# 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](#_Toc34063709)

[版权声明 1](#_Toc34063710)

[更新记录 1](#_Toc34063711)

[目录 2](#_Toc34063713)

[1. 为什么要重新开发烧写工具 3](#_Toc34063714)

[2. 实现原理 4](#_Toc34063715)

[3. uuu使用示例 5](#_Toc34063716)

[3.1 下载运行裸机程序(uboot也是裸机) 5](#_Toc34063717)

[3.2 烧写led.imx到EMMC 6](#_Toc34063718)

[4. 特制的Uboot 8](#_Toc34063719)

[5. EXT4文件系统制作注意事项 9](#_Toc34063720)

[5.1 使用Buildroot制作映像文件时 9](#_Toc34063721)

[5.2 使用mkfs.ext4制作映像文件时 9](#_Toc34063722)

[6. 100ASK\_IMX6ULL\_Flashing\_tool使用说明 10](#_Toc34063723)

[6.1 基础版：专用于100ASK\_IMX6ULL 10](#_Toc34063724)

[6.2 专业版：更强大、更灵活、适合所有开发板 11](#_Toc34063725)

[7. 100ASK\_IMX6ULL\_Flashing\_tool设计说明 13](#_Toc34063726)

[7.1 基础版 13](#_Toc34063727)

[7.1.1 判断设备是否连接：应该全程监测 13](#_Toc34063728)

[7.1.2 判断设备的固件是否已经在运行 13](#_Toc34063729)

[7.1.3 烧写整个系统 13](#_Toc34063730)

[7.1.4 更新内核 13](#_Toc34063731)

[7.1.5 更新设备树 13](#_Toc34063732)

[7.1.6 更新Uboot 13](#_Toc34063733)

[7.1.7 烧写裸机 14](#_Toc34063734)

[7.1.8 上传用户文件到根目录 14](#_Toc34063735)

[7.2 专业版 15](#_Toc34063736)

[7.2.1 判断设备是否连接：应该全程监测 15](#_Toc34063737)

[7.2.2 判断设备的固件是否已经在运行 15](#_Toc34063738)

[7.2.3 运行固件/裸机 15](#_Toc34063739)

[7.2.4 烧写boot/裸机 15](#_Toc34063740)

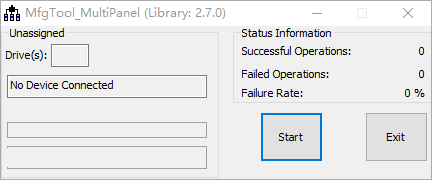
[7.2.5 烧写整个系统 15](#_Toc34063741)

[7.2.6 上传任意文件 15](#_Toc34063742)

[8. GUI的其他版本 16](#_Toc34063743)

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，一键烧写整个映象文件。

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：

a. 通过USB线接收PC文件

b. 通过USB线接收PC的命令

c. 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：

a. 修改u-boot，让它跟uuu的配合更顺畅。

b. 修改制作EXT4文件系统的方法，去掉u-boot未支持的特性

c. 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。

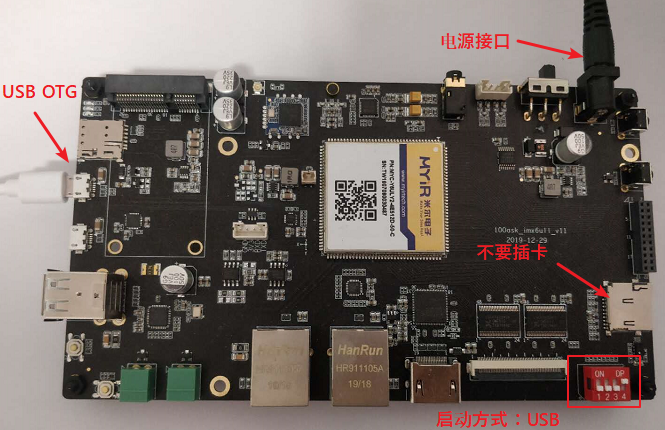
uuu的GITHUB地址为：<https://github.com/NXPmicro/mfgtools>

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。

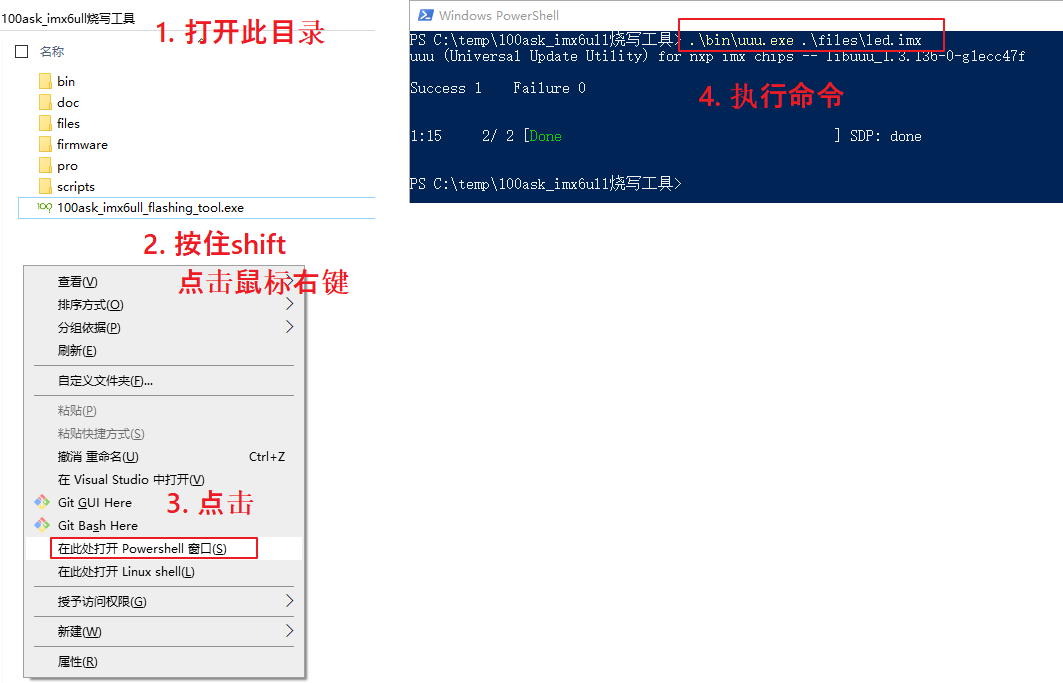
3.1 下载运行裸机程序(uboot也是裸机)

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}

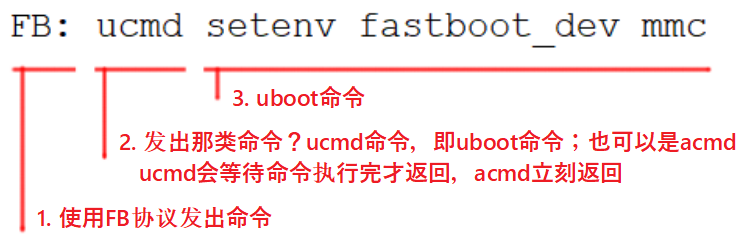
FB: flash bootloader .\files\led.imx

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：



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-gnueabihf-

export PATH=$PATH:/home/book/100ask\_imx6ull-sdk/ToolChain/gcc-linaro-6.2.1-2016.11-x86\_64\_arm-linux-gnueabihf/bin

cd uboot-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。

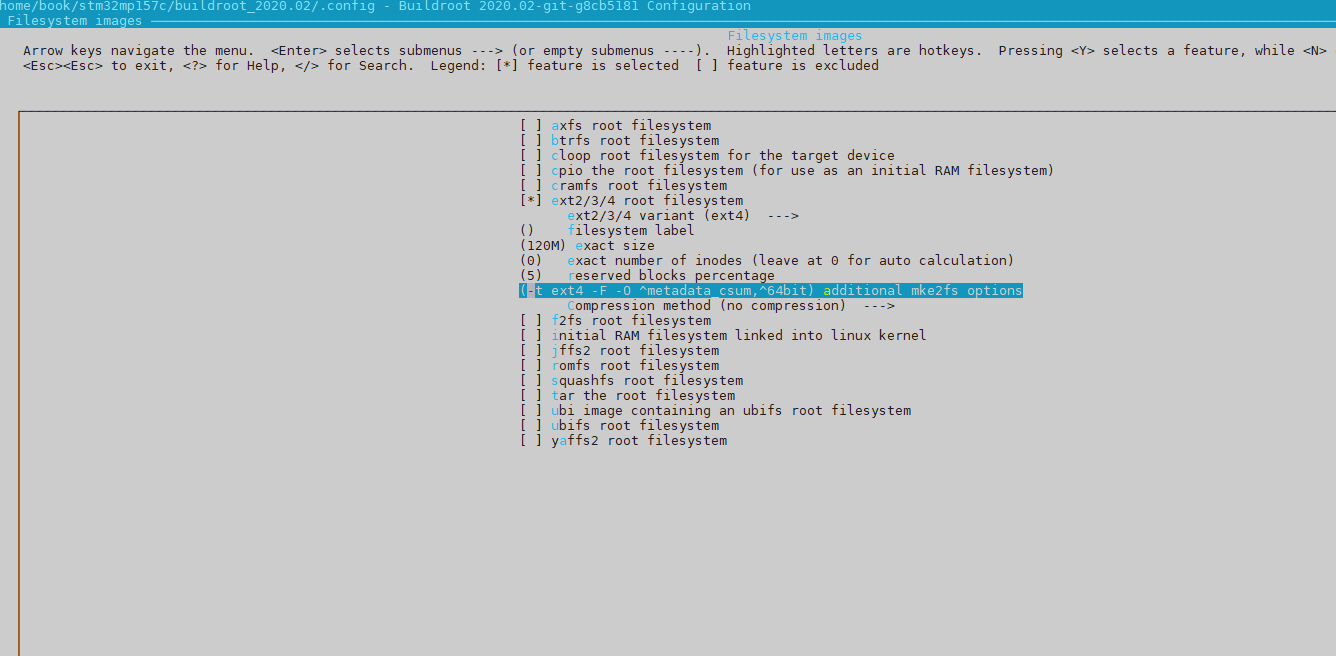
5. EXT4文件系统制作注意事项

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；

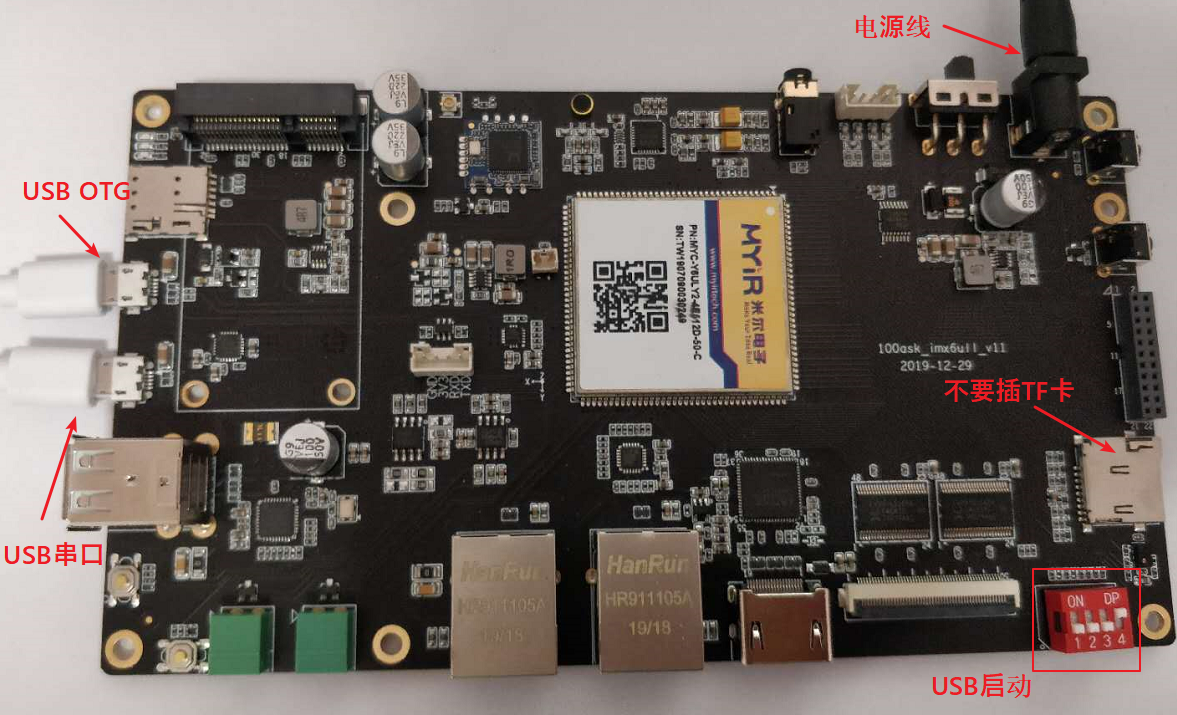
不能先插卡再上电，不能先插卡再上电，不能先插卡再上电！

6.1 基础版：专用于100ASK\_IMX6ULL

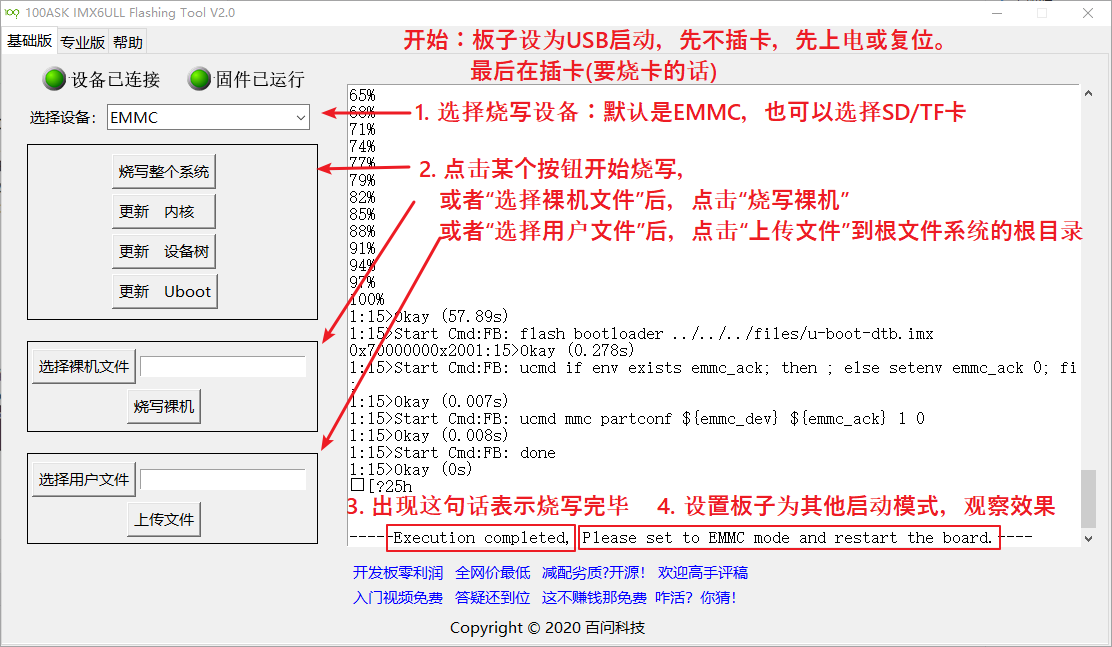
接线、设置USB启动的方式如下：

**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：

a. 烧写整个系统：把emmc.img烧到EMMC上，或是把sdcard.img烧到SD/TF卡上；

b. 更新内核：把zImage上传到根文件系统的/boot目录

c. 更新设备树：把100ask\_imx6ull-14x14.dtb上传到根文件系统的/boot目录

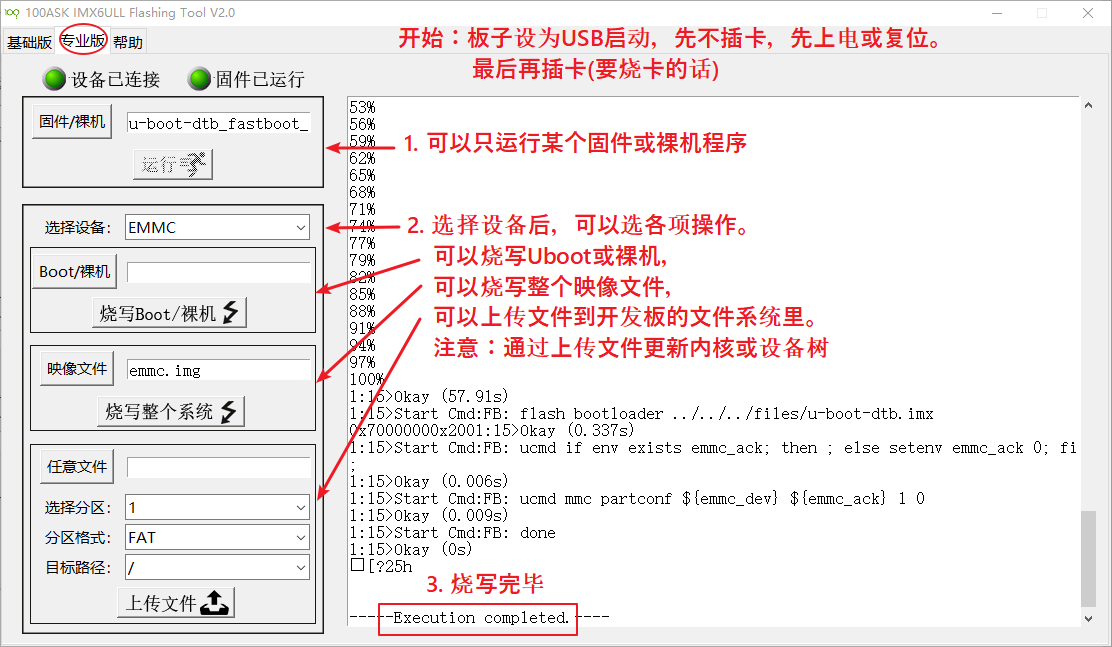
d. 更新Uboot：把u-boot-dtb.imx烧写到EMMC或SD/TF卡；

e. 烧写裸机：把所选裸机文件，烧写到EMMC或SD/TF卡；

f. 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：



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：



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

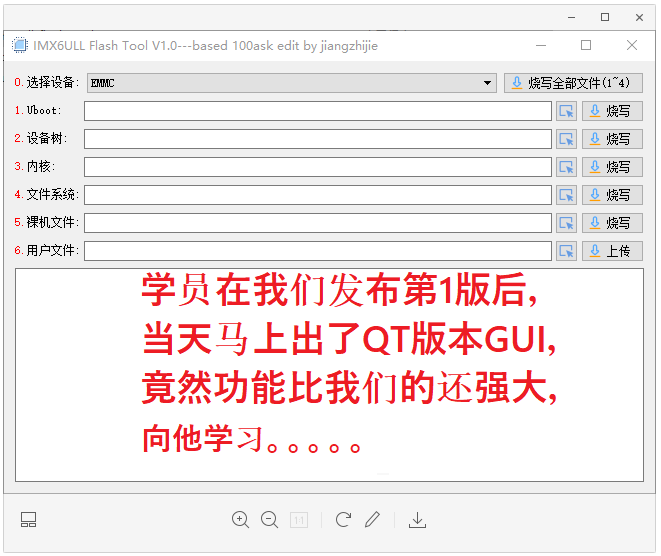
./bin/uuu FB: ucmd setenv FSTYPE <FAT/EXT4> // 根据用户选择设置分区格式

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。