# **第四章 烧写系统**

OKMX6ULL-S开发板目前支持OTG和SD卡两种烧写方式。在用户资料/Linux/烧写工具文件夹中提供了OTG和SD卡烧写的烧写工具，烧录工具中包含出厂镜像，用户可选择任意一种方式进行镜像烧写。

当用户仅对部分镜像做修改时可以进行单独更新镜像。

调试内核时可以选择使用网络更新运行镜像，将镜像下载到开发板的DRAM中运行。

## 4.1 烧写所需镜像

* **说明**
* **256M-nand核心板，仅支持console版文件系统，无qt库，不支持qt显示方式。**
* **eMMC核心板，支持qt和console两种文件系统，烧录工具和出厂时均默认qt版。**
* 出厂镜像路径：用户资料\Linux\镜像

OKMX6ULL-S开发板用户资料里提供了出厂烧写的镜像，镜像包括以下几部分：BootLoader镜像、内核镜像、设备树镜像、Logo图片、文件系统、模块文件。

不同核心板烧录的镜像有差异，以下为各种核心板烧录所需的镜像：

|  |  |  |
| --- | --- | --- |
| **镜像** | **eMMC核心板** | **256M Nand核心板** |
| BootLoader | u-boot-imx6ull14x14evk\_emmc.imx | u-boot-imx6ull14x14evk\_nand.imx |
| 内核镜像 | zImage | zImage |
| 设备树镜像 | okmx6ull-s-emmc.dtb | okmx6ull-s-nand.dtb |
| Logo图片 | logo | logo |
| 文件系统 | rootfs-console.tar | rootfs-console.tar |
| rootfs-qt.tar |
| 模块文件  （烧录时解压到文件系统中） | modules.tar.bz2 | modules.tar.bz2 |

## 4.2 SD卡烧写系统

以OKMX6ULL-S+(256M\_256MNAND)为例，烧写NAND进行操作说明，拨码开关设置参考“[烧写及启动设置](#_1.3  烧写及启动设置)”部分。eMMC版本步骤相同，启动方式不同。

### 4.2.1 制作SD烧写卡

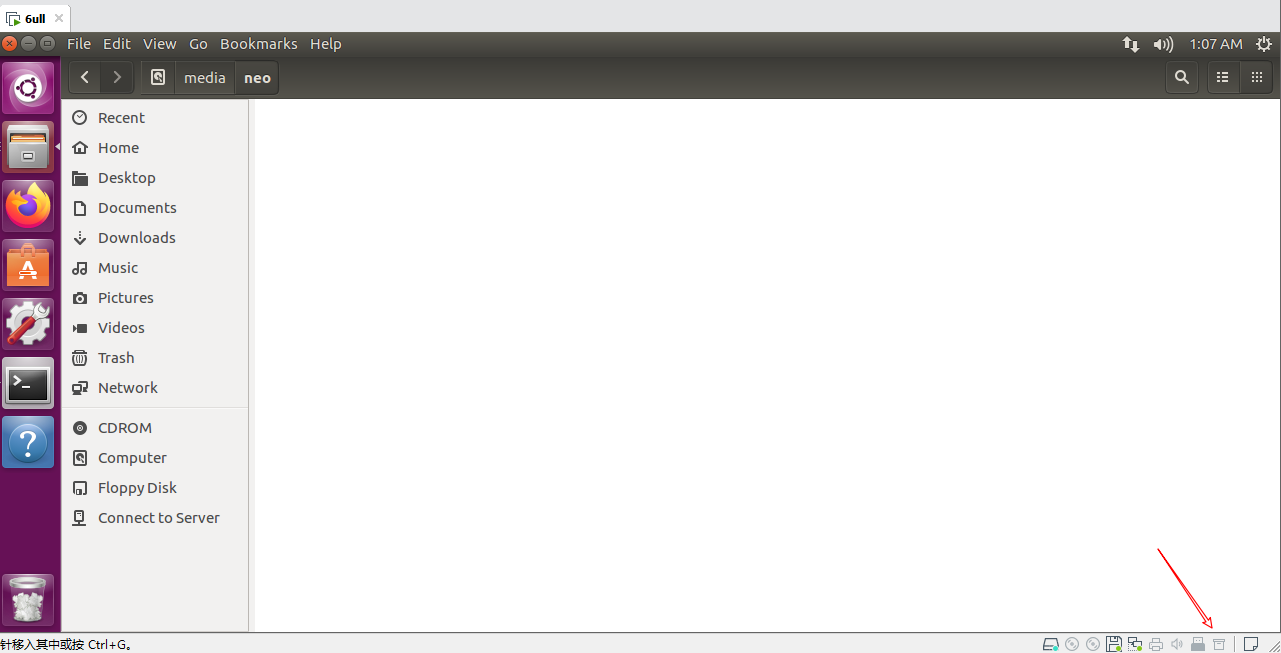
将文件nand-burnsd拷贝到Ubuntu 系统的任一目录，假设为/home/forlinx/work。

* 用户资料\Linux\烧录工具\nand-sdburn.tar.bz2 为256M NAND文件系统。
* 用户资料\Linux\烧录工具\emmc-sdburn.tar.bz2 为4G/8G EMMC文件系统。

1、SD卡使用前使用格式化工具将其格式化为FAT32格式

2、nand-burnsd.zip解压后拷贝到ubuntu系统下的任一目录。例如/home/forlinx/work。

3、使用 USB 读卡器把SD 卡插入到电脑的USB 端口（VMware 虚拟机用户如果U盘没有被虚拟机识别，可以使用如下箭头指向图标将优盘连接到虚拟机）。



4、**待虚拟机识别到SD卡后弹出目录再进行下边烧写操作。**进入/home/forlinx/work/nand-burnsd目录，执行脚本：

|  |
| --- |
| forlinx@ubuntu:~/work/nand-burnsd$ sudo ./burn.sh |

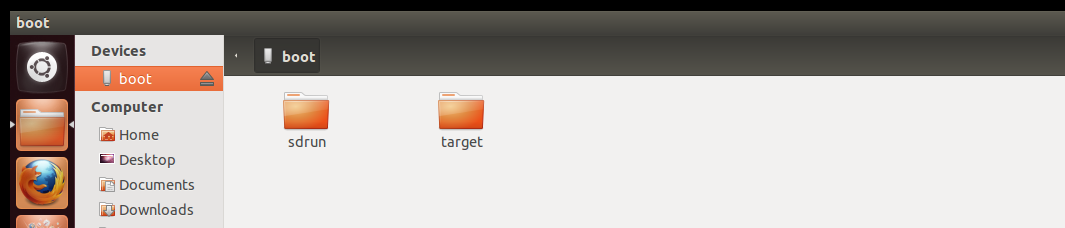
执行上述命令后，终端会列出电脑的硬盘或U盘，对应选择自己的SD 卡，回车。

**注意：**判定自己的U盘是 sda/sdb/sdc 可以根据容量进行判断，比如自己的优盘容量为8G，则其size 为 7761920 KB≈8G，建议用户执行此操作时不要同时插入多个优盘，以免混淆。

这里以我们的操作为例：

|  |
| --- |
| ################################################################################  This script will create a bootable SD card from custom or pre-built binaries.  The script must be run with root permissions and from the bin directory of  the SDK  Example:  $ sudo ./6ullsdburn.sh  Formatting can be skipped if the SD card is already formatted and  partitioned properly.  ################################################################################    Availible Drives to write images to:    # major minor size name  1: 8 16 7761920 sdb    Enter Device Number: 1 //此处选择1    sdb was selected  Checking the device is unmounted  unmounted /dev/sdb1  sdb1 sdb2 sdb3  7757824  ################################################################################  Detected device has 1 partitions already  Re-partitioning will allow the choice of 1 partitions  ################################################################################  Would you like to re-partition the drive anyways [y/n] : y //输入y，回车，等待制卡完成    Now partitioning sdb ...  ################################################################################  Now making 1 partitions  ################################################################################  1+0 records in  1+0 records out  1024 bytes (1.0 kB, 1.0 KiB) copied, 0.0428509 s, 23.9 kB/s  DISK SIZE - 7948206080 bytes  Checking that no-one is using this disk right now ... OK  Disk /dev/sdb: 7.4 GiB, 7948206080 bytes, 15523840 sectors  Units: sectors of 1 \* 512 = 512 bytes  Sector size (logical/physical): 512 bytes / 512 bytes  I/O size (minimum/optimal): 512 bytes / 512 bytes  >>> Created a new DOS disklabel with disk identifier 0x38224bb5.  Created a new partition 1 of type 'W95 FAT32 (LBA)' and of size 500 MiB.  /dev/sdb2:  New situation:  Device Boot Start End Sectors Size Id Type  /dev/sdb1 20480 1044479 1024000 500M c W95 FAT32 (LBA)  The partition table has been altered.  Calling ioctl() to re-read partition table.  Syncing disks.  ################################################################################  Partitioning Boot  ################################################################################  mkfs.fat 3.0.28 (2015-05-16)  mkfs.fat: warning - lowercase labels might not work properly with DOS or Windows    Mount the partitions    Emptying partitions    ################################################################################  Copying files now... will take minutes  ################################################################################  Copying boot partition  copy sdrun/ target/ to SD  Buring the u-boot.imx to sdcard  129+0 records in  129+0 records out  132096 bytes (132 kB, 129 KiB) copied, 0.161529 s, 818 kB/s  431+0 records in  431+0 records out  441344 bytes (441 kB, 431 KiB) copied, 0.422838 s, 1.0 MB/s  Syncing....  Un-mount the partitions  Remove created temp directories  Operation Finished |

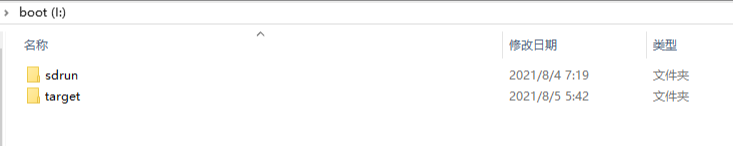
5、卡制作完成后可以看到boot 分区包含sdrun和target两个目录



### 4.2.2 烧写卡介绍

* **说明：**
* 256M Nand文件系统为console版，且只支持consloe版。
* eMMC文件系统默认为qt版，若使用consloe版，可将户资料\Linux\镜像中的rootfs-console.tar.bz2，重命名为rootfs.tar.bz2 ，替换target的同名文件。

制作好的烧写卡包含sdrun和target两个文件夹：



1、sdrun 文件夹内容用于引导系统烧写，一般无需修改；

2、target目录内容会烧写到flash芯片，用户做了镜像修改，需要替换镜像文件时，只需将target目录中的对应文件替换掉，并保持同样的命名，之后再重新进行系统烧写。以下以NAND的SD烧写卡的target内文件做介绍说明：

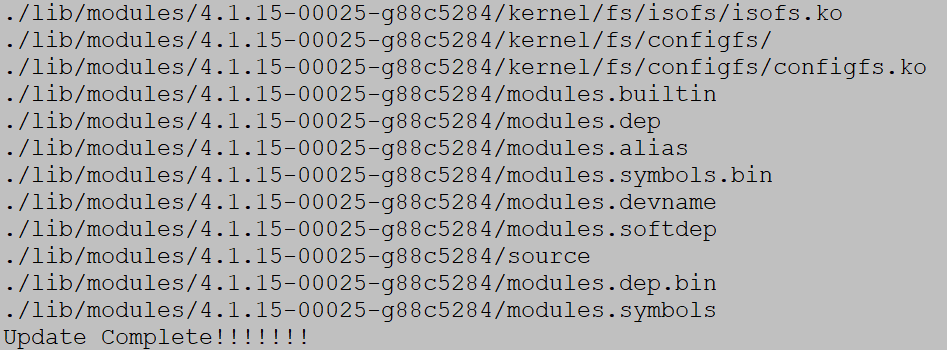
|  |  |
| --- | --- |
| **镜像名称** | **说明** |
| u-boot-imx6ull14x14evk\_nand.imx | BootLoader镜像 |
| zImage | 内核镜像 |
| okmx6ull-s-nand.dtb | 设备树镜像。 |
| logo.bmp | 开机Logo图片。  用户更换开机logo图片，只需重新制作bmp格式图片（制作方法参考：用户资料\应用笔记），命名为logo.bmp替换该文件。 |
| rootfs-console.tar.bz2 | 文件系统，无qt界面及qt库。  用户制作完新的文件系统后，命名为rootfs\_nogpu.tar.bz2并替换该文件，可烧写自己的文件系统。 |
| modules.tar.bz2 | 模块文件（烧录时解压到文件系统中） |

### 4.2.3 SD卡烧写方法

将上一节中制作好的SD卡插入，同时将拨码开关置于下图所示。3、5、8 均为ON，1、2、4、6、7均为 OFF，如下图。此时将SD卡中target的内容烧写到NAND中。



烧写时间较长，等待系统烧写完成之后，串口打印信息：



同时底板上的LED1闪烁。

烧写完成之后，断电，将拨码开关拨到4、5、8均为ON，1、2、3、6、7均为OFF，重新上电，NAND启动。

## 4.3 OTG烧写系统

### 4.3.1 OTG烧录工具简介

* **说明：烧录工具eMMC核心板中默认为qt版的文件系统，若使用console版文件系统，可将mfgtools\Profiles\Linux\OS Firmware \files/linux路径下的rootfs-console.tar.bz2重命名为rootfs-qt.tar.bz2**

OTG烧录使用的是NXP开发的板固件烧录的工具mfgtools，可以烧写uboot,image,dtb,rootfs等镜像。以下对用户在烧录过程中可能用到的文件做简单介绍。

以下路径以：用户资料\Linux\烧写工具\OTG烧写\mfgtools，作为起始路径。

|  |  |  |
| --- | --- | --- |
| **文件名** | **路径** | **说明** |
| mx6ull-4gemmc-512mddr-qt5.6.vbs | mfgtools | 用于烧写eMMC核心板相关镜像 |
| mx6ull-256mnand-256mddr-cmd.vbs | mfgtools | 用于烧写256M Nand核心板相关镜像 |
| ucl2.xml | Mfgtools\Profiles\Linux\OS Firmware | 定义了烧写过程的具体操作步骤和操作内容，用户涉及到单步更新的指令可以查看该文件 |
| 引导相关镜像 | Mfgtools\Profiles\Linux\OS Firmwarefirmware | 文件夹内容用于引导系统烧写，一般无需修改 |
| 烧录到flash中的镜像 | Mfgtools\Profiles\Linux\OS Firmware\files\linux | 文件夹内容用于烧录到flash中，用户修改镜像后，将其重命名为相同的名字并替换，可用于烧写自己的镜像 |

### 4.3.2 OTG烧写方法

* **注意：使用OTG烧写时，不能插入SD卡。**
* 用户资料\Linux\工具\OTG\ mfgtools.rar为OTG烧录工具所在路径

mx6ull-4gemmc-512mddr-qt5.6.vbs 烧写4GeMMC+512MDDR的系统

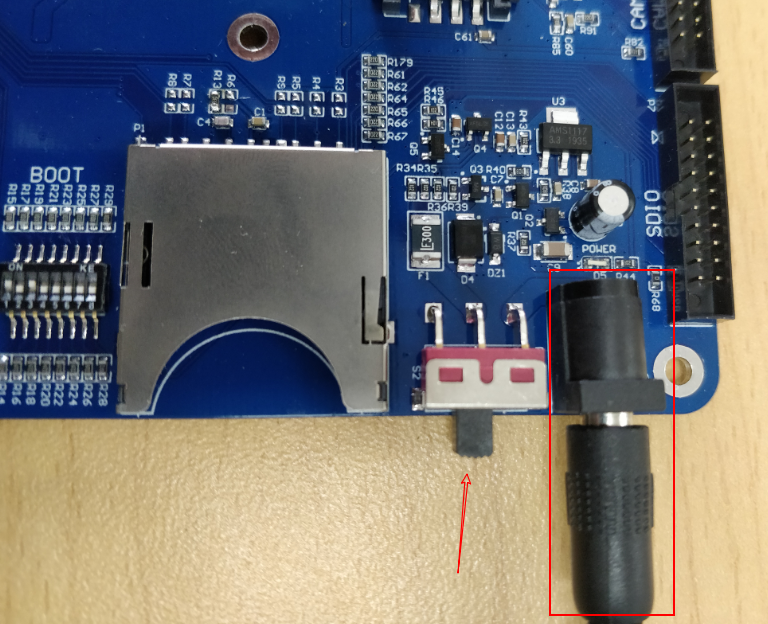
mx6ull-256mnand-256mddr-cmd.vbs 烧写256MNAND+256MDDR系统

**以烧写256M NAND为例进行说明：**

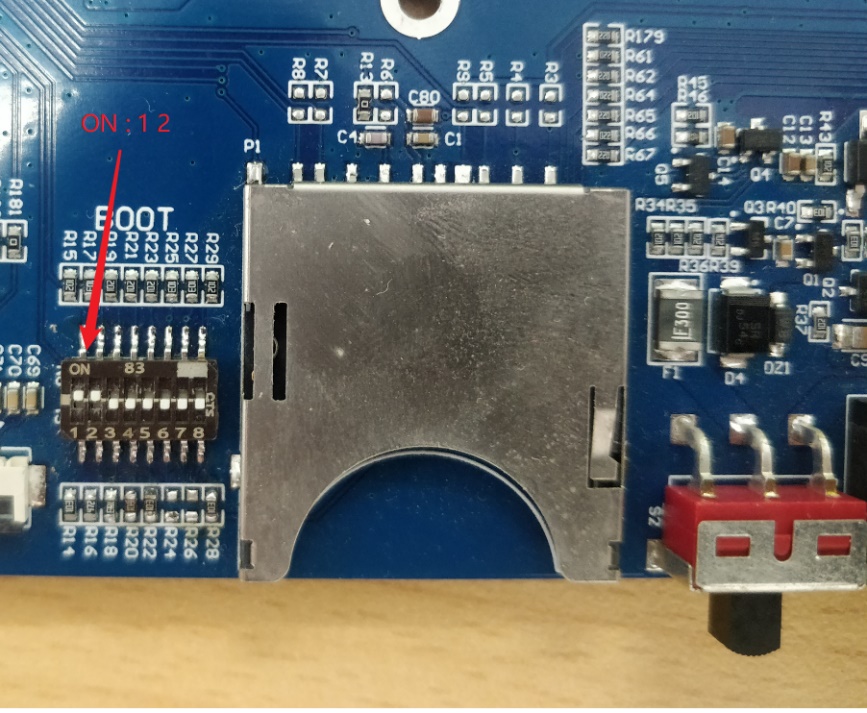
1. 首先将烧写工具Mfg工具拷贝到 windows 下解压，烧写工具路径如下所示：

* 用户资料\Linux\烧录工具\ mfgtools.zip

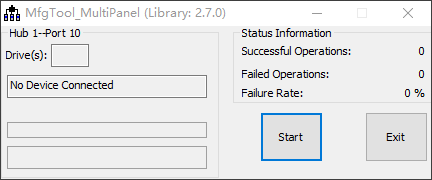
2. 5V电源供电如图，开关拨到供电口端：



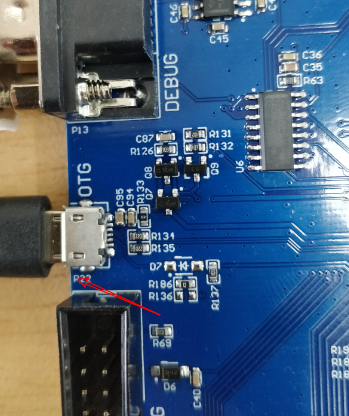
3. 拨码开关设置为 1、2为ON，其他状态任意。如图：



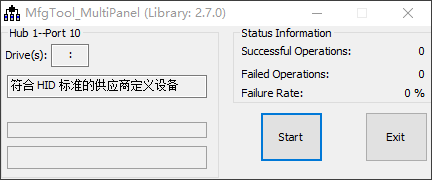
4. 双击“ mx6ull-256mnand-256mddr-cmd.vbs ”（已写好脚本，直接烧写），如下图：



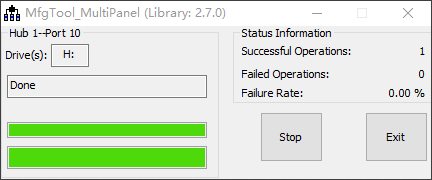
1. 插入USB OTG，如下图：

5.

自动识别为HID，如图：



1. 点击start，开始烧写系统，中间弹出格式化对话框，点击“取消”格式化选项，或者不管它，直到烧写完成。如图：



1. 系统烧写完成之后，出现“Done”，之后点击“stop”，停止。再点击“Exit”，关闭烧写烧写工具。断电，将拨码开关拨到4、5、8均为ON，1、2、3、6、7均为OFF，重新上电，NAND启动。

## 4.4 单步更新内核

### 4.4.1 eMMC单独更新镜像

系统启动后，将待替换设备树放于u盘中，插入到开发板上，在超级终端输入列举U盘文件命令：

|  |
| --- |
| root@fl-imx6ull:~# ls /run/media/sda1/ |

打印信息如下：

|  |
| --- |
| imx6ul-14x14-evk-c-emmc.dtb  logo.bmp  u-boot.imx  zImage  zImagebak |

单独更新uboot

|  |
| --- |
| root@fl-imx6ull:~# echo 0 > /sys/block/mmcblk1boot0/force\_ro //使能写  root@fl-imx6ull:~# dd if=/run/media/sda1/u-boot.imx of=/dev/mmcblk1boot0 bs=512 seek=2  //将uboot镜像写入分区  758+0 records in  758+0 records out  388096 bytes (388 kB, 379 KiB) copied, 0.132975 s, 2.9 MB/s  root@fl-imx6ull:~# echo 1 > /sys/block/mmcblk1boot0/force\_ro //关闭写  root@fl-imx6ull:~# mmc bootpart enable 1 1 /dev/mmcblk1 //使能分区1启动  root@fl-imx6ull:~# sync //保存  root@fl-imx6ull:~# reboot //重启 |

单独更新设备树和内核

|  |
| --- |
| root@fl-imx6ull:~# cp /run/media/sda1/okmx6ull-s-emmc.dtb /run/media/mmcblk1p1/ okmx6ull-s-emmc.dtb  root@fl-imx6ull:~# cp /run/media/sda1/zImage /run/media/mmcblk1p1/zImage  root@fl-imx6ull:~# sync  root@fl-imx6ull:~# reboot |

重启后，到uboot阶段选择对应dtb即可。

### 4.4.2 Nand单步更新内核

单步更新uboot是使用 kobs-ng 工具烧写到 NAND Flash 中去，kobs-ng 是 NXP 官方提供的烧写 uboot 到 NAND Flash 中的工具。

更新dtb以及内核，logo，可以使用nandwrite命令，该命令也已经打包在文件系统中。使用时，要将待烧写的dtb、logo、 zImage文件复制到文件系统的某个目录下（本文示例为/root），然后擦除掉对应分区中的旧固件，再分别使用nandwrite命令将新的固件写到对应的分区中。参考命令如下所示：

系统启动后，将待替换设备树放于u盘中，插入到开发板上，参考命令如下所示：

更新uboot：

|  |
| --- |
| root@fl-imx6ull:~# flash\_erase /dev/mtd0 0 0 //擦除uboot所在分区  Erasing 256 Kibyte @ 7c0000 -- 100 % complete  root@fl-imx6ull:~# kobs-ng init -x /run/media/mmcblk0p1/u-boot.imx //写入uboot镜像  root@fl-imx6ull:~# sync  root@fl-imx6ull:~# reboot |

更新内核：

|  |
| --- |
| root@fl-imx6ull:~# flash\_erase /dev/mtd4 0 0  Erasing 128 Kibyte @ 7e0000 -- 100 % complete  root@fl-imx6ull:~# nandwrite -p /dev/mtd4 /run/media/sda1/target/zImage  Writing data to block 0 at offset 0x0  Writing data to block 1 at offset 0x20000  Writing data to block 2 at offset 0x40000  … |

更新dtb：

|  |
| --- |
| root@fl-imx6ull:~# flash\_erase /dev/mtd3 0 0  Erasing 128 Kibyte @ 2e0000 -- 100 % complete  root@fl-imx6ull:~# nandwrite -s 0x80000 -p /dev/mtd3 /run/media/sda1/target/okmx6ull-s-nand.dtb  Writing data to block 4 at offset 0x80000 |

更新logo:

|  |
| --- |
| root@fl-imx6ull:~# flash\_erase /dev/mtd1 0 0  Erasing 128 Kibyte @ 1e0000 -- 100 % complete  root@fl-imx6ull:~# nandwrite -p /dev/mtd1 /run/media/sda1/target/logo.bmp  Writing data to block 0 at offset 0x0  Writing data to block 1 at offset 0x20000  Writing data to block 2 at offset 0x40000 |