核是3.0的，好像是这么回事！），解决办法是：将kernel/timeconst.pl中第373行的defined()去掉，去掉后如下



## 1.4 nfs挂载网络文件夹

1、使用老的ubuntu14，32位 系统挂载nfs

mount -t nfs -o nolock,vers=2 192.168.1.111:/home/chao/work /mnt

挂载成功

2、使用新的ubuntu18，64位 系统挂载nfs 需要如下操作：

设置Ubuntu18.04系统中的nfs服务支持协议2，修改nfs配置文件

sudo gedit /etc/default/nfs-kernel-server,在文件末尾加入一句：RPCNFSDOPTS="--nfs-version 2,3,4 --debug --syslog"。

原因是从Ubuntu17.04开始，nfs默认只支持协议3和协议4，而kernel中默认支持协议2，所以才会出现挂载失败的情况。

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

mount -t nfs -o nolock,vers=2 192.168.1.118:/home/chao/work /mnt

## 1.5 uboot网络设置

##### param setting #####

[1]setting nfs args

[2]setting tftp args

[3]setting display args

[4]setting default boot

[s]save setting

[q]quit

Enter your selection: 1

--------setting nfs args--------

Enter the PC IP address:(xxx.xxx.xxx.xxx)

192.168.1.118

Enter the IMX6 IP address:(xxx.xxx.xxx.xxx)

192.168.1.6

Enter the IMX6 MAC address:(xx:xx:xx:xx:xx:xx)

12:32:43:34:21:43

Enter NFS directory:(eg: /opt/EmbedSky/root\_nfs)

/home/chao/work

save setting ? (y/n) :

y

Saving Environment to MMC...

Writing to MMC(3)... done

##### param setting #####

[1]setting nfs args

[2]setting tftp args

[3]setting display args

[4]setting default boot

[s]save setting

[q]quit

Enter your selection: 2

--------setting tftp args--------

Enter the TFTP Server(PC) IP address:(xxx.xxx.xxx.xxx)

192.168.1.118

Enter the IMX6 IP address:(xxx.xxx.xxx.xxx)

192.168.1.7

Enter the GateWay IP:(xxx.xxx.xxx.xxx)

192.168.1.1

Enter the bootloader (u-boot or bootimage) image name:

u-boot.bin

Enter the LOGO image name:

Enter the kernel image name:

uImage

Enter the root image name:

rootfs

save setting ? (y/n) :

y

Saving Environment to MMC...

Writing to MMC(3)... done

##### param setting #####

[1]setting nfs args

[2]setting tftp args

[3]setting display args

[4]setting default boot

[s]save setting

[q]quit

Enter your selection: s

Saving Environment to MMC...

Writing to MMC(3)... done

MX6Q SABRESD U-Boot > ping 192.168.1.118

PHY indentify @ 0x0 = 0x001cc915

mx6\_rgmii\_rework fin to rework network!

FEC: Link is Up 796d

Using FEC0 device

host 192.168.1.118 is alive

ping 虚拟机成功，主机不成功。此时需要关闭主机防火墙。

## 1.6 ubutu 开启tftp服务

1、安装tftp服务

sudo apt-get install tftp tftpd openbsd-inetd

2、chmod 777 nfs/\* -R //nfs目录

3、修改配置

gedit /etc/inetd.conf //root用户

tftp dgram udp wait nobody /usr/sbin/tcpd /usr/sbin/in.tftpd /home/chao/nfs

4、重启服务

/etc/init.d/openbsd-inetd restart

in.tftpd -l /home/chao/nfs

5、测试

echo 'ad' > /home/chao/nfs/hello  
tftp 127.0.0.1  
tftp>get hello

6、将文件系统中/etc/init.d/rcS中的“net\_set &”  
修改为“#net\_set &”这句是设置开发板ip用的，会出问题

## 1.7 通过tftp更新emmc内核

启动的时候，停在uboot计时那里，插上网线

1.按一下q键退出天嵌uboot菜单，进入到uboot命令模式

2.配置好uboot的ipaddr和serverip，并确定E9在uboot下能ping通电脑主机

3.打开tftp32小工具软件，指定好uImage文件目录

4.执行以下命令

mmc dev 3 //选中emmc设备

mmc erase 0x800 0x3000 //擦除内核

tftp 0x10800000 uImage //下载内核到RAM中

mmc write 0x10800000 0x800 0x3000 //将下载到ram中的内核，烧到emmc

## 1.8 如何uboot从哪里开始执行

先编译uboot，然后搜索uboot.lds

uboot.lds连接文件指示启动执行流程

.text :

{

board/freescale/mx6q\_sabresd/flash\_header.module (.text.flasheader)

cpu/arm\_cortexa8/start.o

board/freescale/mx6q\_sabresd/libmx6q\_sabresd.a (.text)

lib\_arm/libarm.a (.text)

net/libnet.a (.text)

drivers/mtd/libmtd.a (.text)

drivers/mmc/libmmc.a (.text)

. = DEFINED(env\_offset) ? env\_offset : .;

common/env\_embedded.o(.text)

\*(.text)

}

## 1.9 uboot如何下载内核

1]boot from emmc

[2]boot from sdcard

[3]boot from tftp

[5]download from sdcard

[0]setting boot args

[q]exit to command mode

Enter your selection: 1

mmc3(part 0) is current device

MMC read: dev # 3, block # 2048, count 12288 ... 12288 blocks read: OK

根据uboot打印出来的菜单，搜索menu命令：

U\_BOOT\_CMD(

menu, 3, 0, do\_menu,

)

do\_menu

menu\_shell()//菜单在这里定义

run\_command("run bootcmd\_mmc\_linux", 0);

cmdtp = find\_cmd(argv[0]) //查找菜单对应的命令

(cmdtp->cmd) (cmdtp, flag, argc, argv)//执行命令

搜索is current device找到对应的函数是do\_mmcops

## 1.10 EMMC分区信息

mmc0: new high speed DDR MMC card at address 0001

mmcblk0: mmc0:0001 008G92 7.28 GiB

mmcblk0boot0: mmc0:0001 008G92 partition 1 4.00 MiB

mmcblk0boot1: mmc0:0001 008G92 partition 2 4.00 MiB

mmcblk0: p1

mmcblk0boot1: unknown partition table

mmcblk0boot0: unknown partition table

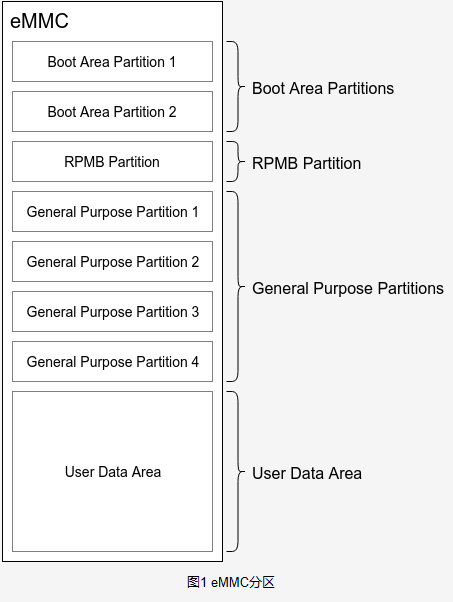
eMMC 标准中，将内部的 Flash Memory 划分为 4 类区域，分别是：

User Data:区域用于存储数据；

boot: 区域用于启动；

replay protected memory block:区域用于存放受保护的数据

general purpose:分区



setenv mxcfb0 video=mxcfb0:dev=lcd,TQ-TFT\_1024600

## 1.11 yocto 配置

免翻墙获取repo工具的方式

通过清华大学镜像源clone repo代码

git clone https://gerrit-google.tuna.tsinghua.edu.cn/git-repo1

本地会同步下一个名为git-repo的工程，工程里包含最新的repo工具，执行repo init就可以直接本地的repo，此时不会再向google的服务器同步最新的repo工具

执行repo init

[git-repo路径]/repo init -u https://android.googlesource.com/platform/manifest

PS：

我还在清华大学镜像网站上看到另外一种方式：

mkdir ~/bin

PATH=~/bin:$PATH

curl https://storage.googleapis.com/git-repo-downloads/repo > ~/bin/repo

chmod a+x ~/bin/repo

update:

上面的方式是将repo添加到环境变量，但是repo运行时，仍然会去google服务器请求最新版本，解决方式是在本地环境变量中指定repo服务器地址

repo的运行过程中会尝试访问官方的git源更新自己，如果想使用tuna的镜像源进行更新，可以将如下内容>复制到你的~/.bashrc里

export REPO\_URL='https://mirrors.tuna.tsinghua.edu.cn/git/git-repo/'

并重启终端模拟器。

# 2 IOMUXC模块

## 2.1 IOMUX 概述

#### IOMUX Hardware Operation：

配置各个引脚的复用功能，可配置成功能引脚，也可配置成普通IO。

中断引脚应该配置成普通IO口，并且设置为输入io引脚。

SW\_MUX\_CTL-Selects the primary or alternate function of a pin. Also enables  
loopback mode when applicable.  
• SW\_SELECT\_INPUT-Controls pin input path. This register is only required when  
multiple pads drive the same internal port.  
• SW\_PAD\_CTL-Control pad slew rate, driver strength, pull-up/down resistance, and  
so on.

#### IOMUX Source Code Structure：

|  |  |
| --- | --- |
| pinctrl-imx.c | i.MX pinctrl core driver |
| pinctrl-imx6q.c | i.MX 6Quad/DualLite pinctrl driver |

IOMUX Programming Interface：  
• imx-pinctrl.txt in Documentation/devicetree/bindings/pinctrl/fsl  
• imx6q-pinctrl.txt in Documentation/devicetree/bindings/pinctrl/fsl  
2.2 IOMUX Controller (IOMUXC)

Every module requires a specific pad setting (such as pull up, keeper, and so on), and for  
each pad, there are up to 8 muxing options (called ALT modes).  
MX6Q\_PAD\_KEY\_COL1\_\_GPIO\_4\_8, //GPIO\_VOL-

(((u64)( 0x0200) << 0 | \ //MUX\_CTRL\_OFS\_SHIFT

((u64)( 5) << 36 | \ //MUX\_MODE\_SHIFT

((u64)( 0x05D0) << 12 \ //MUX\_PAD\_CTRL\_OFS\_SHIFT

((u64)( 0) << 41 | \ // MUX\_PAD\_CTRL\_SHIFT

((u64)( 0x0000) << 24 | \ //MUX\_SEL\_INPUT\_OFS\_SHIFT

((u64)(0) << 59| \ // MUX\_SEL\_INPUT\_SHIFT

(u64)(1<<17) << 41 //MUX\_PAD\_CTRL\_SHIFT

#define MX6Q\_IOMUXC\_BASE\_ADDR (0x02000000 + 0x80000 + 0x60000)=20E0000

\_\_init mx6\_map\_io(void)

mxc\_iomux\_v3\_init(IO\_ADDRESS(MX6Q\_IOMUXC\_BASE\_ADDR));

base = iomux\_v3\_base;// 20E0000

int mxc\_iomux\_v3\_setup\_pad(iomux\_v3\_cfg\_t pad)

{

u32 mux\_ctrl\_ofs = (pad & MUX\_CTRL\_OFS\_MASK) >> MUX\_CTRL\_OFS\_SHIFT;

u32 mux\_mode = (pad & MUX\_MODE\_MASK) >> MUX\_MODE\_SHIFT;

u32 sel\_input\_ofs = (pad & MUX\_SEL\_INPUT\_OFS\_MASK) >> MUX\_SEL\_INPUT\_OFS\_SHIFT;

u32 sel\_input = (pad & MUX\_SEL\_INPUT\_MASK) >> MUX\_SEL\_INPUT\_SHIFT;

u32 pad\_ctrl\_ofs = (pad & MUX\_PAD\_CTRL\_OFS\_MASK) >> MUX\_PAD\_CTRL\_OFS\_SHIFT;

u32 pad\_ctrl = (pad & MUX\_PAD\_CTRL\_MASK) >> MUX\_PAD\_CTRL\_SHIFT;

if (mux\_ctrl\_ofs)

\_\_raw\_writel(mux\_mode, base + mux\_ctrl\_ofs);//5,即ALT5，GPIO[8] of instance: gpio4

if (sel\_input\_ofs)

\_\_raw\_writel(sel\_input, base + sel\_input\_ofs);

if (!(pad\_ctrl & NO\_PAD\_CTRL) && pad\_ctrl\_ofs)// NO\_PAD\_CTRL意思是不需要配置改寄存器

*\_\_raw\_writel(pad\_ctrl, base + pad\_ctrl\_ofs);*

return 0;

}

## GPIO

配置普通IO是输入还是输出，读写IO口电平。

论坛里的关于“客户自动运行自己的程序”不太适合手里的资源，我用的镜像是他们最新发布的，琢磨过程如下、完整启动后，用“ps”查看运行中的进程，如下  
  PID TTY          TIME CMD  
  910 ttySAC0  00:00:00 login  
  911 ttySAC0  00:00:00 sh  
  912 ttySAC0  00:00:00 touchinteractio  
  916 ttySAC0  00:00:00 ps  
看起来默认启动的程序是touchinteractio  
2、cd / 切到根目录下 查找程序出现的地方（grep "touchinteractio" \* -nr）  
etc/embedsky\_conf:34:/test\_file/touchinteraction &  
home/root/.bash\_history:168:cd touchinteraction/  
home/root/.bash\_history:173:./touchinteraction   
home/root/.bash\_history:175:./touchinteraction   
home/root/.bash\_history:177:./touchinteraction   
3、修改etc/embedsky\_conf 34行为自己的应用程序就可以啦  
  
启动过程大同小异，顺藤摸瓜反推一下