

100ASK_IMX6LL 烧写工具设计与使用说明

版权声明

百问科技©2019

深圳百问网科技有限公司版权所有，并保留对本手册及声明的一切权力。

本文档遵守 GPL 协议。

更新记录

类别	100ask-imx6ull 系列文档
文档名	100ask_imx6ull Development Manual
当前版本	1.0
适用型号	IMX6ULL 开发板
编辑	百问科技文档编辑团队
审核	韦东山

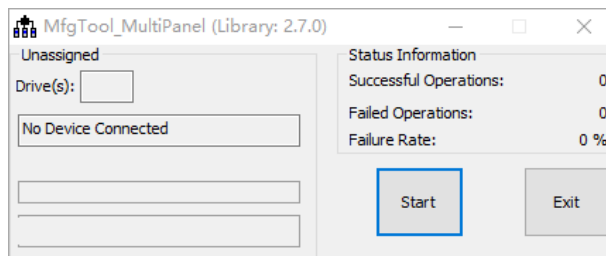
修改日志		
版本	修改时间	更改说明
1.0	2020.03.02	初始版本

目录

100ASK_IMX6LL 烧写工具设计与使用说明	1
版权声明	1
更新记录	1
目录	2
1. 为什么要重新开发烧写工具	3
2. 实现原理	4
3. uuu 使用示例	5
3.1 下载运行裸机程序(uboot 也是裸机)	5
3.2 烧写 led.imx 到 EMMC	6
4. 特制的 Uboot	8
5. EXT4 文件系统制作注意事项	9
5.1 使用 Buildroot 制作映像文件时	9
5.2 使用 mkfs.ext4 制作映像文件时	9
6. 100ASK_IMX6ULL_Flashing_tool 使用说明	10
6.1 基础版：专用于 100ASK_IMX6ULL	10
6.2 专业版：更强大、更灵活、适合所有开发板	11
7. 100ASK_IMX6ULL_Flashing_tool 设计说明	13
7.1 基础版	13
7.1.1 判断设备是否连接：应该全程监测	13
7.1.2 判断设备的固件是否已经在运行	13
7.1.3 烧写整个系统	13
7.1.4 更新内核	13
7.1.5 更新设备树	13
7.1.6 更新 Uboot	13
7.1.7 烧写裸机	14
7.1.8 上传用户文件到根目录	14
7.2 专业版	15
7.2.1 判断设备是否连接：应该全程监测	15
7.2.2 判断设备的固件是否已经在运行	15
7.2.3 运行固件/裸机	15
7.2.4 烧写 boot/裸机	15
7.2.5 烧写整个系统	15
7.2.6 上传任意文件	15
8. GUI 的其他版本	16

1. 为什么要重新开发烧写工具

NXP 公司给 IMX6ULL 提供了烧写工具：mfgtools。它的使用界面如下：



操作很简单，一键烧写整个映像文件。

但是，缺点也很多：

- a. 不能单独烧写 bootloader、内核、设备树

或者说，可以实现这些功能，但是需要你去修改 xml 配置文件，对初学者不友善，对老手也显得麻烦。

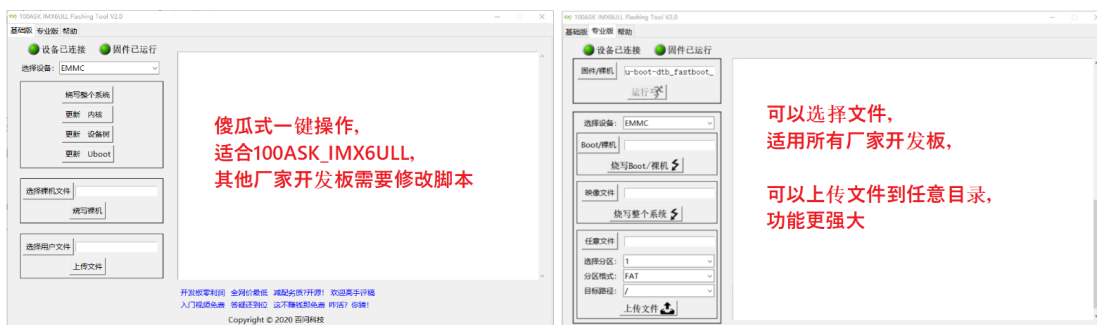
- b. 烧写速度慢

烧写 500M 的系统，耗时 5 分钟(我们的工具可以在 1 分钟内烧完)。

基于上述缺点，我们决定自己开发烧写工具 100ask imx6ull flashing tool，并且完全开源。它有如下特点：

- a. 可以烧写整个系统，也可以分开烧写 bootloader、内核、设备树
- b. 可以上传用户文件到开发板系统中任意目录里
- c. 烧写速度是原厂工具的 5 倍
- d. 支持所有厂家的开发板烧写，注意：不只是支持 100ASK_IMX6ULL

它的界面如下，有基础版、专业版两个页面：

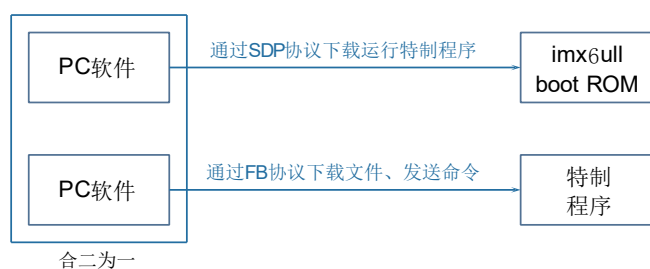


2. 实现原理

IMX6ULL 本身支持 USB 启动，即 PC 通过 USB 线向开发板下载、运行程序。那么我们可以下载一个特殊的程序，这个程序支持：

- 通过 USB 线接收 PC 文件
- 通过 USB 线接收 PC 的命令
- 根据这些命令烧写板子

所以我们需要 2 个软件：PC 软件、特制的程序。如下图所示(imx6ull boot ROM 程序是固化在芯片内部的，设置为 USB 模式时它就会自动运行)：



一开始，PC 软件通过 SDP 协议向 IMX6ULL 芯片下载、运行特制的程序，SDP 协议是 IMX6ULL 厂家自己的协议。

特制的程序支持 Fastboot 协议，这是安卓系统常用的刷机协议。这程序运行起来后，PC 软件就通过 FB 协议与它通信，可以下载文件、发送命令。

特制的程序接收到命令后，就可以去烧写系统。

幸运的是，NXP 公司已经提供了 PC 软件，名为 uuu：Universal Update Utility(又名 mfgtools 3.0)。

支持 Fastboot 协议的 u-boot 也有源码，可惜跟 uuu 不太适配，需要做很多修改。

我们还实现了通过 uuu 向 u-boot 下载文件，通过 u-boot 烧写 FAT、EXT4 分区。可惜 u-boot 并未支持 EXT4 的某些新特性，所以制作 EXT4 文件系统时也需要做些修改。

所以我们要做的就是：

- 修改 u-boot，让它跟 uuu 的配合更顺畅。
- 修改制作 EXT4 文件系统的方法，去掉 u-boot 未支持的特性
- 编写 GUI，让使用更方便

3. uuu 使用示例

要给 uuu 修改配套的 u-boot, 要给它写 GUI, 第一件事当然就是熟悉 uuu 的使用: 先用命令行。

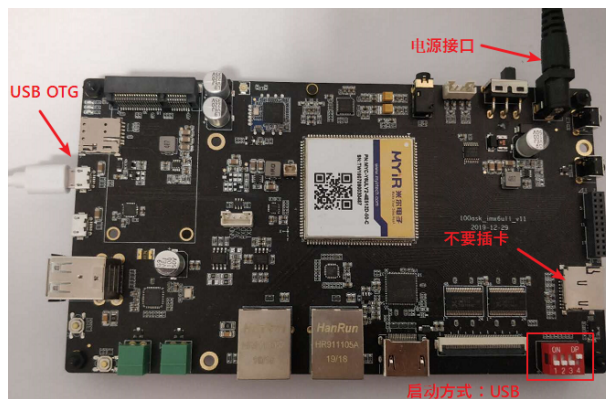
uuu 的 GITHUB 地址为: <https://github.com/NXPmicro/mfgtools>

里面也有编译好的可执行程序: uuu.exe(Windows 用)、uuu(Linux 用), 还有一个 uuu.pdf。

3.1 下载运行裸机程序(uboot也是裸机)

我们在 GITHUB 中已经有现成的 LED 裸机程序和 uboot, 可以直接运行。

首先开发板设置为 USB 启动模式, 不插 SD/TF 卡, 并上电:



然后, 按下图操作:



上图中的命令“.\bin\uuu.exe .\files\led.imx”将会执行内嵌的脚本, 类似如下命令:

```
.\bin\uuu.exe SDP: boot -f ".\files\led.imx"
```

SDP 是协议, uuu 通过 SDP 协议跟板子上的 boot ROM 通信;
boot 是命令, 表示要启动, 后面的“-f”表示要启动哪个文件。

你还可以把它写入一个脚本文件, 比如 led.clst (后缀含义 clst: command list):

```
uuu_version 1.2.39
SDP: boot -f ".\files\led.imx"
SDP: done
```

clst 文件中第 1 行必须写 uuu_version, 表明它适用的最低版本号。

然后在命令行中执行如下命令, 也可以达到同样的效果:

```
.\bin\uuu.exe led.clst
```

3.2 烧写led.imx到EMMC

uuu 本身没有烧写功能，所以需要借助特制的程序：支持 Fastboot 协议的 uboot。

在 GITHUB 中我们也提供了该 uboot。

开发板设置为 USB 模式启动，并打开命令行后，可以如下操作把 led.imx 烧写到 EMMC：

```
.\bin\uuu.exe -b emmc .\firmware\u-boot-dtb_fastboot_100ask.imx .\files\led.imx
```

命令解析：

- a. “-b emmc”：burn emmc，烧写 EMMC
- b. 需要借助特制的程序：.\firmware\u-boot-dtb_fastboot_100ask.imx
- c. 烧写谁？.\files\led.imx

这个命令其实会使用内嵌的脚本来烧写，可以执行“.\bin\uuu.exe -bshow emmc”查看脚本，结果如下：

```
uuu_version 1.2.39

# @_flash.bin          | bootloader
# @_image    [_flash.bin] | image burn to emmc, default is the same as bootloader

# This command will be run when i.MX6/7 i.MX8MM, i.MX8MQ
SDP: boot -f _flash.bin

# This command will be run when ROM support stream mode
# i.MX8QXP, i.MX8QM
SDPS: boot -f _flash.bin

# These commands will be run when use SPL and will be skipped if no spl
# SDPU will be deprecated. please use SDPV instead of SDPU
# {
SDPU: delay 1000
SDPU: write -f _flash.bin -offset 0x57c00
SDPU: jump
# }

# These commands will be run when use SPL and will be skipped if no spl
# if (SPL support SDPV)
# {
SDPV: delay 1000
SDPV: write -f _flash.bin -skipspl
SDPV: jump
# }

FB: ucmd setenv fastboot_dev mmc
FB: ucmd setenv mmcdev ${emmc_dev}
FB: ucmd mmc dev ${emmc_dev}
```

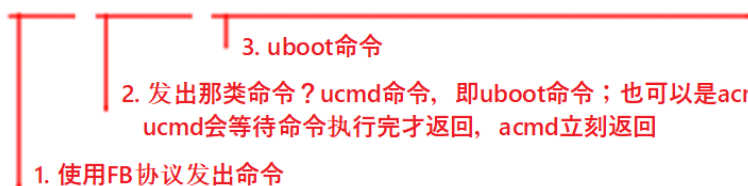
```
FB: flash bootloader_image
FB: ucmd if env exists emmc_ack; then ; else setenv emmc_ack 0; fi;
FB: ucmd mmc partconf ${emmc_dev} ${emmc_ack} 1 0
FB: Done
```

这个脚本支持 IMX6、IMX7、IMX8MM、IMX8Q，我们只关心 IMX6ULL 的话，脚本可以精简为 burn_led.clst，内容如下：

```
uuu_version 1.2.39
SDP: boot -f .\firmware\u-boot-dtb_fastboot_100ask.imx
FB: ucmd setenv fastboot_dev mmc
FB: ucmd setenv mmcdev ${emmc_dev}
FB: ucmd mmc dev ${emmc_dev}
FB: flash bootloader .\files\led.imx
FB: ucmd if env exists emmc_ack; then ; else setenv emmc_ack 0; fi;
FB: ucmd mmc partconf ${emmc_dev} ${emmc_ack} 1 0
FB: Done
```

可以看到除了使用 SDP 协议启动特制的 Uboot 之外，就是使用 FB 协议跟这个 Uboot 通信了。这些 FB 命令举个例子来讲解：

FB: ucmd setenv fastboot_dev mmc

- 
- 3. uboot命令
 - 2. 发出那类命令？ucmd命令，即uboot命令；也可以是acmd
ucmd会等待命令执行完才返回，acmd立刻返回
 - 1. 使用FB协议发出命令

从这些脚本可知， uuu 更多的时候是通过 FB 协议向 Uboot 下载文件、发送命令；核心都是 Uboot，苦活累活是 Uboot 做的。

4. 特制的 Uboot

修改说明有时间再补，主要参考 uuu.pdf，还有根据 uuu 内嵌的脚本添加环境变量。

在 GITHU 中已经有改好的源码，编译方法如下(你的工具链的 PATH 可能跟我们的不同，请自行修改)：

```
export ARCH=arm
export CROSS_COMPILE=arm-linux-gnueabi-
export PATH=$PATH:/home/book/100ask_imx6ull-sdk/ToolChain/gcc-linaro-6.2.1-2016.11-x86_64_arm-linux-gnueabi/bin

cd u-boot-imx_fastboot
cp config_ok .config
make
cp u-boot-dtb.imx u-boot-dtb_fastboot_100ask.imx
```

所得到的 u-boot-dtb_fastboot_100ask.imx 文件就是特制的 Uboot，我们的烧写工具中已经在 firmware 目录里放置了该文件。

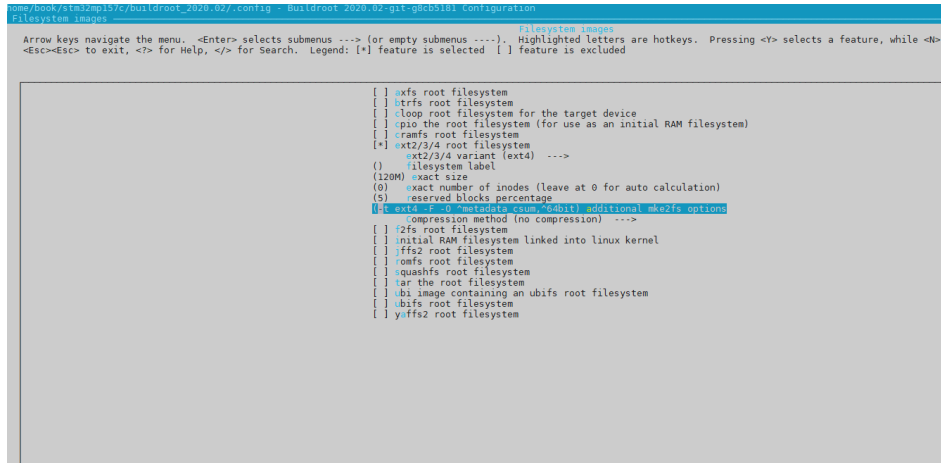
5. EXT4 文件系统制作注意事项

我们的 GUI 功能强大，可以把 PC 上的任意文件发给 Uboot，再借助它的 ext4write 命令烧写到板子上的任意目录中。

但是 Uboot 对 EXT4 的支持并没有跟得上 Linux，比如它不支持新特性：metadata_csum。所以在 PC 上制作 EXT4 映像文件时，要去除这个特性。

5.1 使用Buildroot制作映像文件时

在 Buildroot 源码目录下执行 make menuconfig，如下图添加选项即可：



5.2 使用mkfs.ext4制作映像文件时

我们有时候使用 mkfs.ext4 来制作 EXT4 映像，有些 Linux 系统的 mkfs.ext4 版本比较低，它来本就不支持 metadata_csum 特性，所以不需要加上特别的选项。经测试：1.43 及以下版本的 mkfs.ext4 不需要做特别设置。

对于高于 1.43 的 mkfs.ext4，制作映像文件时要加上参数“-O ^metadata_csum”，它是禁止 metadata_csum 的意思，比如：

```
dd if=/dev/zero of=rootfs.ext4 bs=1024 count=409600
mkfs.ext4 -O ^metadata_csum rootfs.ext4
sudo mount -t ext4 rootfs.ext4 /mnt
sudo tar xjf rootfs.tar.bz2 -C /mnt
sudo umount tmp
```

6. 100ASK_IMX6ULL_Flashing_tool 使用说明

100ASK_IMX6ULL_Flashing_tool 是 uuu 工具的 GUI 前端，操作更便利。从 GITHUB 下载工具后，在“100ask_imx6ull 烧写工具”目录下双击运行“100ask_imx6ull_flashing_tool.exe”。

它有“基础版”、“专业版”两个页面。

“基础版”是专为 100ASK_IMX6ULL 设计的，点击一下即可完成某项烧写。

“专业版”功能更强大，特别是它可以上传文件到某个分区、某个目录。有些厂家的开发板，zImage 和设备树是在第 1 个分区里的，而 100ASK_IMX6ULL 的 zImage 和设备树是在第 2 个分区里，所以这些厂家的开发板就无法使用基础版来烧写，需要使用专业版，指定分区、指定分区格式、指定路径，然后再上传文件。

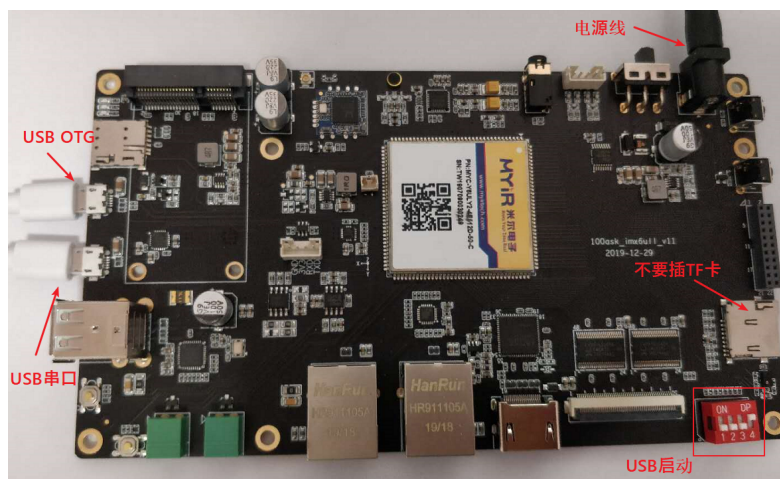
注意：开发板必须设置为 USB 模式，如果要用 SD/TF 卡，必须先上电再插卡；
不能先插卡再上电，不能先插卡再上电，不能先插卡再上电！

6.1 基础版：专用于100ASK_IMX6ULL

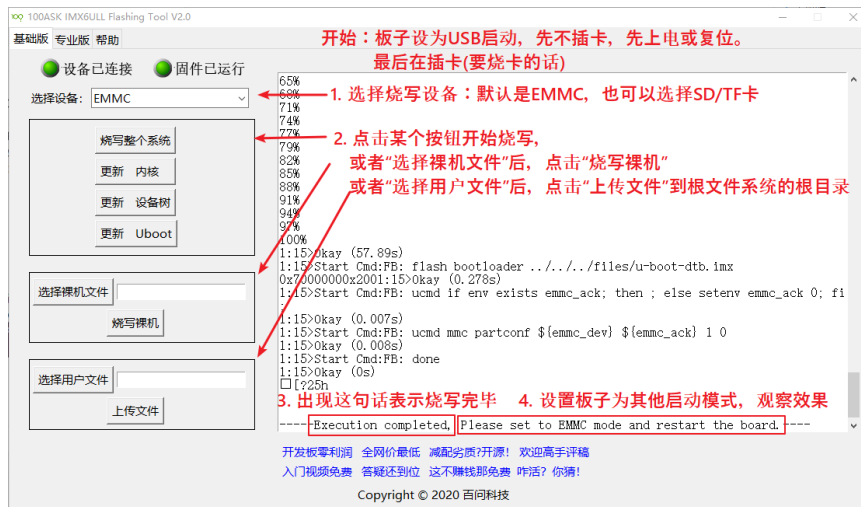
接线、设置 USB 启动的方式如下：

注意：USB 串口线可接可不接，接上只是为了观察烧写过程。

注意：设置为 USB 启动时，先不要接 TF 卡



板子复位或重新上电后，在 APP 里操作即可，一个图就可以列清楚所有步骤：

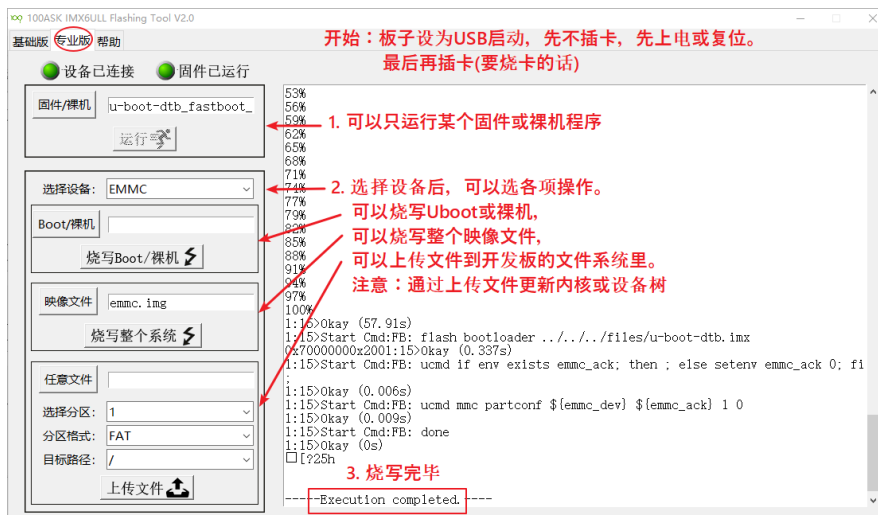


各按钮作用如下：

- 烧写整个系统：把 emmc.img 烧到 EMMC 上，或是把 sdcard.img 烧到 SD/TF 卡上；
- 更新内核：把 zImage 上传到根文件系统的/boot 目录
- 更新设备树：把 100ask_imx6ull-14x14.dtb 上传到根文件系统的/boot 目录
- 更新 Uboot：把 u-boot-dtb.imx 烧写到 EMMC 或 SD/TF 卡；
- 烧写裸机：把所选裸机文件，烧写到 EMMC 或 SD/TF 卡；
- 上传文件：把所选用户文件，上传到根文件系统的/目录

6.2 专业版：更强大、更灵活、适合所有开发板

专业版的强大在于烧写文件时可以选择任意文件，上传文件时可以指定分区、分区格式、目标路径。用法也很简单，一图足以说明：



有些开发板厂家把内核 zImage、设备树放在第 1 个分区里，它通常是 FAT 分区。那么可以使用专业版来更新内核、更新设备树。

比如：

任意文件	zImage
选择分区:	1
分区格式:	FAT
目标路径:	/
<input type="button" value="上传文件"/>	

- 1. 选择文件
- 2. 选择分区
- 3. 指定分区格式
- 4. 指定目标路径，FAT分区只能选择根目录
- 5. 点击上传

几乎所有的开发板的第 2 个分区都是 EXT4 格式，我们可以上传文件到它的任意目录下，比如：

任意文件	100ask_test.wav
选择分区:	2
分区格式:	EXT4
目标路径:	/root
<input type="button" value="上传文件"/>	

- 1. 选择文件
- 2. 选择分区
- 3. 指定分区格式
- 4. 指定目标路径，EXT4分区可以指定任意目录
- 5. 点击上传

7. 100ASK_IMX6ULL_Flashing_tool 设计说明

7.1 基础版

7.1.1 判断设备是否连接：应该全程监测

执行命令：

```
./bin/uuu -lsusb
```

结果中有 "SDP" 或 "FB"

7.1.2 判断设备的固件是否已经在运行

执行命令：

```
./bin/uuu -lsusb
```

结果中有 "FB"

7.1.3 烧写整个系统

如果设备的固件未运行：帮用户运行固件，然后再烧写。

如果设备的固件已经运行：则可以直接烧写。

烧写方法：

执行脚本

```
./bin/uuu scripts/basic/<emmc|sd|...>/write_all.clst
```

7.1.4 更新内核

如果设备的固件未运行：帮用户运行固件，然后再烧写。

如果设备的固件已经运行：则可以直接烧写。

烧写方法：

执行脚本

```
./bin/uuu scripts/basic/<emmc|sd|...>/write_kernel.clst
```

7.1.5 更新设备树

如果设备的固件未运行：帮用户运行固件，然后再烧写。

如果设备的固件已经运行：则可以直接烧写。

烧写方法：

执行脚本

```
./bin/uuu scripts/basic/<emmc|sd|...>/write_dtb.clst
```

7.1.6 更新 Uboot

如果设备的固件未运行：帮用户运行固件，然后再烧写。

如果设备的固件已经运行：则可以直接烧写。

烧写方法：

执行脚本

```
./bin/uuu scripts/basic/<emmc|sd|...>/write_boot.clst
```

7.1.7 烧写裸机

如果设备的固件未运行: 帮用户运行固件, 然后再烧写。

如果设备的固件已经运行: 则可以直接烧写。

烧写方法:

- a. 根据用户选择, 修改脚本, 把要烧写的文件名替换进脚本里:

```
scripts/basic/<emmc|sd|...>/write_noos.clst
```

- b. 执行脚本

```
./bin/uuu scripts/basic/<emmc|sd|...>/write_noos.clst
```

7.1.8 上传用户文件到根目录

如果设备的固件未运行: 帮用户运行固件, 然后再烧写。

如果设备的固件已经运行: 则可以直接烧写。

上传方法:

- a. 先下载文件:

```
./bin/uuu FB: download -f <file>
```

- b. 然后设置环境变量:

```
./bin/uuu FB: ucmd setenv TARGET_FILE <路径> // 根据用户选择设置目标路径
```

- c. 最后执行脚本

```
./bin/uuu scripts/pro/<emmc|sd|...>/write_user_file.clst
```

7.2 专业版

7.2.1 判断设备是否连接：应该全程监测

执行命令：

```
./bin/uuu -lsusb
```

结果中有 "SDP" 或 "FB"

7.2.2 判断设备的固件是否已经在运行

执行命令：

```
./bin/uuu -lsusb
```

结果中有 "FB"

7.2.3 运行固件/裸机

执行命令：

```
./bin/uuu -t 1 <file>
```

file: 默认是 firmware/u-boot-dtb_fastboot_100ask.imx; 用户可选择。

7.2.4 烧写 boot/裸机

如果设备的固件未运行: 帮用户运行固件, 然后再烧写。

如果设备的固件已经运行: 则可以直接烧写。

烧写方法:

a. 根据用户选择修改脚本

```
scripts/pro/<emmc|sd|...>/write_boot.clst
```

b. 执行脚本:

```
./bin/uuu scripts/pro/<emmc|sd|...>/write_boot.clst
```

7.2.5 烧写整个系统

如果设备的固件未运行: 帮用户运行固件, 然后再烧写。

如果设备的固件已经运行: 则可以直接烧写。

烧写方法:

a. 根据用户选择修改脚本

```
scripts/pro/<emmc|sd|...>/write_all.clst
```

b. 执行脚本:

```
./bin/uuu scripts/pro/<emmc|sd|...>/write_all.clst
```

7.2.6 上传任意文件

如果设备的固件未运行: 帮用户运行固件, 然后再烧写。

如果设备的固件已经运行: 则可以直接烧写。

上传方法:

a. 先下载:

```
./bin/uuu FB: download -f <file>
```

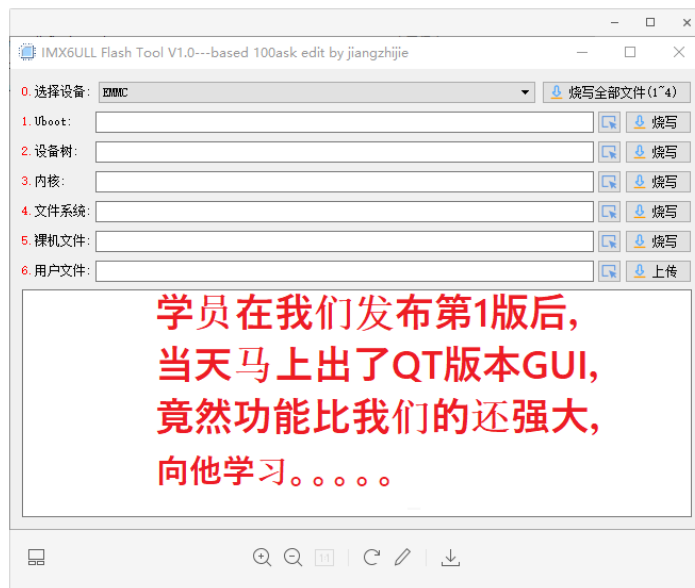
b. 然后设置环境变量:

```
./bin/uuu FB: ucmd setenv SEL_DEV <EMMC | SD | NAND>
```

```
./bin/uuu FB: ucmd setenv PART <1|2|...> // 根据用户选择设置分区
./bin/uuu FB: ucmd setenv FSTYPE <FAT/EXT4> // 根据用户选择设置分区格式
c. 最后执行脚本
./bin/uuu scripts/pro/<emmc|sd|...>/write_user_file.clst
```

8. GUI 的其他版本

我们的学员很厉害，我们刚发布第 1 版时，功能很简单，不能选择文件。是学员做出了 QT 版本的 GUI，可以选择文件。



受此启发，我们才做出了专业版。

这位学员说，他正在 Ubuntu 下调试 GUI，搞定后也会公开源码。

敬请期待。