

# Secure Boot 应用指导

翱捷科技（上海）有限公司

修订 1.2

2020/12/07

---

---



---

# 目录

前言 .....	v
1. 介绍 .....	1
1.1. Trusted模式的技术背景 .....	1
1.2. Trusted模式的镜像文件 .....	1
1.3. 固件启动流程 .....	1
1.4. Secboot使能步骤 .....	3
2. 证书链的制作 .....	5
2.1. 环境准备 .....	5
2.2. 操作步骤 .....	5
3. 制作fuse only下载包 .....	9
3.1. config配置文件 .....	9
3.2. 操作步骤 .....	10
3.2.1. 命令行方式 .....	10
3.2.2. GUI界面的方式 .....	12
3.3. 查看fuse only zip包内容 .....	13
4. 烧写fuse .....	17
4.1. GUI界面方式 .....	17
4.2. cmd命令行方式 .....	18
5. Fuse结构 .....	27
6. 重启模块 .....	29
7. 镜像签名 .....	35
7.1. 如何自动镜像签名 .....	36
7.1.1. UI界面方式 .....	36
7.1.2. 命令行方式 .....	37
7.2. 如何分步实现的客户定制的镜像签名 .....	43
7.2.1. 客户替换证书链文件保存公钥 .....	43
7.2.2. 客户生成公钥Hash .....	45
7.2.3. 从help信息中找到第一步和第二步的命令行。 .....	45
7.2.4. 新建一个本地目录保存Hash文件 .....	46
7.2.5. 执行第一步命令行 .....	47
7.2.6. 客户实现对Hash数字签名 .....	49
7.2.7. 执行第二步命令行 .....	49
7.3. 如何使能Boot33的安全检验功能 .....	52
7.4. 如何选配Boot33验签的image .....	53

---

7.4.1. 单flash分区表中增加新image .....	53
7.4.2. 双flash分区表中增加新image .....	55
7.4.3. fwcerts中增加新image .....	57
7.4.4. boot33源码中增加新image .....	59
7.5. 打包下载包之前更新boot33 .....	61
8. 固件下载 .....	63
8.1. 下载 .....	63
8.1.1. UI界面方式 .....	63
8.1.2. cmd命令行方式 .....	65
8.2. 重启 .....	73
8.2.1. 不使能boot33验签功能，启动的log .....	73
8.2.2. 使能boot33验签功能，启动时boot33的log .....	79
9. 工厂双工位烧写 .....	83
9.1. 为两个工位准备一套版本 .....	83
9.2. 工位1烧写fuse only下载包A .....	83
9.3. 工位2烧写镜像签名的下载包B .....	83

---

# 前言

翱捷科技（上海）有限公司（简称“ASR”）作为一家高科技公司，具备完整、强大的移动智能终端芯片和物联网芯片的研发和产业化能力。公司为移动通讯、物联网和智能终端市场提供更优秀的产品方案和高效的技术支持。

翱捷科技（上海）有限公司提供该文档内容用以支持其客户的产品设计。客户须按照文档中提供的规范、参数来设计其产品。由于客户操作不当而造成的人身伤害或财产损失，本公司不承担任何责任。在未声明前，ASR有权对该文档进行更新。

版权申明：

本文档版权属于翱捷科技（上海）有限公司，任何人未经我司允许而复制转载该文档将承担法律责任。

版权所有 © 2020 翱捷科技（上海）有限公司，保留一切权利。

Copyright © 2020 ASR Microelectronics ( Shanghai ) Co., Ltd.



---

Trusted模式启动用于产品交付给OEM后的阶段，Non-trusted模式仅用于早期的ASR内部的产品开发阶段。启动模式是Trusted模式还是Non-trusted模式由Fuse设置。

### 1.1. Trusted模式的技术背景

Trusted模式提供给用户符合工业标准的数字签名认证技术，运用了SHA256的哈希算法，RSA或者ECDSA加密算法。

### 1.2. Trusted模式的镜像文件

需要加密的固件二进制文件（bin）经过ASR的打包工具arelease.exe，自动添加一个FIP包头，合成新的镜像文件（image）。镜像文件的FIP包头中包含了OEM公钥(OEM's RSA public decryption key) 和数字签名(signature)。

### 1.3. 固件启动流程

如下图所示，分三个阶段介绍固件的数字签名和签名认证过程。

#### a. 固件的签名过程:

客户通过ASR的打包工具arelease.exe在制作下载包的过程中自动对固件的二进制文件(例如: preboot.bin)采用SHA256算法求出哈希值(Hash)，再自动用私钥签名固件的这个Hash，得到固件的数字签名文件。

#### a. 镜像文件的内部结构: 客户通过打包工具arelease.exe把固件的二进制文件转化成镜像文件，为镜像文件添加了FIP的头，FIP中保存了OEM的公钥和数字签名。

b. 镜像文件的签名认证过程:

- OEM公钥的认证过程: 镜像文件 ( image ) 在固件烧写和板子启动的阶段都要进行签名认证。比如: 下载包preboot.img, 从image的FIP头中读出OEM的公钥, 程序对公钥计算出Hash, 并且与fuse中block2区域保存的OEM公钥Hash进行比对。如果两个Hash相同, 说明公钥验证通过。如果不相同, 说明镜像文件中的公钥验证不通过, 退出下载或者启动过程。
- 镜像文件的签名认证过程: OEM公钥验证通过以后, 下载或者启动程序会继续对镜像文件 ( image ) 中的固件二进制数据计算出Hash, 再用FIP头中的OEM公钥对FIP头中的数字签名做解签得到另一个Hash, 如果两个Hash值一致, 说明签名认证通过。下载或者启动程序继续下一个流程。如果不相同, 说明签名认证不通过, 退出下载或者启动过程。

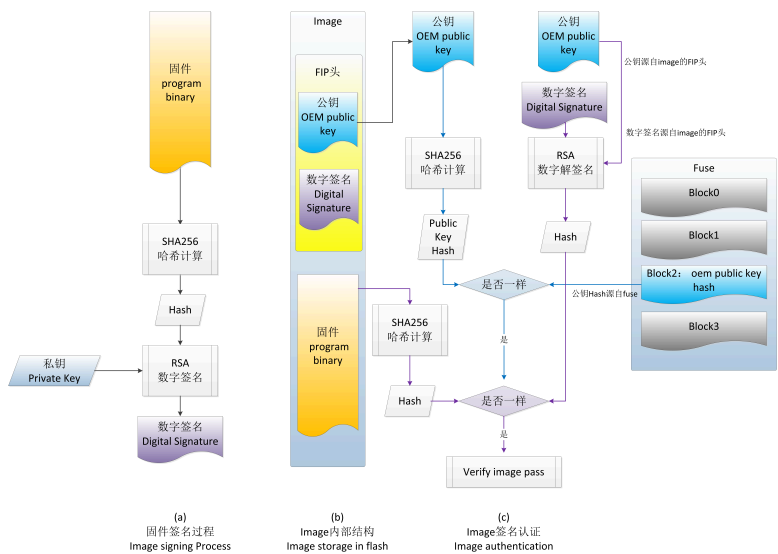


图 1.1. 固件的数字签名以及启动阶段的签名认证的过程图解

固件的签名过程a) 和镜像文件打包过程b) 可分为自动签名模式和分步的客户定制模式两种。

- 自动签名模式：

UI界面或者命令行方式，在打包的过程中自动完成。签名用的私钥保存在工具目录config/security/key，release打包过程直接从本地文件读到私钥，对固件的hash做数字签名。固件的二进制文件转化成镜像文件，为镜像文件添加了FIP的头，FIP中保存了OEM的公钥和数字签名。

- 分步的客户定制模式：

1. 客户提供公钥和私钥。客户的公钥文件覆盖工具的“config/security/key”目录中证书链文件，私钥文件保存在客户的服务器中。
2. 通过命令行方式执行第一阶段的命令，工具自动对固件的二进制文件(例如:preboot.bin)采用SHA256算法求出256个字节的哈希值(Hash)，以Hash的后32个字节作为文件名，内容为Hash值，生成文件，保存到本地目录。
3. 客户读取哈希文件，用远程的私钥对Hash做RSA签名，生成数字签名，生成文件，文件名仍然是Hash的后32个字节，扩展名是.hash.enc,保存在本地目录。
4. 再通过命令行方式执行第二阶段的命令，工具自动计算出固件的hash，以得到数字签名的文件名，读取数字签名文件，得到数字签名。固件的二进制文件转化成镜像文件，为镜像文件添加了FIP的头，FIP中保存了OEM的公钥和数字签名。

## 1.4. Secboot使能步骤

1. 生成私钥和证书
2. 制作烧写OEM公钥的fuse only下载包
3. 烧写fuse only下载包

AES的Root Key和OEM公钥Hash被烧录进Fuse。

4. 重启模块或者下电模块

模块重启以后，Bootrom进入Trusted 模式启用。新的证书开始生效，启动中完成c) 的签名认证过程。

5. 镜像签名

Arelease工具对固件文件完成a) 数字签名过程，生成b) 镜像文件,制作出固件下载包。

6. 固件烧录

烧写镜像文件的过程中必须通过c)签名认证，才能继续烧写过程。

## 7. 模块重启

Trusted Boot必须通过c)签名认证，才能继续启动。



# 2

## 证书链的制作

Arelease.exe是为ASR为客户提供的一个自动打包工具。按下面的操作方法，可以实现证书更新。

### 2.1. 环境准备

1. Windows PC, 建议使用64位机。
2. 获取ASR提供的打包工具。
  - a. 64位windows PC，请下载压缩包：

```
aboot-tools-<tagName>-win-x64.exe。
```

- b. 32位windows PC，请下载压缩包：

```
aboot-tools-<tagName>-win-x86.exe。
```



<tagName>是工具的版本号，以发布此版本的时间定义，举例: aboot-tools-2020.03.25-win-x64.exe。以下命令行的举例，仅以一个tag为例子，以win64位为例。请客户使用最新工具进行操作，按实际的<tagName>替换命令。

### 2.2. 操作步骤

- 1) 解压工具包 ( Win64为例 )

```
双击工具的aboot-tools-<tag name>-win-x64.exe
```

```
自动解压为本地目录about-tools-<tag name>-win-x64
```

```
cd about-tools-<tag name>-win-x64
```

2) 执行Windows的命令窗口cmd.exe。 在窗口 ( console ) 界面输入命令如下 , 获得命令帮助信息 :

```
arelease.exe -h
```

### cmd console.

```
U:\about-tools-2020.10.01-win-x64>arelease.exe -h
Asrmicro About Release Application.

Generate release package for designated product.
Usage: arelease.exe [OPTION]... [FILE]
  -h, --help                Display this help and exit
  -c, --config=config       Products base config directory
                           Omit for current directory
  -I, --image-base=base     Images base directory
  -k, --keygen              Generate security keys
                           --key-alg=alg    Key algorithm: 'rsa' (default), 'ecdsa'
  -l, --list                List all supported products
  -g, --generate            Generate release package
                           --erase-all      Add erase all command before any flash
operations
  --erase-all-only        Only execute erase all command
  --fuse-only              Only generate fuse commands
  --upload-only            Only generate upload commands
  -p, --product=product     Select which product to generate
  -v, --variant=variant     Select product variant
  -i, --images=images       Set image path for image id
                           E.X. -i seagull=seagull.bin,msa=msa.bin
  [FILE]                   The option FILE designated the output filename
                           if not designated, will use $product.zip as
default

Example:
arelease.exe -c . -k --key-alg=rsa❶
arelease.exe -c . -l
arelease.exe -c . -g -p ASR_CRANE_EVB -v CRANE_A0_16MB
arelease.exe -c . -g --erase-all -p ASR_CRANE_EVB -v CRANE_A0_16MB
```

```
arelease.exe -c . -g --erase-all-only -p ASR_CRANE_EVB -v
CRANE_A0_16MB
arelease.exe -c . -g --fuse-only -p ASR_CRANE_EVB -v CRANE_A0_16MB
arelease.exe -c . -g --upload-only -p ASR_CRANE_EVB -v CRANE_A0_16MB
arelease.exe -c . -g -p ASR_CRANE_EVB -v CRANE_A0_16MB
ASR_CRANE_EVB.zip
arelease.exe -c config -I images -g -p ASR_CRANE_EVB -v CRANE_A0_16MB
```

- ❶ 从help帮助信息中可以看到此命令，参数“-k”指 用于制作新的证书链。“-key-alg”用于选择数字签名的算法，可选的有rsa和ecdsa两种算法，举例使用rsa。

3) 输入命令制作新key：

```
arelease.exe -c . -k --key-alg=rsa
```

#### cmd console.

```
G:\aboot-tools-2020.10.01-win-x64>arelease.exe -c . -k --key-alg=rsa
Parsing command line paramters ...
Finished parsing command line paramters.
NOTICE: Creating new key for 'Root Of Trust key'
NOTICE: Creating new key for 'Trusted world key'
NOTICE: Creating new key for 'Non Trusted world key'
NOTICE: Creating new key for 'SCP Firmware Content Certificate key'
NOTICE: Creating new key for 'SoC Firmware Content Certificate key'
NOTICE: Creating new key for 'Trusted OS Firmware Content Certificate
key'
NOTICE: Creating new key for 'Non Trusted Firmware Content Certificate
key'
NOTICE: Creating new key for 'Non Trusted OS Content Certificate key'
}
```

如上信息所示，新的key已经生成。可以切换至key对应的目录下检查新key：

```
cd config\security\key
```

```
dir
```

#### cmd console.

```
G:\aboot-tools-2020.10.01-win-x64\config\security\key>dir
```

```

2020/03/26 00:32      241 ecdsa_non_trusted_world_key.pem
2020/03/26 00:32      241 ecdsa_nt_fw_key.pem
2020/03/26 00:32      241 ecdsa_nt_os_key.pem
2020/03/26 00:32      32 ecdsa_rootpk_sha256.bin
2020/03/26 00:32      241 ecdsa_root_key.pem
2020/03/26 00:32      241 ecdsa_scp_fw_key.pem
2020/03/26 00:32      241 ecdsa_soc_fw_key.pem
2020/03/26 00:32      241 ecdsa_tos_fw_key.pem
2020/03/26 00:32      241 ecdsa_trusted_world_key.pem
2020/03/26 11:00    1,704 rsa_non_trusted_world_key.pem
2020/03/26 11:00    1,704 rsa_nt_fw_key.pem
2020/03/26 11:00    1,708 rsa_nt_os_key.pem
2020/03/26 11:00    1,704 rsa_root_key.pem❶
2020/03/26 11:00      32 rsa_rootpk_sha256.bin❷
2020/03/26 11:00    1,704 rsa_scp_fw_key.pem
2020/03/26 11:00    1,704 rsa_soc_fw_key.pem
2020/03/26 11:00    1,704 rsa_tos_fw_key.pem
2020/03/26 11:00    1,704 rsa_trusted_world_key.pem

```

❶ 新生成的OEM的rsa root key。

❷ 新生成的OEM公钥的Hash ( hash of Rsa public key ) 。

如上信息所示，命令参数选择的是rsa证书，所以rsa相关的证书文件都被自动更新了。

# 3

## 制作fuse only下载包

证书更新以后，需要把OEM公钥的Hash值写入fuse的Block2区域。制作fuse only包，需要用到arelease.exe命令行工具，以命令行方式执行操作，制作用于专门下载fuse的下载包。



about.exe有界面的工具中的releasePage功能无法制作fuse only下载包。必须用命令行工具arelease.exe实现。

### 3.1. config配置文件

```
about-tools-2020.10.01-win-x64\config\product\ASR_CRANE_EVB.json
```

此配置文件中与Secure Boot相关的三行参数配置如下，客户需要配置secureBoot为true。

```
"keyAlg": "rsa", ❶  
"hashAlg": "sha256", ❷  
"secureBoot": true, ❸
```

- ❶ keyAlg设定为rsa算法，客户如果要改选ECDSA算法，修改这项参数。
- ❷ hashAlg设定为sha256算法。
- ❸ secureBoot默认设定为false。如果设置true，打包程序会输出fuse.dat。如果设置false，打包程序不输出fuse.dat。要烧写fuse only，客户需要手动修改它为true。

## 3.2. 操作步骤

有 两种方式可以制作 fuseonly的包 1. Arelease.exe (命令行方式) 2. Aboot.exe (GUI方式)

### 3.2.1. 命令行方式

1 ) 在windows的cmd.exe的窗口输入如下命令，查询可以被支持的产品类型：

```
arelease.exe -c . -l
```

#### cmd console.

```
U:\about-tools-2020.10.01-win-x64>arelease.exe -c . -l
Parsing command line paramters ...
Finished parsing command line paramters.
Supported product list:
  ORDER      PRODUCT          VARIANT
  0.          ASR_CRANE_EVB
  0.0                CRANEGM_A0_16MB
  0.1                CRANEGM_A0_32MB
  0.2                CRANEG_Z2_16+8MB
  0.3                CRANEG_Z2_32+8MB
  0.4                CRANEG_Z2_32MB
  0.5                CRANE_A0_08MB
  0.6                CRANE_A0_16+8MB❶
  0.7                CRANE_A0_16MB❷
  0.8                CRANE_A0_32MB
  1.          ASR_JACANA_EVB
  1.0                JACANA_EVB
```

❶ 芯片是CRANE\_A0，有外置16MB flash和内置8MB flash。

❷ 芯片是CRANE\_A0，仅外置16MB flash。

客户需要针对板子的硬件配置，选对产品类型。以下举例皆选用CRANE\_A0\_16MB的板子来解释操作过程。

2 ) 输入命令，制作Fuse only的下载包：

```
arelease.exe -c . -g --fuse-only -p ASR_CRANE_EVB -v CRANE_A0_16MB
```

-v 是指选定的产品类型。产品类型是通过-l参数先查询出来。

-g --fuse-only是指制作出仅用于烧写fuse的下载包。

## cmd console.

```
G:\aboot-tools-2020.10.01-win-x64>arelease.exe -c . -g --fuse-only -p
ASR_CRANE_EVB -v CRANE_A0_16MB
Parsing command line paramters ...
Release package file name not specified, use product name as base.
Finished parsing command line paramters.
Processing all images needed ...
  Opening image "__fuse_app_img__" from "G:\aboot-tools-2020.10.01-win-
x64\images\2019.01.15\fuse.img" ... ❶
  Done.
  Generating fuse image "fuse" with id "fuse.dat" ... ❷
  Done.
Done.
Calculating total progress weight ...
Done.
Generating download commands ...
Done.
Generating target release package ...
Done.
Release package generated successfully!
```

- ❶ 使用fuse.img，注册fuse下载命令。
- ❷ 输出fuse的数据文件fuse.dat，它被打包入下载包。



“aboot-tools-2020.03.25-win-x64\images\2019.01.15”目录下有启动阶段硬件配置依赖的固件包。“2019.01.15”是CRANE的Bootrom版本号，fuse.img生成在“2019.01.15”目录下。“2019.10.10”目录是CRANEG的Bootrom版本号。如果产品选的是CRANEG的类型，fuse.img会生成在“2019.10.10”目录下。

在当前目录下新生成下载包文件

ASR\_CRANE\_EVB\_CRANE\_A0\_16MB\_FUSE.zip。



以上举例都是以单flash的CRANE芯片为例，如果是双flash的CRANE芯片3601s，产品类型是CRANE\_A0\_16+8MB，则新生成下载包文件ASR\_CRANE\_EVB\_CRANE\_A0\_16+8MB.zip。

**cmd console.**

```
G:\about-tools-2020.10.01-win-x64>dir
2020/10/12  00:32          4,287,087 about.exe
2020/10/12  00:32          102,910 AbootPG.pdf
2020/10/12  00:32       2,694,970 AbootUG.pdf
2020/10/12  00:32       1,788,928 adownload.exe
2020/10/12  00:32       2,725,888 arelease.exe
2020/10/12  13:10          6,905 ASR_CRANE_EVB_CRANE_A0_16MB_FUSE.zip❶
```

**❶** 是生成的Fuse only的下载包。

**3.2.2. GUI界面的方式**

在windows操作系统中，双击Aboot.exe打开GUI界面，进入Release page。按照下图的选择制作的release的包为fuse only功能的包。

- 选中一个 产品类型。
- 选择package type是fuseonly。

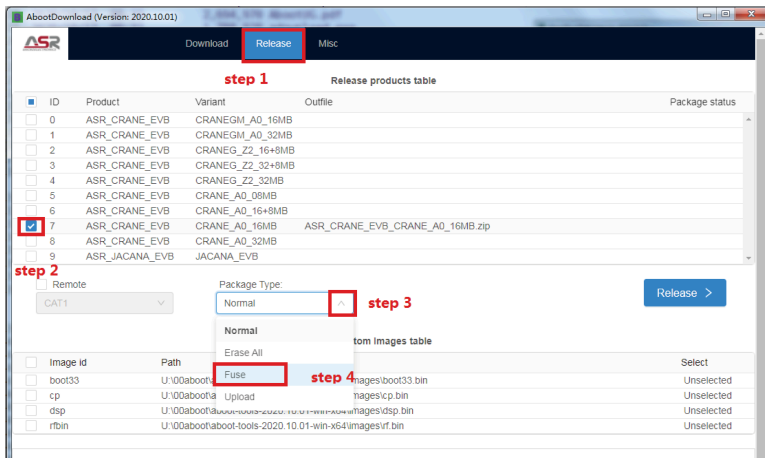


图 3.1. aboot.exe选择制作Fuse only的下载包

- 点击Release按钮，开始做zip包。



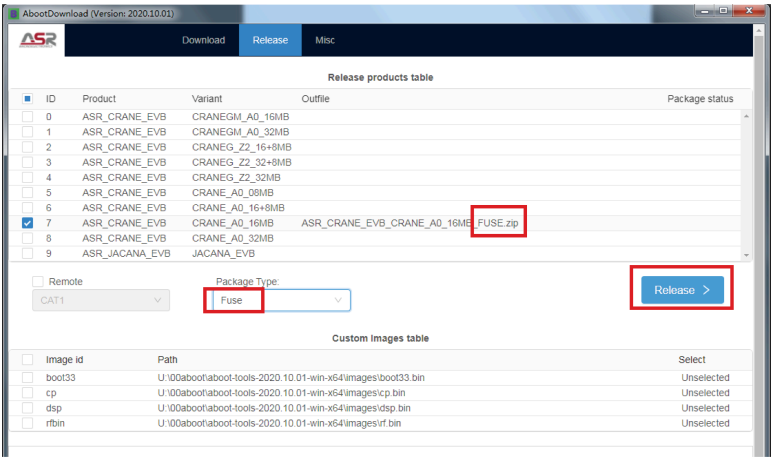


图 3.2. aboot.exe制作Fuse only的下载包

- 做zip包结束。 Done表示做包成功， Fail表示做包失败。

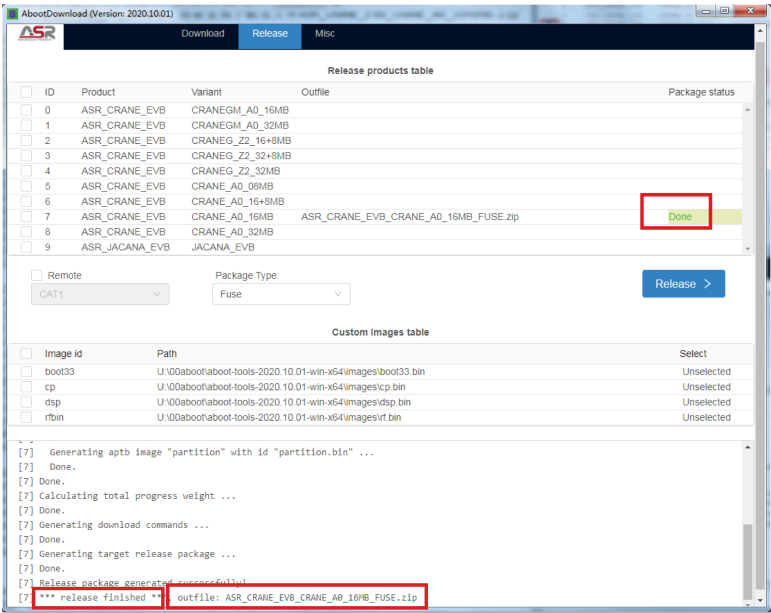


图 3.3. aboot.exe制作Fuse only的下载包成功

### 3.3. 查看fuse only zip包内容

可以用当前目录下的aboot.exe，以GUI界面方式，把fuse包中的文件和命令信息显示出来。

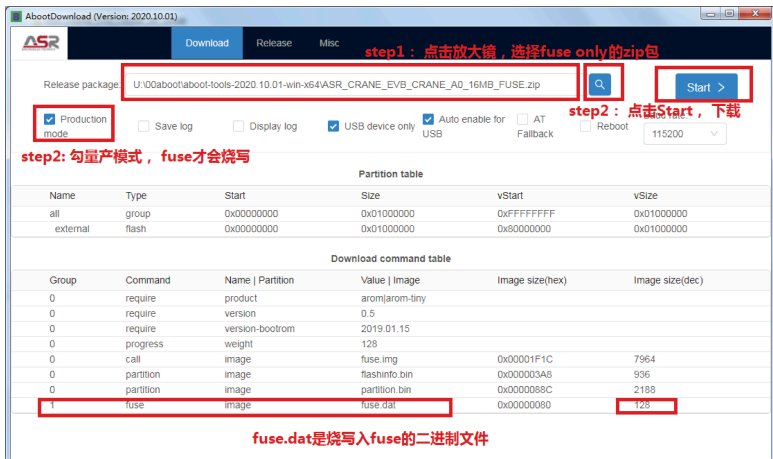


图 3.4. about.exe解析Fuse only的下载包的信息



GUI模式下Abboot.exe烧写fuse包，要勾选上**production mode**，才会烧写fuse。开发模式不允许烧写fuse，见 Download Command table的command: fuse 没有被选中。



命令行模式下Adownload.exe烧写fuse包，要加参数-m，以量产模式烧写，才会烧写fuse。

如果对下载包里的数据解压，可以看到它由下列文件构成。

**cmd console.**

```
G:\about-tools-2020.03.25-win-x64\ASR_CRANE_EVB_CRANE_A0_16MB_FUSE>dir
2020/10/12  14:54                652 download.json
2020/10/12  14:54                 5 fota.json
2020/10/12  14:54                128 fuse.dat❶
2020/10/12  00:32            7,964 fuse.img❷
2020/10/12  14:52            2,184 partition.bin
```

- ❶ fuse.dat是二进制文件。在下载的过程中，fuse命令下发时，它作为128个字节的数据源写入fuse区域。
- ❷ fuse.img用于加载fuse命令的程序。



fuse除了block2的AES Root Key的字节是fuse命令运行时候算出的随机数，其他的字节来源于fuse.dat。

如下图所示，fuse.dat中只有0x40h~0x59h有数据，也就是对应fuse的Block2，是OEM公钥的Hash值，其他都是0x0。具体比特位的含义见Fused章节的描述。

fuse.dat																
	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00000000h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000010h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000020h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000030h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000040h:	65	7C	CA	56	FC	85	96	D5	B9	58	27	A5	D1	98	E4	4F
00000050h:	D7	0D	EA	CE	8D	E9	7A	E5	D9	27	56	2E	DA	7A	0D	CA
00000060h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000070h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

图 3.5. fuse.dat的二进制内容

如果用户发现包里没有找到fuse.dat，配置文件可能被人工修改了。请检查文件

```
about-tools-2020.10.01-win-x64\config\product\ASR_CRANE_EVB.json
```

```
"keyAlg": "rsa",
"hashAlg": "sha256",
"secureBoot": true, ❶
```

- ❶ 如果secureBoot是false，就不会生成fuse.dat，改成true，就会生成fuse.dat。



# 4

## 烧写fuse

烧写fuse下载包分为两种方式: UI界面方式和cmd命令行方式。在下载包的烧写过程中会把OEM公钥的Hash烧入fuse的block2。并且置位block0中的sbe比特为1, 标志启动方式为Trusted模式。板子重启以后进入Trusted模式。

### 4.1. GUI界面方式

双击Aboot.exe

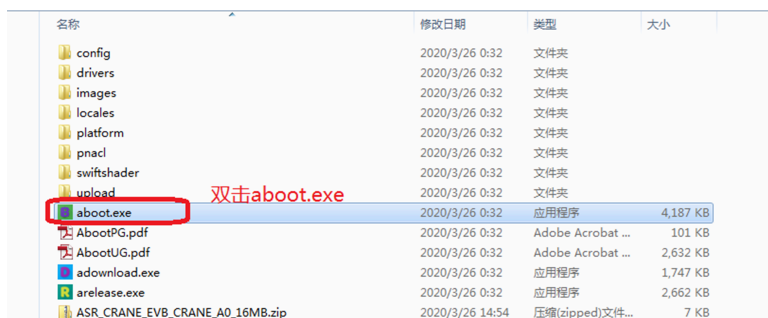


图 4.1. 工具aboot.exe

1. 点击放大镜选择上一章节制作的fuse only的下载包：

ASR\_CRANE\_EVB\_CRANE\_A0\_16MB\_FUSE.zip



Aboot.exe会自动搜索本目录下的zip文件，选最新的zip文件为默认的下載包。如果工具自动选择的下载包不是客户的目标文件。请客户点击放大镜，在文件树中人工选择。

1. 点击Start按钮，进入下载过程。

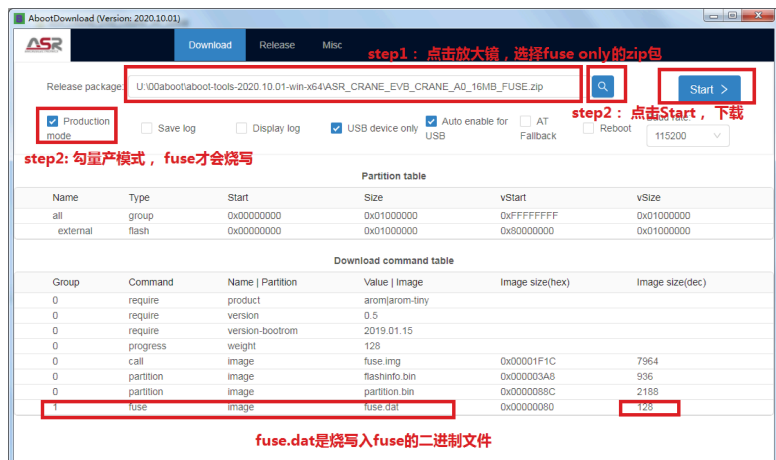


图 4.2. aboot.exe解析fuse only下载包

4.2. cmd命令行方式

命令行输入以下命令，通过帮助信息找到下载命令。

```
adownload.exe -h
```

cmd console.

```
U:\00aboot\aboot-tools-2020.10.01-win-x64>adownload.exe -h
Arsmicro aboot download console application.

Download release package FILE to boards.
Usage: adownload.exe [OPTION]... [FILE]
-h, --help                Display this help and exit
-p, --port=port           Use named serial ports separate with comma
-a, --auto-enable         Or auto enable arom usb ports device
-u, --usb-only            Use arom usb ports only
-d, --dump-enable        Enable dump download protocol packet
-s, --speed=speed        Use given speed for serial communication
--baud=speed             Supported baud rates:
                          (115200, 230400, 460800, 921600, 1842000,
                          3686400)
-m, --production          Running in mass prodcuton mode, default
is upgrade mode❶
-f, --at-fallback         Send AT cmd to fallback to download mode
-r, --reboot              Reboot device after finished
```

`-q, --quit` Quit application after any port finished

Example:

```
adownload.exe -p COM1,COM2 -s 115200 aboot.zip
adownload.exe -u -a -s 115200 aboot.zip
adownload.exe -p COM1 -a -s 115200 aboot.zip
```

<1>-m 是进入量产模式的烧写，要烧写fuse，前提必须是量产模式的烧写，请加-m。adownload无-m，就是默认方式，也就是开发模式烧写，不烧写fuse。

命令行输入以下命令，下载fuse only的包

```
adownload.exe -u -a -q -m -s 115200 ASR_CRANE_EVB_CRANE_A0_16MB_FUSE.zip
```

参数-u是只自动检测usb端口，不包括uart端口。

参数-a是自动检测到usb端口，就进入下载模式。

参数-q是下载结束以后退出程序。

参数-m是进入量产烧写模式。



如果adownload命令没有用-q，下载结束后不会自动退出。如果用户又用Aboot.exe这种UI方式运行下载程序，会遇到执行操作错误提示，如果要用Aboot.exe，请记得adownload要退出。两种工具不能同时处于下载模式。



如果download命令没有用-m， fuse在开发模式下不会烧写。

**cmd console.**

```
G:\about-tools-2020.10.01-win-x64>adownload.exe -u -a -q -m -s 115200
  ASR_CRANE_EVB_CRANE_A0_16MB_FUSE.zip
parsing command line paramters ...
usb only enabled.
usb port auto enabled.
serial baud rate specified at "115200"
running in production mode.
release package file name specified is
  "ASR_CRANE_EVB_CRANE_A0_16MB_FUSE.zip"
finished parsing command line paramters.
initializing about release package...
```

```
extracting download.json (652 Bytes)...
extracting partition.bin (2 KB)...
### ABOOT_EVENT_DOWNLOAD_INIT ###
initialized aboot release package successfully.
Starting lwIP, IPv4 disable
starting aboot download engine...
ip6 linklocal address: FE80::278:60FF:FE51:DC1C
download engine running in production mode!
extracting download.json (652 Bytes)...
### ABOOT_EVENT_DOWNLOAD_START ###
aboot download engine started successfully.
getting serial devices list...
```

把板子usb端接在PC的usb口，按住板子上的usb下载键，并上电，板子进入下载模式。

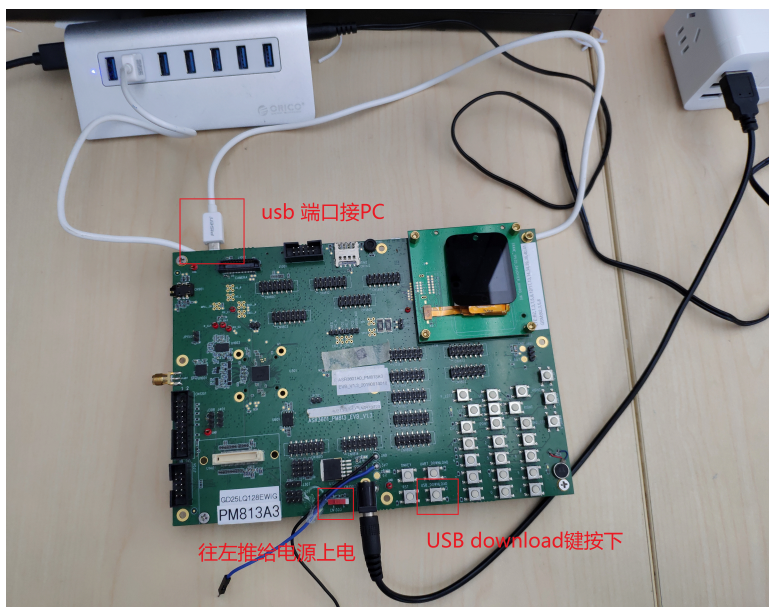


图 4.3. 板子的硬件接口图

烧写过程输出的log如下：

#### cmd console.

```
G:\xdb_patch\aboot-tools-2020.10.01-win-x64>adownload.exe -u -a -q -m -s
115200 ASR_CRANE_EVB_CRANE_A0_16MB_FUSE.zip
parsing command line paramters ...
usb only enabled.
```



```
usb port auto enabeld.
serial baud rate specified at "115200"
running in production mode.
release package file name specified is
  "ASR_CRANE_EVB_CRANE_A0_16MB_FUSE.zip"
finished parsing command line paramters.
initializing about release package...
extracting download.json (652 Bytes)...
extracting partition.bin (2 KB)...
### ABOUT_EVENT_DOWNLOAD_INIT ###
initialized about release package successfully.
Starting lwIP, IPV4 disable
starting about download engine...
ip6 linklocal address: FE80::278:60FF:FE51:DC1C
download engine running in production mode!
extracting download.json (652 Bytes)...
### ABOUT_EVENT_DOWNLOAD_START ###
about download engine started successfully.
getting serial devices list...
### ABOUT_EVENT_DEVICE_NEW ###
{
    "displayName" : "ASR Modem Device (COM3)",
    "enabled" : false,
    "event" : 5,
    "locationInfo" : "Port_#0003.Hub_#0005",
    "order" : 0,
    "path" : "COM3",
    "productId" : 12311,
    "progress" : 0,
    "status" : "ONLINE",
    "vendorId" : 11980
}
### ABOUT_EVENT_DEVICE_CHANGE ###
{
    "displayName" : "ASR Modem Device (COM3)",
    "enabled" : true,
    "event" : 6,
    "locationInfo" : "Port_#0003.Hub_#0005",
    "order" : 1,
    "path" : "COM3",
    "productId" : 12311,
    "progress" : 0,
    "status" : "ONLINE",
    "triggered" : false,
    "vendorId" : 11980
}
```

```
### ABOUT_EVENT_DEVICE_CHANGE ###
{
    "displayName" : "ASR Modem Device (COM3)",
    "enabled" : true,
    "event" : 6,
    "locationInfo" : "Port_#0003.Hub_#0005",
    "order" : 1,
    "path" : "COM3",
    "productId" : 12311,
    "progress" : 0,
    "status" : "CONNECTING",
    "triggered" : false,
    "vendorId" : 11980
}
<COM3> new device arrived.
enabling device <COM3> into downloading mode...
device <COM3> enabled successfully.
connecting to serial device <COM3>...
connected to serial device <COM3> successfully!
starting to fire device <COM3>...
device <COM3> fired successfully.
Reading from COM port (ReadIOCompletion): Unknown error code 31
<COM3> device offlined
extinguishing device <COM3>...
device <COM3> extinguished successfully.
disconnecting from serial device <COM3>...
received all devices closed message.
### ABOUT_EVENT_DEVICE_CHANGE ###
{
    "displayName" : "ASR Modem Device (COM3)",
    "enabled" : true,
    "event" : 6,
    "locationInfo" : "Port_#0003.Hub_#0005",
    "order" : 1,
    "path" : "COM??",
    "productId" : 12311,
    "progress" : 0,
    "status" : "OFFLINE",
    "triggered" : false,
    "vendorId" : 11980
}
disconnected from serial device <COM3> successfully!
### ABOUT_EVENT_DEVICE_CHANGE ###
{
    "displayName" : "ASR Modem Device (COM3)",
    "enabled" : true,
```

```
"event" : 6,
"locationInfo" : "Port_#0003.Hub_#0005",
"order" : 1,
"path" : "COM3",
"productId" : 12311,
"progress" : 0,
"status" : "ONLINE",
"triggered" : false,
"vendorId" : 11980
}
### ABOUT_EVENT_DEVICE_CHANGE ###
{
    "displayName" : "ASR Modem Device (COM3)",
    "enabled" : true,
    "event" : 6,
    "locationInfo" : "Port_#0003.Hub_#0005",
    "order" : 1,
    "path" : "COM3",
    "productId" : 12311,
    "progress" : 0,
    "status" : "CONNECTING",
    "triggered" : false,
    "vendorId" : 11980
}
<COM3> device online ❶
connecting to serial device <COM3>...
connected to serial device <COM3> successfully!
starting to fire device <COM3>...
device <COM3> fired successfully.
### ABOUT_EVENT_DEVICE_CHANGE ###
{
    "displayName" : "ASR Modem Device (COM3)",
    "enabled" : true,
    "event" : 6,
    "locationInfo" : "Port_#0003.Hub_#0005",
    "order" : 1,
    "path" : "COM3",
    "productId" : 12311,
    "progress" : 0,
    "status" : "RUNNING",
    "triggered" : false,
    "vendorId" : 11980
}
<COM3> processing command [get bootrom info]
<COM3> processing command [require product arom]
```

```

<COM3> Checking product
<COM3> OKAY [ 0.011s]
<COM3> processing command [require version 0.5]
<COM3> Checking version
<COM3> OKAY [ 0.013s]
<COM3> processing command [require version-bootrom 2019.01.15]
<COM3> Checking version-bootrom
<COM3> OKAY [ 0.018s]
<COM3> processing command [progress weight 0]
<COM3> setting total progress
<COM3> OKAY [ 0.013s]
<COM3> target reported max download size of 64768 bytes
<COM3> extracting fuse.img (7 KB)...
<COM3> processing command [call fuse.img]②
<COM3> Sending 'fuse.img' (7 KB)
<COM3> OKAY [ 0.082s]
<COM3> verifying
<COM3> OKAY [ 0.218s]
<COM3> calling
<COM3> #=> Executing fuse application...
<COM3> OKAY [ 0.010s]
<COM3> target reported max download size of 32768 bytes
<COM3> extracting fuse.dat (128 Bytes)...
<COM3> processing command [fuse fuse.dat]③
<COM3> Downloading 'fuse.dat'
<COM3> OKAY [ 0.018s]
<COM3> programming fuse blocks
<COM3> #=> fuse: sbe = 0, sys_boot_ctrl=0x8001
<COM3> #=> Non trust boot mode.④
<COM3> #=> fuse: secure_key_access_disable = 0, sys_boot_ctrl=0x8001
<COM3> #=> life_cycle before: 0x3
<COM3> #=> life_cycle after: 0x3
<COM3> #=> fuse set secure_boot_enable for trust boot mode.⑤
<COM3> #=> fuse disable aib override and strap pin
<COM3> #=> fuse: secure_key_access_disable = 0
<COM3> #=> fuse disable secure_key_access for aes root key⑥
<COM3> OKAY [ 0.032s]
<COM3> waiting for all operations to complete
<COM3> #=>

```

pid	name	state	Q	pri	stack
( used)	base addr   current				
<COM3> #=>	-   isr_stack	-	-	-	1024 (
	280)   0xd100f9c0   0xffffffff				
<COM3> #=>	1   idle	pending	Q	15	160 (
	128)   0xd1002a10   0xd1002a30				

```

<COM3> #=>          2 | main                      | bl rx    _ | 7 | 2560
( 1628) | 0xd1002ab0 | 0xd1003360
<COM3> #=>          3 | ipv6                      | bl rx    _ | 4 | 1024 (
452) | 0xd1007110 | 0xd10073f0
<COM3> #=>          4 | udp                      | bl rx    _ | 5 | 1024 (
292) | 0xd10078a4 | 0xd1007ba8
<COM3> #=>          5 | heartbeat                | bl rx    _ | 6 | 512 (
196) | 0xd10034d4 | 0xd1003628
<COM3> #=>          6 | ethos                    | bl rx    _ | 2 | 1024 (
416) | 0xd10025c4 | 0xd10028cc
<COM3> #=>          7 | aboot                    | bl rx    _ | 8 | 2560
( 1680) | 0xd100607c | 0xd10069d0
<COM3> #=>          8 | flasher                  | running  Q | 9 | 1536 (
592) | 0xd1006b08 | 0xd1006eb8
<COM3> #=>          | SUM                      |          |   | 11424
( 5664)
<COM3> OKAY [ 0.060s]
<COM3> all finished. total time: 0.534s⑦
stopping aboot download engine...
### ABOUT_EVENT_DOWNLOAD_STOP ###

```

- ❶ board进入下载模式后，PC端自动找到usb device的com口。
- ❷ fuse.img注册了烧写fuse的命令。
- ❸ fuse的命令烧写fuse.dat中的oem公钥hash数据至fuse的block2，算出随机数作为AES root key烧写到block1。
- ❹ fuse没有被烧写前，还是Non-Trusted模式。
- ❺ fuse的block2被烧入了oem公钥hash数据后，程序自动置位fuse中的block0的secure\_boot\_enable 比特为1。板子重启以后，启动阶段读到sbe=1，进入Trusted Boot模式。
- ❻ fuse的block1，AES的root key被烧写入一个随机数以后，程序自动置位Fuse中的block0的secure\_key\_access 比特为1。板子重启以后，GEU模块没有权限读出block1里的root key值，只有AES才有访问权限。
- ❼ 下载包烧写完毕。板子需要重启以后才会进入Trusted Boot模式。



# 5

## Fuse结构

Crane采用了TSMC HD FUSE，如图所示，Fuse由4个Block组成。每个Block有256Bits. 每个block的具体bit的定义请参见文档 "Crane register XLS file GEU part (SW side)"。

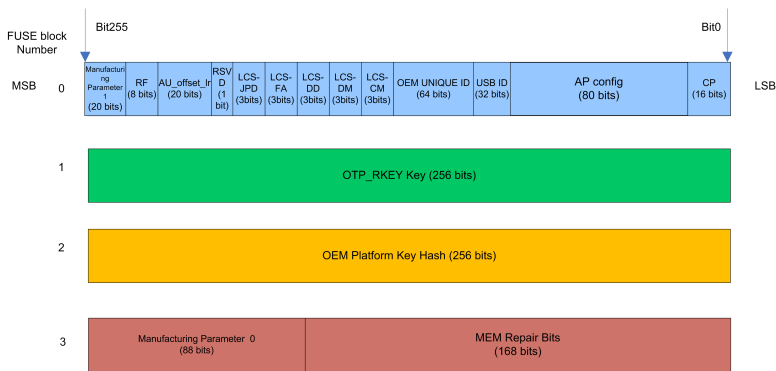


图 5.1. craneA0 Fuse比特分布

如上图所示，4个block的定义：

### 1. block0：256bits

- Manufacturing Parameter 1
- RF
- AU\_offset\_lir
- LCS(life cycle)
- OEM UNIQUE ID
- USB ID

- 
- AP config
  - CP config

## 2. block1 : 256bits

- OTP\_RKEY           Key : 唯一标识某个板子的随机数，供AES使用的Root Key。fuse命令执行的时候，算出随机数，写下Root key。 此key被写入以后，读取权限被限制，block0中的secure\_key\_access 比特被置1，从此GEU模块无权限读取其值，仅AES模块有权限读取。

## 3. block2 : 256bits

- OEM Platform Key Hash : oem的公钥Hash，用于验证镜像文件中的公钥是否可用。Arelease读取了这个文件的内容写入fuse的烧写包内。OEM公钥Hash被烧写一次以后，block0的secure\_boot\_enable 比特被置1。板子重启以后，读到sbe=1，进入Trusted模式，禁止对block2再改写。如果需要核实hash的数据内容,在烧写fuse前，可以打开打开下载包里的fuse.dat 或者打开制作包环境里的配置文件config\security\key\rsa\_rookpk\_sha256.bin。

## 4. block3 : 256bits

- Manufacturing Parameter 0
- MEM Repair Bits

Fuse的每个Bit写1有效，而且只能写一次。不用改写的Bit，需保持fuse命令输入的数据源中此Bit为0。 比如：fuse only的烧写包就只需要对应block2的256bits有价值，其他3个Block的bits为0。

Block1和Block3与配置有关，与安全启动无关，所以其中没有烧写过的Bits，如果客户有再次配置的需求，可以再次烧写fuse。



BLOCK1 烧写过以后就变成不可访问，所以fuse再次烧写时候，不会对已经有root key的block1再改写。



BLOCK2 OEM公钥Hash烧写一次以后，被设置Trusted模式sbe=1。所以fuse再次烧写时候，读到sbe=1，就不会对block2再改写。



# 6

## 重启模块

Fuse only下载包被烧写以后，必须重启板子。

重启过程中bootrom会读取sbe比特，如果sbe=1，启动进入Trusted boot模式。

如果要看板子启动的串口log，请按图接uart端口到PC。

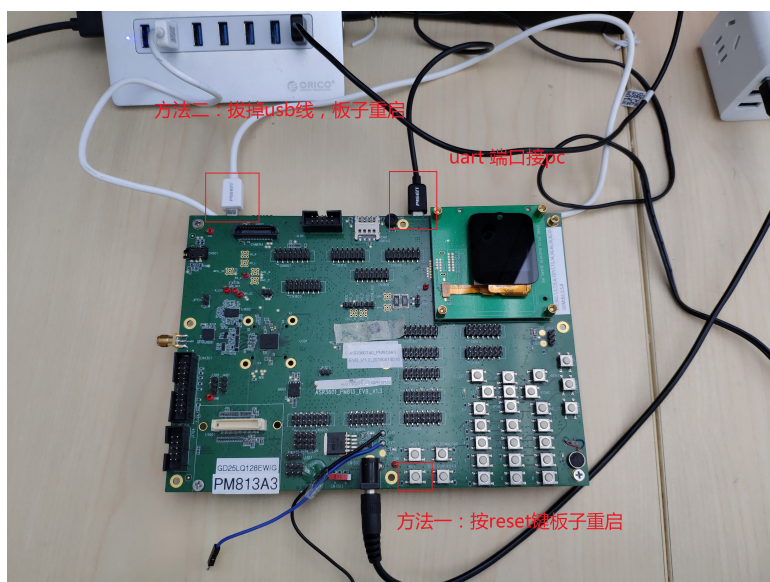


图 6.1. 板子的usb和uart的接口图

在PC端打开用sscom工具或者putty工具监测AP uart端口。

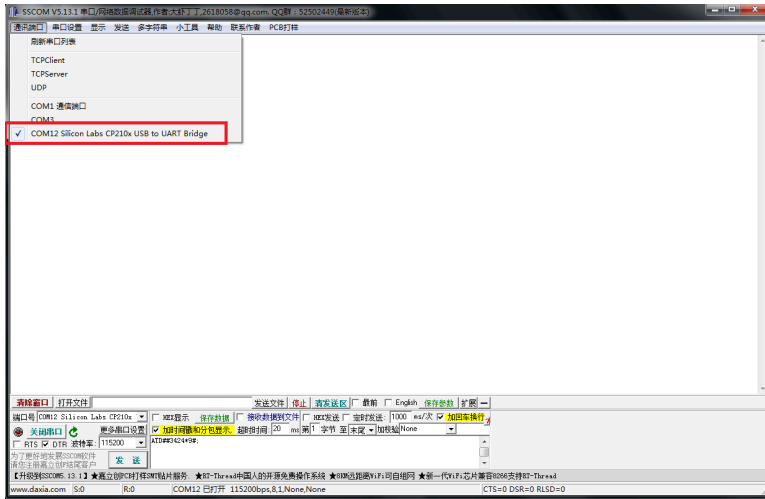


图 6.2. sscom看串口log

可以有这些方法重启模块：

- 按reset键，板子重启。
- 拔掉usb线，板子重启。

板子重启时候，sscom窗口看到串口log输入如下。

**sscom窗口.**

```
[16:39:00.416]收←◆CHIP_ID: 0x6731, REV_ID: 0xA0
PLATFORM: CRANE (SILICON)
CRANE hardware initialization complete.

AROM! (Version: 2019.01.15)
Initializing crypto library...
Loading...
welcome to the trusted boot rom world.
QSPI clock configured at 13 MHz
Detecting QSPI flash devices...
Found a known spi flash device "GD25LQ128D"
SPI nor flash: Manufacturer ID: 0xC8, Device ID: 0x6018
Detected peb size is 4096, the first good peb number is 0
Found preboot volume, size is 15528 bytes
BL1: Trying to load and verify BL2 image from volume preboot...
verify 6 failed, rc=2❶
BL1: Load or verify BL2 image failed.
```

```
[16:39:00.483]收-Found bootloader volume, size is 30460 bytes
BL1: Trying to load and verify BL2 image from volume bootloader...
verify 6 failed, rc=2
BL1: Load or verify BL2 image failed.
Load or verify bootloader image failed.
Booting failed, force into usb downloading mode.
```

- ❶ 板子flash中读出的是换key以前旧的preboot.img，其FIP头中的公钥经过hash比对，和fuse中存的oem公钥的hash不一致，验证报错，退出启动。说明fuse烧写已经生效。



需要进入下一下章节用新证书签名的镜像文件烧写固件。

如果板子重启以后。用户用fuse only下载包再次尝试fuse烧写，会看到如下错误提示。说明新证书已经生效了，拒绝用老证书进行fuse烧写。

#### cmd console.

```
G:\aboot-tools-2020.10.01-win-x64>adownload.exe -u -a -q -s 115200
ASR_CRANE_EVB_CRANE_A0_16MB_FUSE.zip
parsing command line paramters ...
auto quit enabeld.
usb only enabeld.
usb port auto enabeld.
serial baud rate specified at "115200"
running in production mode.
release package file name specified is "ASR_CRANE_EVB_CRANE_A0_16MB.zip"
finished parsing command line paramters.
initializing aboot release package...
extracting download.json (652 Bytes)...
extracting partition.bin (2 KB)...
### ABOT_EVENT_DOWNLOAD_INIT ###
initialized aboot release package successfully.
Starting lwIP, IPV4 disable
starting aboot download engine...
ip6 linklocal address: FE80::32:F0FF:FE77:5B6D
download engine running in production mode!
extracting download.json (652 Bytes)...
### ABOT_EVENT_DOWNLOAD_START ###
aboot download engine started successfully.
getting serial devices list...
### ABOT_EVENT_DEVICE_NEW ###
{
```

```
"displayName" : "ASR Modem Device (COM3)",
"enabled" : false,
"event" : 5,
"locationInfo" : "Port_#0003.Hub_#0005",
"order" : 0,
"path" : "COM3",
"productId" : 12311,
"progress" : 0,
"status" : "ONLINE",
"vendorId" : 11980
}
### ABOUT_EVENT_DEVICE_CHANGE ###
{
    "displayName" : "ASR Modem Device (COM3)",
    "enabled" : true,
    "event" : 6,
    "locationInfo" : "Port_#0003.Hub_#0005",
    "order" : 1,
    "path" : "COM3",
    "productId" : 12311,
    "progress" : 0,
    "status" : "ONLINE",
    "triggered" : false,
    "vendorId" : 11980
}
### ABOUT_EVENT_DEVICE_CHANGE ###
{
    "displayName" : "ASR Modem Device (COM3)",
    "enabled" : true,
    "event" : 6,
    "locationInfo" : "Port_#0003.Hub_#0005",
    "order" : 1,
    "path" : "COM3",
    "productId" : 12311,
    "progress" : 0,
    "status" : "CONNECTING",
    "triggered" : false,
    "vendorId" : 11980
}
<COM3> new device arrived.
enabling device <COM3> into downloading mode...
device <COM3> enabled successfully.
connecting to serial device <COM3>...
Opening COM3: Unknown error code 170
<COM3> device offlined
extinguishing device <COM3>...
```

```

### ABOUT_EVENT_DEVICE_CHANGE ###
{
    "displayName" : "ASR Modem Device (COM3)",
    "enabled" : true,
    "event" : 6,
    "locationInfo" : "Port_#0003.Hub_#0005",
    "order" : 1,
    "path" : "COM3",
    "productId" : 12311,
    "progress" : 0,
    "status" : "ONLINE",
    "triggered" : false,
    "vendorId" : 11980
}
### ABOUT_EVENT_DEVICE_CHANGE ###
{
    "displayName" : "ASR Modem Device (COM3)",
    "enabled" : true,
    "event" : 6,
    "locationInfo" : "Port_#0003.Hub_#0005",
    "order" : 1,
    "path" : "COM3",
    "productId" : 12311,
    "progress" : 0,
    "status" : "CONNECTING",
    "triggered" : false,
    "vendorId" : 11980
}
<COM3> device online
connecting to serial device <COM3>...
connected to serial device <COM3> successfully!
starting to fire device <COM3>...
device <COM3> fired successfully.
### ABOUT_EVENT_DEVICE_CHANGE ###
{
    "displayName" : "ASR Modem Device (COM3)",
    "enabled" : true,
    "event" : 6,
    "locationInfo" : "Port_#0003.Hub_#0005",
    "order" : 1,
    "path" : "COM3",
    "productId" : 12311,
    "progress" : 0,
    "status" : "RUNNING",
    "triggered" : false,
    "vendorId" : 11980
}

```

```

}
<COM3> processing command [get bootrom info]
<COM3> processing command [require product arom]
<COM3> Checking product
<COM3> OKAY [ 0.010s]
<COM3> processing command [require version 0.5]
<COM3> Checking version
<COM3> OKAY [ 0.014s]
<COM3> processing command [require version-bootrom 2019.01.15]
<COM3> Checking version-bootrom
<COM3> OKAY [ 0.013s]
<COM3> processing command [progress weight 0]
<COM3> setting total progress
<COM3> OKAY [ 0.013s]
<COM3> target reported max download size of 64768 bytes
<COM3> extracting fuse.img (7 KB)...
<COM3> processing command [call fuse.img]
<COM3> Sending 'fuse.img' (7 KB)
<COM3> OKAY [ 0.081s]
<COM3> verifying
<COM3> #=> verify 6 failed, rc=2❶
<COM3> FAILED (remote: image verify failed)
<COM3> all finished. total time: 0.175s
### ABOUT_EVENT_DEVICE_CHANGE ###
{
    "displayName" : "ASR Modem Device (COM3)",
    "enabled" : true,
    "event" : 6,
    "locationInfo" : "Port_#0003.Hub_#0005",
    "order" : 1,
    "path" : "COM3",
    "productId" : 12311,
    "progress" : 0,
    "status" : "FAILED",
    "triggered" : false,
    "vendorId" : 11980
}
stopping about download engine...
### ABOUT_EVENT_DOWNLOAD_STOP ###

```

❶ 下载包fuse.img的FIP头中的公钥经过hash比对，和fuse中存的oem公钥的hash不一致，验证报错，退出下载。说明fuse烧写已经生效。

重启板子后还必须按照下一章节操作，重新烧写固件。

镜像签名过程有两种方式：

- 自动签名模式：

UI界面或者命令行方式，在打包的过程中自动完成。

签名用的私钥保存在工具目录config/security/key，release打包过程直接从本地文件读到私钥，对固件的hash做数字签名。固件的二进制文件转化成镜像文件，为镜像文件添加了FIP的头，FIP中保存了OEM的公钥和数字签名。

- 分步的客户定制模式：

1. 客户提供公钥和私钥。客户的公钥文件覆盖工具的“config/security/key”目录中证书链文件，私钥文件保存在客户的服务器中。
2. 通过命令行方式执行第一阶段的命令，工具自动对固件的二进制文件(例如:preboot.bin)采用SHA256算法求出256个字节的哈希值(Hash)，以Hash的后32个字节作为文件名，内容为Hash值，生成文件，保存到本地目录。
3. 客户读取哈希文件，用远程的私钥对Hash做RSA签名，生成数字签名，生成文件，文件名仍然是Hash的后32个字节，扩展名是.hash.enc,保存在本地目录。
4. 再通过命令行方式执行第二阶段的命令，工具自动计算出固件的hash，以得到数字签名的文件名，读取数字签名文件，得到数字签名。固件的二进制文件转化成镜像文件，为镜像文件添加了FIP的头，FIP中保存了OEM的公钥和数字签名。

## 7.1. 如何自动镜像签名

用about工具UI界面方式或者arelease工具命令行方式自动对固件包做数字签名，并且打包。

### 7.1.1. UI界面方式

1. 勾选板子对应的产品型号。
2. 点击Release按钮开始做下载包。

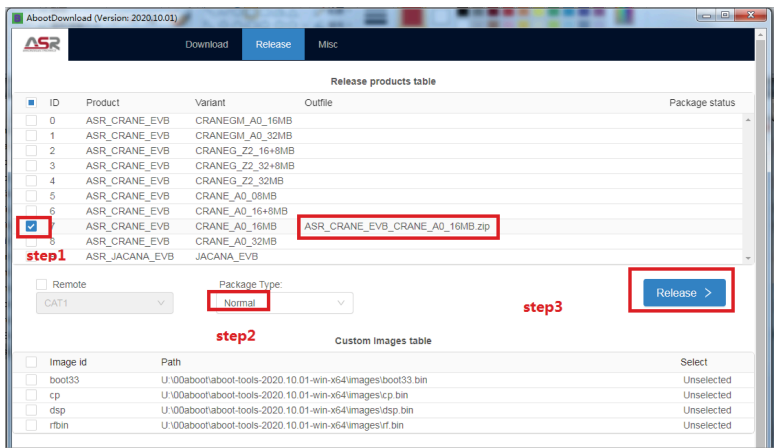


图 7.1. Aboot做下载包

下载包制作成功的界面如下



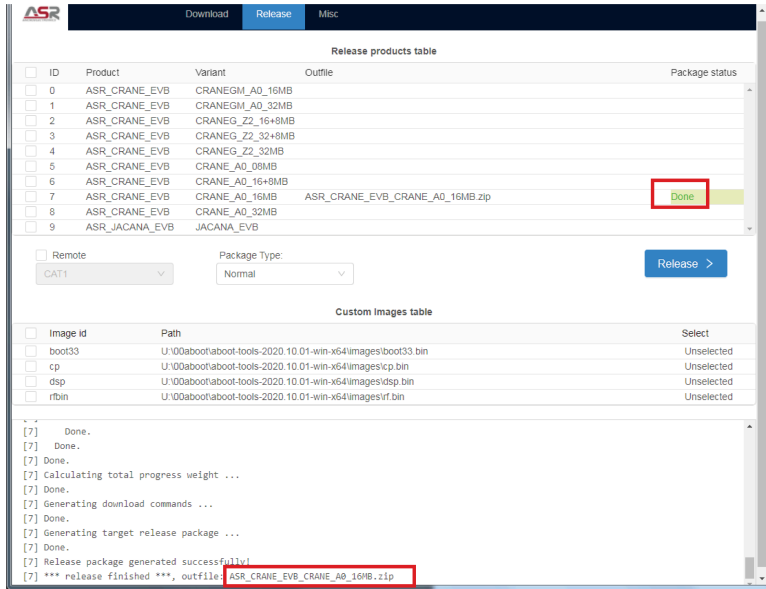


图 7.2. 下载包制作成功

## 7.1.2. 命令行方式

执行Windows的命令窗口cmd.exe。在console界面输入命令如下，

```
cd aboot-tools-<tagName>-win-x64
```

```
arelease.exe -h
```

### cmd console.

```
D:\00aboot\aboot-tools-2020.12.07-win-x64>arelease.exe -h
Asrmicro Aboot Release Application.
```

Generate release package for designated product.

Usage: arelease.exe [OPTION]... [FILE]

- h, --help Display this help and exit
- c, --config=config Products base config directory
- Omit for current directory
- I, --image-base=base Images base directory
- k, --keygen Generate security keys
- key-alg=alg Key algorithm: 'rsa' (default), 'ecdsa'
- l, --list List all supported products
- g, --generate Generate release package

```

--erase-all      Add erase all command before any flash
operations
--erase-all-only Only execute erase all command
--fuse-only       Only generate fuse commands
--upload-only     Only generate upload commands
--stage1          Generate hash files used by external encrypter
--stage2          Use external encrypted hash files to generate
certificates
--stage-dir       Out dir for stage1 or input dir for stage2
-p, --product=product Select which product to generate
-v, --variant=variant Select product variant
-i, --images=images Set image path for image id
                  E.X. -i cp=cp.bin,dsp=dsp.bin
[FILE]           The option FILE designated the output filename
                  if not designated, will use $product.zip as
default

```

## Example:

```

arelease.exe -c . -k --key-alg=rsa
arelease.exe -c . -l
arelease.exe -c . -g -p ASR_CRANE_EVB -v CRANE_A0_16MB❶
arelease.exe -c . -g --erase-all -p ASR_CRANE_EVB -v CRANE_A0_16MB
arelease.exe -c . -g --erase-all-only -p ASR_CRANE_EVB -v
CRANE_A0_16MB
arelease.exe -c . -g --fuse-only -p ASR_CRANE_EVB -v CRANE_A0_16MB
arelease.exe -c . -g --upload-only -p ASR_CRANE_EVB -v CRANE_A0_16MB
arelease.exe -c . -g -p ASR_CRANE_EVB -v CRANE_A0_16MB
ASR_CRANE_EVB.zip
arelease.exe -c . -g -p ASR_CRANE_EVB -v CRANE_A0_16MB --stage1 --
stage-dir=out
arelease.exe -c . -g -p ASR_CRANE_EVB -v CRANE_A0_16MB --stage2 --
stage-dir=in
arelease.exe -c config -I images -g -p ASR_CRANE_EVB -v CRANE_A0_16MB

```

- ❶ 从help帮助信息中可以看到此命令。-v是选择-I中查询到的对应自己的板子的产品类型。

通过-I参数查询到的产品类型与板子的芯片和flash大小有关。

```
arelease.exe -l
```

## cmd console.

```

D:\00aboot\aboot-tools-2020.12.07-win-x64>arelease -l
Parsing command line paramters ...

```

Finished parsing command line paramters.

Supported product list:

ORDER	PRODUCT	VARIANT
0.	ASR_CRANE_EVB	
0.0		CRANEGM_A0_16+8MB
0.1		CRANEGM_A0_16MB <sup>❶</sup>
0.2		CRANEGM_A0_32MB <sup>❷</sup>
0.3		CRANEGM_A0_8MB
0.4		CRANEG_Z2_16+8MB <sup>❸</sup>
0.5		CRANEG_Z2_32+8MB
0.6		CRANEG_Z2_32MB
0.7		CRANE_A0_08MB <sup>❹</sup>
0.8		CRANE_A0_16+8MB <sup>❺</sup>
0.9		CRANE_A0_16MB <sup>❻</sup>
0.10		CRANE_A0_32MB <sup>❼</sup>
1.	ASR_JACANA_EVB	
1.0		JACANA_EVB

- ❶ CRANEG\_A0或者CRANEM\_A0芯片，QSPI flash是16MB。CRANEM\_A0的QSPI flash是芯片内部。
- ❷ CRANEG\_A0或者CRANEM\_A0芯片，QSPI flash是32MB。
- ❸ CRANEG\_Z2芯片，外部QSPI flash是16MB，内部SPI flash是8MB
- ❹ CRANE\_A0的芯片，外部QSPI flash是8MB。
- ❺ CRANE\_A0的芯片，外部QSPI flash是16MB，内部SPI flash是8MB。
- ❻ CRANE\_A0的芯片，外部QSPI flash是16MB。
- ❼ CRANE\_A0的芯片，外部QSPI flash是32MB。

如何从启动log中确认是不是有内部SPI flash：

**sscom or putty console.**

```
2020-03-30 15:59:27,230 INFO # crane log: #####
2020-03-30 15:59:27,230 INFO # crane log: Version ID : 0
2020-03-30 15:59:27,230 INFO # crane log: AP 16MB
2020-03-30 15:59:27,230 INFO # crane log: No embedded flash.❶
2020-03-30 15:59:27,230 INFO # crane log: #####
```

- ❶ 没有内置的8MBytes的spi flash。

**sscom or putty console.**

```
2020-03-30 15:59:27,230 INFO # crane log: #####
```

```
2020-03-30 15:59:27,230 INFO      # crane log: Version ID : 0
2020-03-30 15:59:27,230 INFO      # crane log: AP 16MB
2020-03-30 15:59:27,230 INFO      # crane log: Has embedded flash.❶
2020-03-30 15:59:27,230 INFO      # crane log: #####
```

❶ 有内置的8MBytes的spi flash。

输入以下命令：

```
arelease.exe -c . -g -p ASR_CRANE_EVB -v CRANE_A0_16MB
```

自动生成下载包ASR\_CRANE\_EVB\_CRANE\_A0\_16MB.zip。生成的文件名与-v选定的产品类型一致。

## cmd console.

```
G:\aboot-tools-2020.10.01-win-x64>arelease.exe -c . -g -p ASR_CRANE_EVB
-v CRANE_A0_16MB
Parsing command line paramters ...
Release package file name not specified, use product name as base.
Finished parsing command line paramters.
Processing all images needed ...
    Generating fip image "preboot" with id "preboot.img" ...❶
        Opening image "preboot_bin" from "G:\aboot-tools-2020.10.01-win-
x64\images\2019.01.15\preboot.bin" ...
        Done.
    Done.
    Generating fip image "flasher" with id "flasher.img" ...❷
        Opening image "flasher_bin" from "G:\aboot-tools-2020.10.01-win-
x64\images\2019.01.15\flasher.bin" ...
        Done.
    Done.
    Generating finf image "flashinfo" with id "flashinfo.bin" ...
    Done.
    Opening image "rd" from "G:\aboot-tools-2020.10.01-win-x64\images
\ReliableData.bin" ...
    Done.
    Opening image "apn" from "G:\aboot-tools-2020.10.01-win-x64\images
\apn.bin" ...
    Done.
    Opening image "cp" from "G:\aboot-tools-2020.10.01-win-x64\images
\cp.bin" ...
    Done.
```

```

Opening image "dsp" from "G:\aboot-tools-2020.10.01-win-x64\images
\dsp.bin" ...
Done.
Opening image "rfbin" from "G:\aboot-tools-2020.10.01-win-x64\images
\rf.bin" ...
Done.
Opening image "updater" from "G:\aboot-tools-2020.10.01-win-x64\images
\updater.bin" ...
Done.
Generating fuse image "fuse" with id "fuse.dat" ... ❸
Done.
Generating fip image "fwcerts" with id "fwcerts.bin" ...
  Using already opened/generated image "cp" with id "cp.bin" ...
  Done.
  Using already opened/generated image "dsp" with id "dsp.bin" ...
  Done.
Done.
Generating aptb image "partition" with id "partition.bin" ...
Done.
Generating ubi image "bootloader" with id "bootloader.ubi" ...
  Using already opened/generated image "flashinfo" with id
"flashinfo.bin" ...
  Done.
  Using already opened/generated image "partition" with id
"partition.bin" ...
  Done.
  Using already opened/generated image "preboot" with id
"preboot.img" ...
  Done.
  Generating fip image "bootloader_img" with id "bootloader.img" ...
    Opening image "boot2_bin" from "G:\aboot-tools-2020.10.01-win-
x64\images\2019.01.15\boot2.bin" ...
    Done.
    Opening image "boot33" from "G:\aboot-tools-2020.10.01-win-
x64\images\boot33.bin" ...
    Done.
  Done.
Done.
Generating fota images ...
  Generating group image "system" with id "system.img" ...
    Using already opened/generated image "partition" with id
"partition.bin" ...
    Done.
    Using already opened/generated image "fwcerts" with id
"fwcerts.bin" ...

```

```

Done.
Using already opened/generated image "rd" with id
"ReliableData.bin" ...
Done.
Using already opened/generated image "apn" with id "apn.bin" ...
Done.
Using already opened/generated image "cp" with id "cp.bin" ...
Done.
Using already opened/generated image "dsp" with id "dsp.bin" ...
Done.
Using already opened/generated image "rfbin" with id "rf.bin" ...
Done.
Done.
Done.
Calculating total progress weight ...
Done.
Generating download commands ...
Done.
Generating target release package ...
Done.
Release package generated successfully!

```

- ❶ preboot.bin被用config\security\key\中rsa的文件换成的私钥做了数字签名了，添加fip头，fip头中保存了OEM的公钥和镜像文件的数字签名。
- ❷ flasher.bin也被数字签名了，封装成flasher.img
- ❸ fuse.dat, aboot-tools-2020.10.01-win-x64\config\product\ASR\_CRANE\_EVB.json中客户配置了secureBoot=true，所以release过程会生成fuse.dat包。UI界面的非量产的固件下载的过程默认不会勾选烧写fuse.dat。



fuse的block1 和block2如果烧写过bit了，就会跳过对这个block区域的改写。block0和block3如果需要改写配置，客户定制了新的针对config补充配置的fuse.dat, 用户可以手工勾选上fuse命令。

如果解压下载包，可以看到包里的文件如下：

#### cmd console.

```

G:\aboot-tools-2020.10.01-win-x64>cd ASR_CRANE_EVB_CRANE_A0_16MB
G:\aboot-tools-2020.10.01-win-x64\ASR_CRANE_EVB_CRANE_A0_16MB>dir

```

2020/03/26	00:32	12,015	apn.bin
2020/03/26	17:49	98,304	bootloader.ubi
2020/03/26	00:32	3,947,242	cp.bin
2020/03/26	17:49	2,845	download.json
2020/03/26	00:32	995,658	dsp.bin
2020/03/26	17:49	31,272	flasher.img
2020/03/26	17:49	720	flashinfo.bin
2020/03/26	17:49	101	fota.json
2020/03/26	17:49	128	fuse.dat
2020/03/26	17:49	2,232	fwcerts.bin
2020/03/26	17:49	2,184	partition.bin
2020/03/26	17:49	16,428	preboot.img
2020/03/26	00:32	65,536	ReliableData.bin
2020/03/26	00:32	20,480	rf.bin
2020/03/26	17:49	5,083,136	system.img
2020/03/26	00:32	4,096	updater.bin

## 7.2. 如何分步实现的客户定制的镜像签名

仅支持命令行的方式。

### 7.2.1. 客户替换证书链文件保存公钥

config\security\key目录下存放着证书链文件。如下，

**cmd console.**

```
D:\00aboot\aboot-tools-2020.12.07-win-x64\config\security\key>dir
2020/11/30  14:42                241  ecdsa_non_trusted_world_key.pem
2020/11/30  14:42                241  ecdsa_nt_fw_key.pem
2020/11/30  14:42                241  ecdsa_nt_os_key.pem
2020/11/30  14:42                 32  ecdsa_rootpk_sha256.bin
2020/11/30  14:42                241  ecdsa_root_key.pem
2020/11/30  14:42                241  ecdsa_scp_fw_key.pem
2020/11/30  14:42                241  ecdsa_soc_fw_key.pem
2020/11/30  14:42                241  ecdsa_tos_fw_key.pem
2020/11/30  14:42                241  ecdsa_trusted_world_key.pem
2020/11/30  15:14                451  rsa_non_trusted_world_key.pem
2020/11/30  15:14                451  rsa_nt_fw_key.pem
2020/11/30  15:14                451  rsa_nt_os_key.pem
2020/11/30  15:33                 32  rsa_rootpk_sha256.bin
2020/11/30  15:14                451  rsa_root_key.pem
2020/11/30  15:14                451  rsa_scp_fw_key.pem
2020/11/30  15:14                451  rsa_soc_fw_key.pem
```

2020/11/30	15:14	451	rsa_tos_fw_key.pem
2020/11/30	15:14	451	rsa_trusted_world_key.pem

工具自带的证书链文件保存的是私钥的信息。

举例：rsa\_root\_key.pem

**cmd console.**

```
-----BEGIN PRIVATE KEY-----
MIIeWAIBADANBgkqhkiG9w0BAQEFAASCBKowggSmAgEAAoIBAQC9EdtmGTfTeTCY
ZHXLU/CjgxLhmtF7mIegYqASZJeyZVeF9lwpODdsi6iGmtJmgA2fcYckQlQ1a1p
4F1p8cBh1w8X9bYAY03qLP0AG/EmveqFMhPHIwGT/ZGaN5jF6JlV67ppgSykS2pR
BEDWQSBkyJx/dwsSj+axpx03Yw1xgvt0IDRiB/hz2kr+WMqYQGk750N8zA7HH7/A
Lgr0Y7LhRG7H1t8CQNrz1nCs8PGGoJ/Krm/j7h6+L+CKIS81zYd4c6yEOMsiBc
QF+EzVHt2croDn+403+6n58TQGQCFiesIwQA1HAIurYjbv9VlwoH4QrMKxf49AvL
szuG2rbfAgMBAECggEBALg3KXN26AJ+urQm12PdGmr3hRTlug+kjqtXXvhj8y/
gMPb1lyof88c+pg0kyKGQaM776onMH1Qm7+LadoFVTsggExMosD1FP+toB2yyYlJ
NdZMeAHYqv5yYW4vEM4ZvzsRrSDarbtFg92wRFRnz/VL1w1fCHhuv5eo/6lVejnr
e8IpMyLaDFLXoeCvjJAPOfCBdShIz/mCFAj3hAurSyxXg4AMiC6TXi8Q+toNQCXQ
Es60r+xw9zJa/TpcPuDf9XRfWvpXpCJFY2iKkI5sB08P7HBnM6bRactP+sUNG3K2
KVVGfBUuvPtJX+uNvnnxiUGJrJ03H0RgAwK1zid58CkCgYEA4kaApe3F7NHwkoAA
PCUMmRyRUE40b8wwenN6BbMUE5z9iMUH+Yx+2BU5u5ZMdo01oad+Awgv7orOkz0/
5FL+wIGNLSor6LyXW0nvwCLMB2KfCwXpMg4aAD40zE7mxwO/s3nHdpC9kyw13p5S
kZa0fk7XaduU20jnTTSngCdIx50CgYEA3JB3COVZK1rBtgZ3lqhK2YKY3VK9zUUy
IEJ7X6eLIYluIC50EQf1rvHkMXLF/sdwPQC+eGA2UWbNEPlgJXgHyuTMY/RVR7V8
vfjQyZK19/C3b+Ick1c6Fom3ciyWPK7bvz938S7A2515P9Kbq0RnNJUf0lwlVOHx
eLhi8QwblascGyEaz8se6UHpuQkam36guJrtbsgtL6gF6fUBr/HL50Aht1z8F1WZ
dTnoYlDbpUdpBxL0ve9yVSzfVagK/HNRjUQKYAyfsRIpUCmgcy6h1TgfvCjL8tUz
5Zu9l0toe4cztbtcmF0FJ4TUEwVHxAbUp0qnyLhnaHkmoFuBzgzEpZnt6qECgYEA
ihz0TN/AvIMyNZdLH032wMkki8KaE4W247h9pOI3ICn8ga49ODDnxGq36LtiVC2K
vmhQfPRSTxa8s0ZUZGGjD4vB6P1AwYyFtjeuFu+EPuO/GhJObv1LhH1SH9SHpywf
clx98MrD/T7XS/3HeLF2ptiSG2sz9mzNVjivzBkk2c0CgYEAjvmh8GkbZ1KlqApa
uaGOrF/TlvxPyYDSKL9eAGIBrMMumjgAkGE2qbzDUvhCXX/G8Yr4lAXwSN0rg2od
ba7T4ztipZDixVrfYm42UK87dyP+D7JaziQFeJQejrSue/4u3XL/Naw7F4zFKkx
eq6Quzdu1okLGX8Uz+5fB9U4Nbo=
-----END PRIVATE KEY-----
```

客户需要把自己服务器上的私钥对应的公钥文件取出。覆盖工具里的证书链文件。

举例：rsa\_root\_key.pem

**cmd console.**



```

-----BEGIN PUBLIC KEY-----
MIIBIjANBgqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA58XquNqlqDsUOsIHphUR
8ln2BYg9TaGDBTRh+7C7+ep0ev7vxakwaBPgtJmtI8WjvCYAmu0a/RJqgPW3lb+f
n0cBxxQvJvtVfpimLF8y7g0jLMddDK7qggWUU0+2Pe5GNW/jH4ehCVMLmDi9/13x
KDnPr1hZTzKwZZI/soZD4aOWP1L45KUS6NfPwBieh53sE1fohziYwqBhROVn2gDU
nmq1i+n0ZNftZ2oAH394cTHzL5/15ZGIo62N7X1cNaDIRz5WQe8RFVOnEzWHq6ML
Qt6Dyc+sZ8VoBVQqD++sgGvP89eA/nI3DBHjzkbAO4o19UqkcHoiLh1K1qhQejJz
VwIDAQAB
-----END PUBLIC KEY-----

```

## 7.2.2. 客户生成公钥Hash

用下面的命令生成rootkey的hash值。制作fuse only包时需要读此hash值。

```

openssl rsa -in rsa_root_key.pem -pubin -outform der | openssl dgst -
sha256 -binary -out rsa_rootpk_sha256.bin

```

## 7.2.3. 从help信息中找到第一步和第二步的命令。

执行Windows的命令窗口cmd.exe。在console界面输入命令如下，

```
cd aboot-tools-<tagName>-win-x64
```

```
arelease.exe -h
```

### cmd console.

```

D:\00aboot\aboot-tools-2020.12.07-win-x64>arelease.exe -h
Asrmicro Aboot Release Application.

```

Generate release package for designated product.

Usage: arelease.exe [OPTION]... [FILE]

```

-h, --help           Display this help and exit
-c, --config=config  Products base config directory
                        Omit for current directory
-I, --image-base=base Images base directory
-k, --keygen          Generate security keys
    --key-alg=alg     Key algorithm: 'rsa' (default), 'ecdsa'
-l, --list            List all supported products
-g, --generate        Generate release package

```

```

--erase-all      Add erase all command before any flash
operations
--erase-all-only Only execute erase all command
--fuse-only       Only generate fuse commands
--upload-only     Only generate upload commands
--stage1          Generate hash files used by external encrypter
--stage2          Use external encrypted hash files to generate
certificates
--stage-dir       Out dir for stage1 or input dir for stage2
-p, --product=product Select which product to generate
-v, --variant=variant Select product variant
-i, --images=images Set image path for image id
                  E.X. -i cp=cp.bin,dsp=dsp.bin
[FILE]           The option FILE designated the output filename
                  if not designated, will use $product.zip as
default

```

Example:

```

arelease.exe -c . -k --key-alg=rsa
arelease.exe -c . -l
arelease.exe -c . -g -p ASR_CRANE_EVB -v CRANE_A0_16MB
arelease.exe -c . -g --erase-all -p ASR_CRANE_EVB -v CRANE_A0_16MB
arelease.exe -c . -g --erase-all-only -p ASR_CRANE_EVB -v
CRANE_A0_16MB
arelease.exe -c . -g --fuse-only -p ASR_CRANE_EVB -v CRANE_A0_16MB
arelease.exe -c . -g --upload-only -p ASR_CRANE_EVB -v CRANE_A0_16MB
arelease.exe -c . -g -p ASR_CRANE_EVB -v CRANE_A0_16MB
ASR_CRANE_EVB.zip
arelease.exe -c . -g -p ASR_CRANE_EVB -v CRANE_A0_16MB --stage1 --
stage-dir=out❶
arelease.exe -c . -g -p ASR_CRANE_EVB -v CRANE_A0_16MB --stage2 --
stage-dir=in❷
arelease.exe -c config -I images -g -p ASR_CRANE_EVB -v CRANE_A0_16MB

```

- ❶ 第一步，生成固件的Hash，把hash写入文件，保存在目录stage-dir
- ❷ 第二步，从stage-dir中读出和固件对应的数字签名文件，读出数字签名写入Fip头。完成镜像签名。

## 7.2.4. 新建一个本地目录保存Hash文件

```
mkdir out
```

## 7.2.5. 执行第一步命令行

```
arelease.exe -c . -g -p ASR_CRANE_EVB -v CRANE_A0_16MB --stage1 --stage-dir=out
```

--stage-dir 是用于存放输出Hash文件的目录

### cmd console.

```
D:\00aboot\aboot-tools-2020.12.07-win-x64>arelease.exe -c . -g -p
ASR_CRANE_EVB -v CRANE_A0_16MB --stage1 --stage-dir=out
Parsing command line paramters ...
Finished parsing command line paramters.
Processing all images needed ...
    Generating fip image "preboot" with id "preboot.img" ...
        Opening image "preboot_bin" from "D:\00aboot\aboot-tools-2020.12.07-
win-x64\images\2019.01.15\preboot.bin" ...
        Done.
    Done.
    Generating fip image "flasher" with id "flasher.img" ...
        Opening image "flasher_bin" from "D:\00aboot\aboot-tools-2020.12.07-
win-x64\images\2019.01.15\flasher.bin" ...
        Done.
    Done.
    Processing "ubi" image "bootloader" with id "bootloader.ubi"
        Using already opened/generated image "preboot" with id
"preboot.img" ...
        Done.
        Generating fip image "bootloader_img" with id "bootloader.img" ...
            Opening image "boot2_bin" from "D:\00aboot\aboot-tools-2020.12.07-
win-x64\images\2019.01.15\boot2.bin" ...
            Done.
            Generating lzma image "boot33_lzma" with id "boot33.lzma" ...
                Opening image "boot33" from "D:\00aboot\aboot-tools-2020.12.07-
win-x64\images\boot33.bin" ...
                Done.
            Done.
        Done.
    Done.
    Generating fip image "fwcerts" with id "fwcerts.bin" ...
        Opening image "cp" from "D:\00aboot\aboot-tools-2020.12.07-win-
x64\images\cp.bin" ...
        Done.
        Opening image "dsp" from "D:\00aboot\aboot-tools-2020.12.07-win-
x64\images\dsp.bin" ...
```

```
Done.
Done.
Done.
Generate stage1 files successfully!
```

完成第一步后，out目录下新生成了需要签名的固件文件的Hash文件。

### cmd console.

```
D:\00aboot\aboot-tools-2020.12.07-win-x64>cd out

D:\00aboot\aboot-tools-2020.12.07-win-x64\out>dir
2020/12/07  16:25                256
13616abaa76acae608a8bd8101d62c641c23d666b402090174323495a6c72832.hash
2020/12/07  16:25                256
3d3bc11ac40ae7cbf57680133472b876ef4dbe9f9ad833839c68a60281f13fdf.hash
2020/12/07  16:25                256
689f5b7f1f28c3991f42f88c302c013161b7386efdbea0cb3d40891b2c300b7c.hash
2020/12/07  16:25                256
7400ba0e7f0546762d16e9fc6702b1a74c7aeb8c89f48cea1990f7f7d5977e94.hash
2020/12/07  16:25                256
84ca08b16a1bf22202158a62878561f82227a3d326b0607a730c518c2d47885d.hash
2020/12/07  16:25                256
94d020f479f09d98ddc7ff33947c789cccc702f58a3a165deeee5884a099f2c3.hash
2020/12/07  16:25                256
b03d12741694a84f68d340b1fb49d8a2409cd9393d48a42ad43bab60e58440c0.hash
2020/12/07  16:25                256
d8f526279041cc4032c51f5d58496c84c0820f3899d5733044225eb857f71ac2.hash
```

Hash文件中有256个字节。后32个字节作为文件名。Hash是对固件二进制数据进行SHA256哈希计算的结果。

比如preboot.bin是需要签名的固件，就会有对应的Hash文件生成。看下面的两个图。文件名“3d3b11ac.....df”,对应的hash文件内容的后32个字节，“0x3D 0x3B 0xC1 0x1A .....0xDF”。

图 7.3. 生成的Hash文件名

图 7.4. 生成的Hash文件的内容

## 7.2.6. 客户实现对Hash数字签名

客户从本地读取Hash文件，

例如：

```
13616abaa76acae608a8bd8101d62c641c23d666b402090174323495a6c72832.hash
```

用客户服务器中的私钥对Hash做数字签名。把签名文件保存成文件。例如保存在in的目录中。数字签名文件名同Hash的文件名，但扩展名改为.hash.enc。

例如：

```
13616abaa76acae608a8bd8101d62c641c23d666b402090174323495a6c72832.hash.enc
```

客户保存的数字签名文件：

**cmd console.**

```
D:\00aboot\aboot-tools-2020.12.07-win-x64\in>dir
2020/12/07  16:25                256
    13616abaa76acae608a8bd8101d62c641c23d666b402090174323495a6c72832.hash.enc
2020/12/07  16:25                256
    3d3bc11ac40ae7cbf57680133472b876ef4dbe9f9ad833839c68a60281f13fdf.hash.enc
2020/12/07  16:25                256
    689f5b7f1f28c3991f42f88c302c013161b7386efdbea0cb3d40891b2c300b7c.hash.enc
2020/12/07  16:25                256
    7400ba0e7f0546762d16e9fc6702b1a74c7aeb8c89f48cea1990f7f7d5977e94.hash.enc
2020/12/07  16:25                256
    84ca08b16a1bf22202158a62878561f82227a3d326b0607a730c518c2d47885d.hash.enc
2020/12/07  16:25                256
    94d020f479f09d98ddc7ff33947c789cccc702f58a3a165deeee5884a099f2c3.hash.enc
2020/12/07  16:25                256
    b03d12741694a84f68d340b1fb49d8a2409cd9393d48a42ad43bab60e58440c0.hash.enc
2020/12/07  16:25                256
    d8f526279041cc4032c51f5d58496c84c0820f3899d5733044225eb857f71ac2.hash.enc
```

## 7.2.7. 执行第二步命令行

```
arelease.exe -c . -g -p ASR_CRANE_EVB -v CRANE_A0_16MB --stage2 --stage-
dir=in
```

--stage-dir 是用于存放输入的数字签名文件的目录

软件再次对固件数据求hash，其值与第一步的hash值相同，从hash的后32个字节得到数字签名对应的文件名，找到了对应的数字签名文件后，读取数字签名，并写入image的FIP头，完成镜像文件签名。签名后的文件打包成输出的zip文件，例如：ASR\_CRANE\_EVB\_CRANE\_A0\_16MB.zip。

### cmd console.

```
D:\00aboot\aboot-tools-2020.12.07-win-x64>arelease.exe -c . -g -p
ASR_CRANE_EVB -v CRANE_A0_16MB --stage2 --stage-dir=in
Parsing command line paramters ...
Release package file name not specified, use product name as base.
Finished parsing command line parameters.
Processing all images needed ...
    Generating fip image "preboot" with id "preboot.img" ...
        Opening image "preboot_bin" from "D:\00aboot\aboot-tools-2020.12.07-
win-x64\images\2019.01.15\preboot.bin" ...
        Done.
    Done.
    Generating fip image "flasher" with id "flasher.img" ...
        Opening image "flasher_bin" from "D:\00aboot\aboot-tools-2020.12.07-
win-x64\images\2019.01.15\flasher.bin" ...
        Done.
    Done.
    Generating finf image "flashinfo" with id "flashinfo.bin" ...
    Done.
    Opening image "rd" from "D:\00aboot\aboot-tools-2020.12.07-win-
x64\images\ReliableData.bin" ...
    Done.
    Opening image "apn" from "D:\00aboot\aboot-tools-2020.12.07-win-
x64\images\apn.bin" ...
    Done.
    Opening image "cp" from "D:\00aboot\aboot-tools-2020.12.07-win-
x64\images\cp.bin" ...
    Done.
    Opening image "dsp" from "D:\00aboot\aboot-tools-2020.12.07-win-
x64\images\dsp.bin" ...
    Done.
    Opening image "rfbin" from "D:\00aboot\aboot-tools-2020.12.07-win-
x64\images\rf.bin" ...
    Done.
    Opening image "updater" from "D:\00aboot\aboot-tools-2020.12.07-win-
x64\images\updater.bin" ...
    Done.
    Generating fip image "fwcerts" with id "fwcerts.bin" ...
        Using already opened/generated image "cp" with id "cp.bin" ...
```

```
Done.
Using already opened/generated image "dsp" with id "dsp.bin" ...
Done.
Done.
Generating aptb image "partition" with id "partition.bin" ...
Done.
Generating ubi image "bootloader" with id "bootloader.ubi" ...
Using already opened/generated image "flashinfo" with id
"flashinfo.bin" ...
Done.
Using already opened/generated image "partition" with id
"partition.bin" ...
Done.
Using already opened/generated image "preboot" with id
"preboot.img" ...
Done.
Generating fip image "bootloader_img" with id "bootloader.img" ...
Opening image "boot2_bin" from "D:\00aboot\aboot-tools-2020.12.07-
win-x64\images\2019.01.15\boot2.bin" ...
Done.
Generating lzma image "boot33_lzma" with id "boot33.lzma" ...
Opening image "boot33" from "D:\00aboot\aboot-tools-2020.12.07-
win-x64\images\boot33.bin" ...
Done.
Done.
Done.
Done.
Generating fota images ...
Generating group image "system" with id "system.img" ...
Using already opened/generated image "partition" with id
"partition.bin" ...
Done.
Using already opened/generated image "fwcerts" with id
"fwcerts.bin" ...
Done.
Using already opened/generated image "rd" with id
"ReliableData.bin" ...
Done.
Using already opened/generated image "apn" with id "apn.bin" ...
Done.
Using already opened/generated image "cp" with id "cp.bin" ...
Done.
Using already opened/generated image "dsp" with id "dsp.bin" ...
Done.
Using already opened/generated image "rfbin" with id "rf.bin" ...
Done.
```

```

Done.
Done.
Done.
Calculating total progress weight ...
Done.
Generating download commands ...
Done.
Generating target release package ...
Done.
Release package generated successfully!

D:\00aboot\aboot-tools-2020.12.07-win-x64>

```

### 7.3. 如何使能Boot33的安全检验功能

工具默认都会对cp.bin签名，并把证书存入fwcerts.bin中。默认的boot33不对cp的bin验签。如果客户需要使能验签的功能，在boot33的编译阶段，修改boot33的在Makefile，启用如下配置行。在boot33运行阶段，即会对被签名image进行摘要验签，验签失败会走power down流程关机。

#### startup/bootloader/Makefile.

```

52 ##### BOOT33 SECBOOT START #####
53 # enable below part for BOOT33 SECBOOT support
54
55 #cflag += -DMBEDTLS_CONFIG_FILE=\"mbedtls_config.h\" \❶
56     -DTF_MBEDTLS_KEY_ALG_ID=1 \
57     -DTF_MBEDTLS_HASH_ALG_ID=1 \
58     -DBOOT33_SECBOOT_SUPPORT
59 #srcs += src/fip.c \❷
60     src/secboot.c \
61     src/asn1parse.c \
62     src/oid.c \
63     src/sha256.c \
64     src/md.c \
65     src/x509.c \
66     src/pkparse.c \
67     src/pk.c \
68     src/pk_wrap.c \
69     src/rsa.c \
70     src/rsa_internal.c \
71     src/md_wrap.c \
72     src/bignum.c \
73     src/platform_util.c \

```



```

74      src/mbdttl_common.c \
75      src/mbdttl_crypto.c \
76      src/mbdttl_x509_parser.c
77
78 ##### BOOT33 SECBOOT END #####

```

- ❶ 如果使能安全检验功能，remove “#”，启用这一行
- ❷ 如果使能安全检验功能，remove “#”，启用这一行

## 7.4. 如何选配Boot33验签的image

默认加入签名，交由boot33进行验签的image包括CP.bin，对应CP\_IMAGE\_ID。

如果客户需要拓展验签image，需要在config目录中配json文件，如何配置如下所示：

以用户新增一个“customer\_app.bin”分区为例，通过如下json修改，即可在arelease过程中，对customer\_app.bin进行签名，并将证书信息录入fwcerts.bin中，在boot33运行阶段，便会对该image进行验签动作。



用户需要清楚板子是单flash还是双flash，单/双flash是使用不同的分区表json文件。如果不清楚板子是否有内置的8M的spi flash，可以看启动的log。

- 如果有“No embedded flash”表示没有内置flash，对应单flash的layout文件“config/partition/CRANE\_SINGLE\_FLASH\_LAYOUT.json”。
- 如果有“Has embedded flash”表示有内置flash，对应双flash的layout文件“config/partition/CRANE\_DUAL\_FLASH\_LAYOUT.json”。



用户新增customer\_app.bin的验签，下列json文件中需要增加这些有“+”标志的新增行。

### 7.4.1. 单flash分区表中增加新image

修改config/partition/CRANE\_SINGLE\_FLASH\_LAYOUT.json文件。增加customer\_app，其格式是raw，size=0表示根据文件的实际大小计算使用的空间。类同cp.bin的配置。

**config/partition/CRANE\_SINGLE\_FLASH\_LAYOUT.json.**

```
{
  "name": "CRANE_SINGLE_FLASH_LAYOUT",
  "partitions": [
    {
      "name": "external",
      "format": "flash",
      "vstart": "0x80000000",
      "partitions": [
        {
          "name": "bootloader",
          "format": "ubi",
          "size": "144KiB"
        },
        {
          "name": "system",
          "format": "part",
          "size": "0",
          "partitions": [
            {
              "name": "ptable",
              "format": "raw",
              "size": "4KiB"
            },
            {
              "name": "fwcerts",
              "format": "raw",
              "size": "12KiB"
            },
            {
              "name": "rd",
              "format": "raw",
              "size": "64KiB"
            },
            {
              "name": "apn",
              "format": "raw",
              "size": "32KiB"
            },
            {
              "name": "cp", ❶
              "format": "raw",
              "size": "0"
            },
          ],
        },
      ],
    },
  ],
}
```

```

        {
            "name": "dsp",
            "format": "raw",
            "size": "0"
        },
        {
            "name": "rfb",
            "format": "raw",
            "size": "0"
+       },
+       {
+           "name": "customer_app", ❷
+           "format": "raw",
+           "size": "0"
+       }
    ]
},
{

```

- ❶ 默认验签的image包括CP.bin，对应ID是字符串**CP\_IMAGE\_ID**，见于CRANE\_EVB.json
- ❷ 用户新增加的验签的image包括customer\_app.bin，对应ID是字符串**APP\_IMAGE\_ID**，需要在CRANE\_EVB.json中增加定义。

### 7.4.2. 双flash分区表中增加新image

修改config/partition/CRANE\_DUAL\_FLASH\_LAYOUT.json文件。在内置flash的分区描述中增加customer\_app。其配置类同dsp.bin。

**config/partition/CRANE\_DUAL\_FLASH\_LAYOUT.json.**

```

{
    "name": "CRANE_DUAL_FLASH_LAYOUT",
    "partitions": [
        {
            "name": "external",
            "format": "flash",
            "vstart": "0x80000000",
            "partitions": [
                {
                    "name": "bootloader",
                    "format": "ubi",
                    "size": "144KiB"
                },
            ],
        },
    ],
}

```

```
{
  "name": "system",
  "format": "part",
  "size": "0",
  "partitions": [
    {
      "name": "ptable",
      "format": "raw",
      "size": "4KiB"
    },
    {
      "name": "fwcerts",
      "format": "raw",
      "size": "12KiB"
    },
    {
      "name": "rd",
      "format": "raw",
      "size": "64KiB"
    },
    {
      "name": "apn",
      "format": "raw",
      "size": "32KiB"
    },
    {
      "name": "cp", ❶
      "format": "raw",
      "size": "0"
    }
  ]
},
```

...省略

```
{
  "name": "internal",
  "format": "flash",
  "vstart": "0x90000000",
  "partitions": [
    {
      "name": "system2",
      "format": "part",
      "size": "0",
      "partitions": [
        {
```

```

        "name": "dsp",
        "format": "raw",
        "size": "0"
    },
    {
        "name": "rfb",
        "format": "raw",
        "size": "0"
    },
+    {
+    {
+        "name": "customer_app", ❷
+        "format": "raw",
+        "size": "0"
+    }
    ]
},

```

### 7.4.3. fwcerts中增加新image

只有定义在crane\_evb.json文件中的fwcerts的fip段的images，才会arelease的打包过程中被签名，并证书存入fwcerts.bin。要支持customer\_app.bin被boot33验签，就需要添加其ID在fwcerts的fip段。

#### config/template/CRANE\_EVB.json.

```

{
    "name": "CRANE_EVB",
    "commands": [
        ...省略
        {
            "command": "flash",
            "name": "fwcerts",
            "type": "image",
            "partition": "fwcerts",
            "group": "2"
        },
        {
            "command": "flash",
            "name": "rd",
            "type": "image",
            "partition": "rd",
            "group": "2"
        },
    ],
    {

```

```

    "command": "flash",
    "name": "apn",
    "type": "image",
    "partition": "apn",
    "group": "2"
  },
  {
    "command": "flash",
    "name": "cp",
    "type": "image",
    "partition": "cp",
    "group": "2"
  },
  {
    "command": "flash",
    "name": "dsp",
    "type": "image",
    "partition": "dsp",
    "group": "2"
  },
+ {
+   "command": "flash",
+   "name": "customer_app",❶
+   "type": "image",
+   "partition": "customer_app",
+   "group": "2"
+ },
  {
    "command": "flash",
    "name": "rfbin",
    "type": "image",
    "partition": "rfbin",
    "group": "2"
  },
  ...省略

  {
    "name": "fwcerts",
    "image": "fwcerts.bin",
    "io": "out",
    "format": "fip",
    "fip": {
      "images": [
        {

```

```

        "id": "CP_IMAGE_ID",
        "image": "cp"
    },
    {
        "id": "DSP_IMAGE_ID",
        "image": "dsp"
+    },
+    {
+        "id": "APP_IMAGE_ID", ❷
+        "image": "customer_app"
    }
],
"callBoot2": false,
"certonly": true
}
},
...省略

```

- ❶ 增加commonid，烧写customer\_app.bin，类同cp.bin，都是image，都是group“2”。
- ❷ 要实现boot33对customer\_app.bin签名，并将证书信息录入fwcerts.bin中，需要增加id为**APP\_IMAGE\_ID**，对应的image是customer\_app.bin。

#### 7.4.4. boot33源码中增加新image

可选验签image项，在如下位置中配置ID与基本信息，如上操作已经用aboot工具或者arelease命令行工具配合预定义了UUID/OID，其中OID定义与aboot工具对镜像签名时使用的OID对应一致，在镜像签名时，OID包含在certification content中的x509 extension段，交由boot33配合进行验签。

boot33运行阶段，会对在CRANE\_EVB.json中配置入fwcert的fip段的images的ITEM都进行验签，未配置入fwcert的fip段的image，会skip掉。

如下调用中的ITEM\_xxx定义与同行注释中提及的CRANE\_EVB.json中使用xxx\_IMAGE\_ID是一一对应关系。

#### startup/bootloader/src/secboot.c.

```

891 #ifdef BOOT33_SECBOOT_SUPPORT
892     #include "secboot.h"
893     secboot_init();

```

```

894     Timer0_enable(TRUE);
895     /*
896     currenttly do not support image inside internal 8M spi_nor
flash
897     below items under secboot check , also open config to customer
898     refer to their owner fip image config inside of ABOOT
config.json
899     */
900     secboot_item_check(ITEM_CP);    //item CP_IMAGE_ID of fip .json
❶
901     secboot_item_check(ITEM_DSP);    //item DSP_IMAGE_ID of
fip .json ❷
902     secboot_item_check(ITEM_APP);    //item APP_IMAGE_ID of
fip .json ❸
903     secboot_item_check(ITEM_USER1); //item USER1_IMAGE_ID of
fip .json ❹
904     secboot_item_check(ITEM_USER2); //item USER2_IMAGE_ID of
fip .json❺
905     secboot_item_check(ITEM_USER3); //item USER3_IMAGE_ID of
fip .json❻
906     Timer0_enable(FALSE);
907 #endif

```

- ❶ ITEM\_CP对应**CP\_IMAGE\_ID** , **CP\_IMAGE\_ID**在CRANE\_EVB.json中已配置入fwcerts的fip段 , 被验签。
- ❷ ITEM\_DSP对应**DSP\_IMAGE\_ID** , **DSP\_IMAGE\_ID**在CRANE\_EVB.json中未配置入fwcerts的fip段 , 不被验签。
- ❸ ITEM\_APP对应**APP\_IMAGE\_ID** , **APP\_IMAGE\_ID**在CRANE\_EVB.json中如果客户新配置入fwcerts的fip段 , 被验签。
- ❹ ITEM\_USER1对应**USER1\_IMAGE\_ID** , **USER1\_IMAGE\_ID** 在CRANE\_EVB.json中未配置入fwcerts的fip段 , 不被验签。
- ❺ ITEM\_USER2对应**USER2\_IMAGE\_ID** , **USER2\_IMAGE\_ID** 在CRANE\_EVB.json中未配置入fwcerts的fip段 , 不被验签。
- ❻ ITEM\_USER3对应**USER3\_IMAGE\_ID** , **USER3\_IMAGE\_ID** 在CRANE\_EVB.json中未配置入fwcerts的fip段 , 不被验签。

如果客户自行添加APP\_IMAGE\_ID/DSP\_IMAGE\_ID/USER1\_IMAGE\_ID/USER2\_IMAGE\_ID/USER3\_IMAGE\_ID入fwcerts的fip段 , boot33运行阶段会验签新增image。



## 7.5. 打包下载包之前更新boot33

替换打包环境中的images/boot33.bin

**cmd console.**

```
G:\aboot-tools-2020.03.25-win-x64\images>dir
    2019.01.15 ❶
    2019.10.10 ❷
    apn.bin
    boot33.bin ❸
    cp.bin
    dsp.bin
    ReliableData.bin
    rf.bin
    updater.bin
```

- ❶ 用于存放与CRANE板子的bootrom相匹配的文件。例如：preboot.bin, flasher.bin, boot2.bin, fuse.bin和fuse.img。
- ❷ 用于存放与CRANEG板子的bootrom相匹配的文件。例如：preboot.bin, flasher.bin, boot2.bin, fuse.bin和fuse.img。
- ❸ 更新boot33.bin，适用于任何板子。

更新了boot33.bin以后，通过aboot带界面的工具或者arelease命令行方式做下载包ASR\_CRANE\_EVB\_CRANE\_A0\_16MB.zip。



---

# 8

## 固件下载

---

### 8.1. 下载

用aboot工具UI界面方式或者adownload工具命令行方式下载固件。

#### 8.1.1. UI界面方式

生成的下载包，用Aboot.exe的UI界面方式解析包的内容如下。默认是开发模式下，不勾选fuse.dat，也禁止勾选fuse.dat.所以要手动勾上production mode, 量产模式下才烧写fuse。

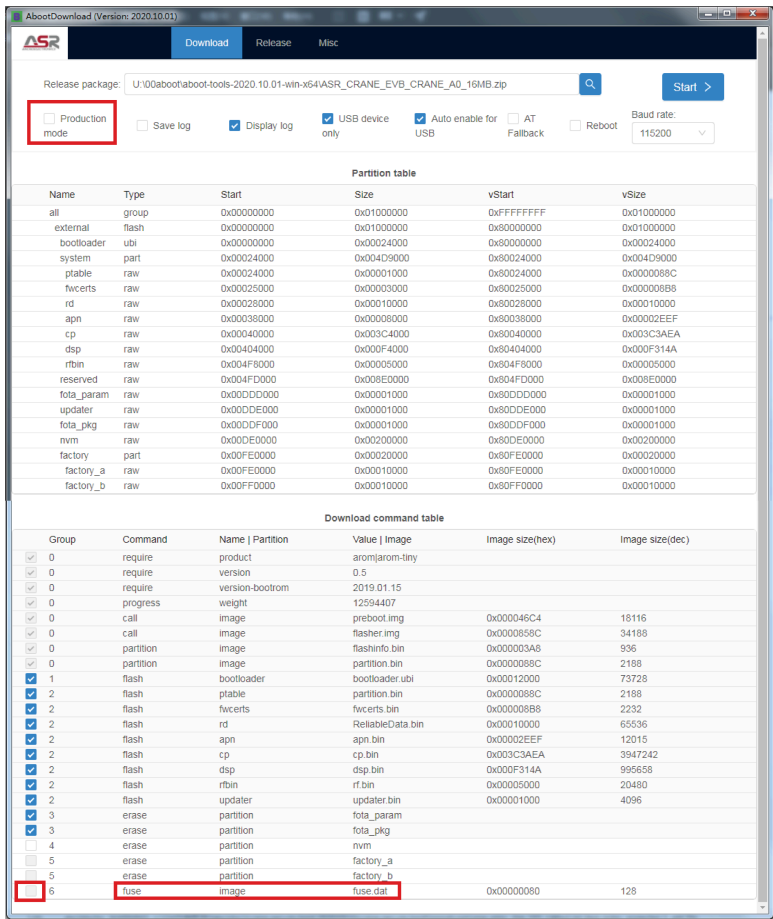


图 8.1. ui方式做下载包

- 勾选production mode量产模式

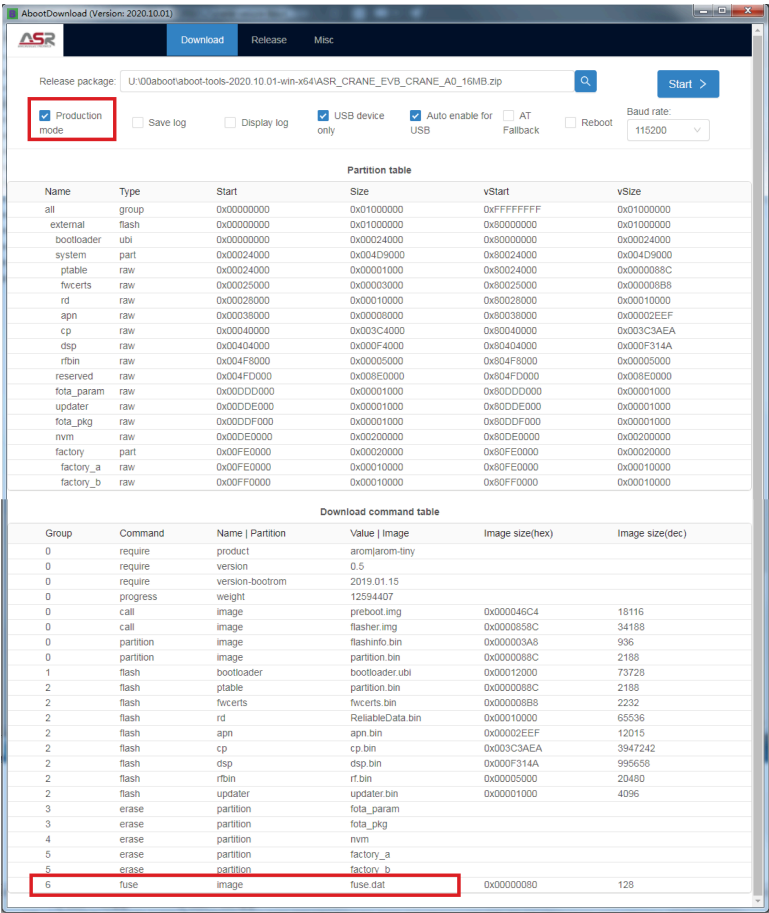


图 8.2. ui方式做下载包

8.1.2. cmd命令行方式

命令行输入以下命令，通过帮助信息找到下载的命令。

```
adownload.exe -h
```

cmd console.

```
U:\00aboot\aboot-tools-2020.10.01-win-x64>adownload.exe -h
Asrmicro about download console application.

Download release package FILE to boards.
Usage: adownload.exe [OPTION]... [FILE]
    -h, --help            Display this help and exit
```

```

-p, --port=port      Use named serial ports separate with comma
-a, --auto-enable    Or auto enable aram usb ports device
-u, --usb-only       Use aram usb ports only
-d, --dump-enable    Enable dump download protocol packet
-s, --speed=speed    Use given speed for serial communication
                    Supported baud rates:
                    (115200, 230400, 460800, 921600, 1842000,
                    3686400)
-m, --production     Running in mass prodcutin mode, default
is upgrade mode❶
-f, --at-fallback     Send AT cmd to fallback to download mode
-r, --reboot         Reboot device after finished
-q, --quit           Quit application after any port finished

```

#### Example:

```

adownload.exe -p COM1,COM2 -s 115200 aboot.zip
adownload.exe -u -a -s 115200 aboot.zip
adownload.exe -p COM1 -a -s 115200 aboot.zip

```

<1>-m 是进入量产模式的烧写，要烧写fuse，前提必须是量产模式的烧写，请加-m。adownload无-m，就是默认方式，也就是开发模式烧写，不烧写fuse。

命令行输入以下命令，下载固件zip包。

```
adownload.exe -u -a -q -m -s 115200 ASR_CRANE_EVB_CRANE_A0_16MB.zip
```

参数-a是自动检测到usb端口，就进入下载模式。

参数-u是只自动检测usb端口，不包括uart端口。

参数-q是下载结束以后退出程序。

参数-m是量产烧写模式



如果adownload命令没有用-q，下载结束后不会自动退出。如果用户又用Abboot.exe这种UI方式运行下载程序，会遇到执行操作错误提示，如果要用Abboot.exe，[请记得](#) adownload要退出。两种工具不能同时处于下载模式。



如果adownload命令没有用-m， fuse在开发模式下不会烧写。

adownload烧写固件的log如下：

### cmd console.

```
G:\aboot-tools-2020.10.01-win-x64>adownload.exe -u -a -q -m -s 115200
  ASR_CRANE_EVB_CRANE_A0_16MB.zip
parsing command line paramters ...
auto quit enabeld.
usb only enabeld.
usb port auto enabeld.
serial baud rate specified at "115200"
running in production mode.
release package file name specified is "ASR_CRANE_EVB_CRANE_A0_16MB.zip"
finished parsing command line paramters.
initializing aboot release package...
extracting download.json (2 KB)...
extracting partition.bin (2 KB)...
### ABOT_EVENT_DOWNLOAD_INIT ###
initialized aboot release package successfully.
Starting lwIP, IPV4 disable
ip6 linklocal address: FE80::200:EBFF:FE01:F654
starting aboot download engine...
download engine running in production mode!
extracting download.json (2 KB)...
### ABOT_EVENT_DOWNLOAD_START ###
aboot download engine started successfully.
getting serial devices list...
### ABOT_EVENT_DEVICE_NEW ###
{
    "displayName" : "ASR Modem Device (COM3)",
    "enabled" : false,
    "event" : 5,
    "locationInfo" : "Port_#0003.Hub_#0005",
    "order" : 0,
    "path" : "COM3",
    "productId" : 12311,
    "progress" : 0,
    "status" : "ONLINE",
    "vendorId" : 11980
}
### ABOT_EVENT_DEVICE_CHANGE ###
{
    "displayName" : "ASR Modem Device (COM3)",
    "enabled" : true,
    "event" : 6,
    "locationInfo" : "Port_#0003.Hub_#0005",
```

```
    "order" : 1,
    "path" : "COM3",
    "productId" : 12311,
    "progress" : 0,
    "status" : "ONLINE",
    "triggered" : false,
    "vendorId" : 11980
}
### ABOUT_EVENT_DEVICE_CHANGE ###
{
    "displayName" : "ASR Modem Device (COM3)",
    "enabled" : true,
    "event" : 6,
    "locationInfo" : "Port_#0003.Hub_#0005",
    "order" : 1,
    "path" : "COM3",
    "productId" : 12311,
    "progress" : 0,
    "status" : "CONNECTING",
    "triggered" : false,
    "vendorId" : 11980
}
<COM3> new device arrived.
enabling device <COM3> into downloading mode...
device <COM3> enabled successfully.
connecting to serial device <COM3>...
connected to serial device <COM3> successfully!
starting to fire device <COM3>...
device <COM3> fired successfully.
### ABOUT_EVENT_DEVICE_CHANGE ###
{
    "displayName" : "ASR Modem Device (COM3)",
    "enabled" : true,
    "event" : 6,
    "locationInfo" : "Port_#0003.Hub_#0005",
    "order" : 1,
    "path" : "COM3",
    "productId" : 12311,
    "progress" : 0,
    "status" : "RUNNING",
    "triggered" : false,
    "vendorId" : 11980
}
<COM3> processing command [get bootrom info]
<COM3> processing command [require product arom]
<COM3> checking product
```



```

<COM3> OKAY [ 0.011s]
<COM3> processing command [require version 0.5]
<COM3> Checking version
<COM3> OKAY [ 0.013s]
<COM3> processing command [require version-bootrom 2019.01.15]
<COM3> Checking version-bootrom
<COM3> OKAY [ 0.014s]
<COM3> processing command [progress weight 12618851]
<COM3> setting total progress
<COM3> OKAY [ 0.013s]
<COM3> target reported max download size of 64768 bytes
<COM3> extracting preboot.img (16 KB)...
<COM3> processing command [call preboot.img]
<COM3> Sending 'preboot.img' (16 KB)
<COM3> OKAY [ 0.145s]
<COM3> verifying①
<COM3> OKAY [ 0.219s]
<COM3> calling#
<COM3> #=> Executing preboot application...
<COM3> #=> fuse: sbe = 1, sys_boot_ctrl=0x148020
<COM3> #=> ### Trusted boot mode. ##②
<COM3> #=> Found PMIC with Id: 0x3b
<COM3> #=> Disabling PMIC watchdog
<COM3> #=> PMIC watchdog is disabled
<COM3> #=> power_up_reason=0x40.
<COM3> #=> Disable PMIC fault wakeup.
<COM3> #=> CRANE: set PM813_DVC_SET_REG = 0x0, set bit7 to 0.
<COM3> #=> CRANE: set cpu core voltage PM813_BUCK_VOLTAGE_SET_REG = 0xa4
<COM3> #=> Freq change done: CR5 416MHZ -> 624MHZ, AXI 156MHZ -> 208MHZ
<COM3> #=> PMU_MAIN_CRSR(0xd4050028) = 0x3
<COM3> #=> PSRAM PHY frequency changed to 78
<COM3> #=> [PSRAM] psram_init_crane_a0.③
<COM3> #=> #####
<COM3> #=> Version ID : 0
<COM3> #=> AP 16MB④
<COM3> #=> No embedded flash.⑤
<COM3> #=> #####
<COM3> #=> #####
<COM3> #=> MP DEVICE
<COM3> #=> MR[2] VALUE = 0x5
<COM3> #=> #####
<COM3> #=> psram cache enabled !!! :      cache line=[192B]
<COM3> #=> Do Global Reset
<COM3> #=> Enable fast_miss_acc

```

```
<COM3> #=> PSRAM PHY frequency changed to 175
<COM3> #=> set fip image load address spaces to 0x7e000000
<COM3> OKAY [ 0.088s]
<COM3> target reported max download size of 64768 bytes
<COM3> extracting flasher.img (30 KB)...
<COM3> processing command [call flasher.img]
<COM3> Sending 'flasher.img' (30 KB)
<COM3> OKAY [ 0.178s]
<COM3> verifying⑥
<COM3> OKAY [ 0.157s]
<COM3> calling
<COM3> #=> Executing flasher application...
<COM3> OKAY [ 0.009s]
<COM3> target reported max download size of 2097152 bytes
<COM3> extracting flashinfo.bin (720 Bytes)...
<COM3> processing command [partition flashinfo.bin]
<COM3> Sending 'flashinfo.bin' (0 KB)
<COM3> OKAY [ 0.020s]
<COM3> setting partition
<COM3> #=> QSPI clock configured at 104 MHz
<COM3> #=> Manufacturer ID: 0xC8, Device ID: 0x6018
<COM3> #=> Found a known spi flash device "GD25LQ128D"⑦
<COM3> OKAY [ 0.013s]
<COM3> target reported max download size of 2097152 bytes
<COM3> extracting partition.bin (2 KB)...
<COM3> processing command [partition partition.bin]⑧
<COM3> Sending 'partition.bin' (2 KB)
<COM3> OKAY [ 0.028s]
<COM3> setting partition
<COM3> #=> ptn 0 name='all' start=00000000 size=01000000 type=group
depth=0
<COM3> #=> ptn 1 name='external' start=00000000 size=01000000 type=flash
depth=1
<COM3> #=> ptn 2 name='bootloader' start=00000000 size=00024000 type=ubi
depth=2
<COM3> #=> ptn 3 name='system' start=00024000 size=004d9000 type=part
depth=2
<COM3> #=> ptn 4 name='ptable' start=00024000 size=00001000 type=raw
depth=3
<COM3> #=> ptn 5 name='fwcerts' start=00025000 size=00003000 type=raw
depth=3
<COM3> #=> ptn 6 name='rd' start=00028000 size=00010000 type=raw depth=3
<COM3> #=> ptn 7 name='apn' start=000380a0 size=00008000 type=raw
depth=3
<COM3> #=> ptn 8 name='cp' start=00040000 size=003c4000 type=raw depth=3
```

```
<COM3> #=> ptn 9 name='dsp' start=00404000 size=000f4000 type=raw
depth=3
<COM3> #=> ptn 10 name='rfbin' start=004f8000 size=00005000 type=raw
depth=3
<COM3> #=> ptn 11 name='reserved' start=004fd000 size=008e0000 type=raw
depth=2
<COM3> #=> ptn 12 name='fota_param' start=00ddd000 size=00001000
type=raw depth=2
<COM3> #=> ptn 13 name='updater' start=00dde000 size=00001000 type=raw
depth=2
<COM3> #=> ptn 14 name='fota_pkg' start=00ddf000 size=00001000 type=raw
depth=2
<COM3> #=> ptn 15 name='nvm' start=00de0000 size=00200000 type=raw
depth=2
<COM3> #=> ptn 16 name='factory_a' start=00fe0000 size=00010000 type=raw
depth=2
<COM3> #=> ptn 17 name='factory_b' start=00ff0000 size=00010000 type=raw
depth=2
<COM3> (flasher) parse partition table successfully.
<COM3> OKAY [ 0.072s]
<COM3> target reported max download size of 2097152 bytes
<COM3> extracting bootloader.ubi (96 KB)...
<COM3> processing command [flash bootloader bootloader.ubi]⑨
<COM3> Erasing 'bootloader'...
<COM3> (flasher) erasing 'bootloader' scheduled in background
<COM3> OKAY [ 0.012s]
<COM3> Sending sparse 'bootloader' 1/1 (96 KB)...
<COM3> OKAY [ 0.018s]
<COM3> writing 'bootloader' 1/1...
<COM3> (flasher) writing 'bootloader' scheduled in background
<COM3> OKAY [ 0.013s]
<COM3> target reported max download size of 2097152 bytes
<COM3> extracting partition.bin (2 KB)...
<COM3> processing command [flash ptable partition.bin]
<COM3> Sending sparse 'ptable' 1/1 (4 KB)...
<COM3> OKAY [ 0.014s]
<COM3> writing 'ptable' 1/1...
<COM3> (flasher) writing 'ptable' scheduled in background
<COM3> OKAY [ 0.012s]
<COM3> target reported max download size of 2097152 bytes
<COM3> extracting fwcerts.bin (2 KB)...
<COM3> processing command [flash fwcerts fwcerts.bin]
<COM3> Sending sparse 'fwcerts' 1/1 (4 KB)...
<COM3> OKAY [ 0.014s]
<COM3> writing 'fwcerts' 1/1...
```

```

<COM3> (flasher) writing 'fwcerts' scheduled in background
<COM3> OKAY [ 0.012s]
<COM3> target reported max download size of 2097152 bytes
<COM3> extracting ReliableData.bin (64 KB)...
<COM3> processing command [flash rd ReliableData.bin]
<COM3> Sending sparse 'rd' 1/1 (4 KB)...
<COM3> OKAY [ 0.014s]
<COM3> Writing 'rd' 1/1...
<COM3> (flasher) writing 'rd' scheduled in background
<COM3> OKAY [ 0.011s]
<COM3> target reported max download size of 2097152 bytes
<COM3> extracting apn.bin (11 KB)...
<COM3> processing command [flash apn apn.bin]

```

省略....

```

<COM3> #=>      pid | name                | state   Q | pri | stack
( used) | base addr | current
<COM3> #=>      - | isr_stack          | -       - | - | 1024 (
772) | 0xd100f9c0 | 0xffffffff
<COM3> #=>      1 | idle              | pending Q | 15 | 160 (
128) | 0xd1002a10 | 0xd1002a30
<COM3> #=>      2 | main              | b1 mbox _ | 7 | 2560 (
524) | 0xd1002ab0 | 0xd10032c0
<COM3> #=>      3 | ipv6              | b1 rx   _ | 4 | 1024 (
452) | 0xd1007110 | 0xd10073f0
<COM3> #=>      4 | udp               | b1 rx   _ | 5 | 1024 (
292) | 0xd10078a4 | 0xd1007ba8
<COM3> #=>      5 | heartbeat         | b1 rx   _ | 6 | 512 (
196) | 0xd10034d4 | 0xd1003628
<COM3> (flasher) partition "factory_b" erased successfully.
<COM3> #=>      6 | ethos             | b1 rx   _ | 2 | 1024 (
416) | 0xd10025c4 | 0xd10028cc
<COM3> #=>      7 | aboot             | b1 rx   _ | 8 | 2560
( 1680) | 0xd100607c | 0xd10069d0
<COM3> #=>      8 | flasher           | running Q | 9 | 1536 (
924) | 0xd1006b08 | 0xd1006f58
<COM3> #=>      | SUM              |         |   | 11424
( 5384)
<COM3> OKAY [ 8.293s]
<COM3> all finished. total time: 25.197s10
### ABOOT_EVENT_DEVICE_CHANGE ###
{
    "displayName" : "ASR Modem Device (COM3)",
    "enabled" : true,
    "event" : 6,

```

```

        "locationInfo" : "Port_#0003.Hub_#0005",
        "order" : 1,
        "path" : "COM3",
        "productId" : 12311,
        "progress" : 100,
        "status" : "SUCCEEDED",
        "triggered" : false,
        "vendorId" : 11980
    }
    stopping about download engine...
    ### ABOT_EVENT_DOWNLOAD_STOP ###

```

- ❶ preboot.img被数字签名验证。
- ❷ preboot.img签名验证通过，preboot的执行，打印出Trusted boot，说明fuse已经烧写过。
- ❸ preboot初始化CRANE\_A0的psram。
- ❹ PSRAM的型号AP 16M，这个bit是从fuse读出的。
- ❺ 无内置的flash，这个bit是从fuse读出的。
- ❻ 签名验证flasher.img。
- ❼ flasher.img验证通过，初始成功nor flash GD25LQ128D。
- ❽ flasher程序打印分区表。
- ❾ 烧写bootloader.ubi。下面省略的log有继续烧写rd，cp和dsp等固件。
- ❿ 所有固件烧写完毕。



烧写各个镜像文件，如果签名验证不通过。都会打印出错log，终止烧写过程。

## 8.2. 重启

烧写完固件以后，必须重启模块，才能进入Trusted Boot。下面两种方法可以重启：

- 按reset键，板子重启。
- 拔掉usb线，板子重启。

### 8.2.1. 不使能boot33验签功能，启动的log

重启模块后，Trusted Boot的串口log如下：

## sscom窗口.

```
[16:56:23.015]收←◆CHIP_ID: 0x6731, REV_ID: 0xA0
PLATFORM: CRANE (SILICON)
CRANE hardware initialization complete.

AROM! (Version: 2019.01.15)
Initializing crypto library...
Loading...
Welcome to the trusted boot rom world.
QSPI clock configured at 13 MHz
Detecting QSPI flash devices...
Found a known spi flash device "GD25LQ128D"
SPI nor flash: Manufacturer ID: 0xC8, Device ID: 0x6018
Detected peb size is 4096, the first good peb number is 0
Found preboot volume, size is 16340 bytes
BL1: Trying to load and verify BL2 image from volume preboot...❶

[16:56:23.262]收←◆BL1: Okay.
Executing preboot application...
fuse: sbe = 1, sys_boot_ctrl=0x158020
### Trusted boot mode. ##❷
Found PMIC with Id: 0x3b
Disabling PMIC watchdog
PMIC watchdog is disabled
power_up_reason=0x20.
Disable PMIC fault wakeup.
CRANE set PM813_DVC_SET_REG = 0x0, set bit7 to 0.
CRANE: set cpu core voltage PM813_BUCK_VOLTAGE_SET_EG = 0xa4
get restart cmd = 0x0
PMU_SD_ROT_WAKE_CLR(0xd428287c) = 0x2000

[16:56:23.409]收←◆set restart status = 0x0
Freq change done: CR5 416MHz -> 624MHz, AXI 156MHz -> 208MHz
PMU_MAIN_CRSR(0xd4050028) = 0x3
PSRAM PHY frequency changed to 78
[PSRAM] psram_init_crane_a0.
#####
Version ID : 0
AP 16MB
No embedded flash.
#####
#####
MP DEVICE
MR[2] VALUE = 0x5
```

```
#####
psram cache enabled !!! : cache line=[192B]
Do Global Reset
Enable fast_miss_acc
PSRAM PHY frequency changed to 175
set fip image load address spaces to 0x7e000000
Found bootloader volume, size is 58120 bytes
BL1: Trying to load and verify BL2 image from volume bootloader...③

[16:56:23.607]收←♦BL1: OKay.
Executing boot2 application...④
QSPI clock configured at 104 MHz
Manufacturer ID: 0xC8, Device ID: 0x6018
Found a known spi flash device "GD25LQ128D"
ptn 0 name='all' start=00000000 size=01000000 type=group depth=0
ptn 1 name='external' start=00000000 size=01000000 type=flash depth=1
ptn 2 name='bootloader' start=00000000 size=00024000 type=ubi depth=2
ptn 3 name='system' start=00024000 size=004d9000 type=part depth=2
ptn 4 name='ptable' start=00024000 size=00001000 type=raw depth=3
ptn 5 name='fw
[16:56:23.652]收←♦certs' start=00025000 size=00003000 type=raw depth=3
ptn 6 name='rd' start=00028000 size=00010000 type=raw depth=3
ptn 7 name='apn' start=00038000 size=00008000 type=raw depth=3
ptn 8 name='cp' start=00040000 size=003c4000 type=raw depth=3
ptn 9 name='dsp' start=00404000 size=000f4000 type=raw depth=3
ptn 10 name='rfsbin' start=004f8000 size=00005000 type=raw depth=3
ptn 11 name='reserved' start=004fd000 size=008e0000 type=raw depth=2
ptn 12 name='fota_param' start=00ddd000 size=00001000 type=raw depth=2
ptn
[16:56:23.698]收←♦ 13 name='updater' start=00dde000 size=00001000
type=raw depth=2
ptn 14 name='fota_pkg' start=00ddf000 size=00001000 type=raw depth=2
ptn 15 name='nv' start=00de0000 size=00200000 type=raw depth=2
ptn 16 name='factory_a' start=00fe0000 size=00010000 type=raw depth=2
ptn 17 name='factory_b' start=00ff0000 size=00010000 type=raw depth=2
I will use MPU 2
0x01,0x00,0x64,0xAA,0x78,0x56,0x34,0x12,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x77,0xC4,0xF4,0x76,0x1B,0x94,0x41,0x7C,
0xB3,0x43,0xB0,0xEA,0x44,0x3A,0xAC
[16:56:23.743]收←♦,0xD7,
0x00,0xB0,0x01,0x00,0x00,0x00,0x00,0x00,
0xEA,0x3A,0x3C,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x8A,0xBD,0x24,0xEB,0xE0,0x69,0x4A,0xCD,
```

```

0x88,0x15,0x30,0xA9,0x50,0xB9,0x3A,0x2B,
0x00,0xF0,0x3D,0x00,0x00,0x00,0x00,0x00,
0x4A,0x31,0x0F,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x26,0x31,0x02,0xFB,0xF9,0x8E,0x42,0xF8,
0x98,0xDF,0xB0,0xE1,0xD3,0x51,0x8E,0x54,
0xD8,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0xED,0x03,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x0
[16:56:23.788] 收←♦0,0x00,0x00,0x00,0x00,0x00,
0x2B,0xCA,0xE7,0xA1,0x39,0x0F,0x43,0xD2,
0x82,0x58,0x45,0xD5,0x63,0xE0,0x0D,0x57,
0xC8,0x04,0x00,0x00,0x00,0x00,0x00,0x00,
0xEF,0x03,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0xB8,0x08,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x30,0x82,0x03,0xE9,0x30,0x82,0x02,0xA1,
0xA0,0x03,0x02,0x01,0x02,0x02,0x0
[16:56:23.833] 收←♦9,0x00,
0xEF,0x11,0xB9,0x7A,0xF2,0x98,0xA3,0x2F,
0x30,0x3D,0x06,0x09,0x2A,0x86,0x48,0x86,
0xF7,0x0D,0x01,0x01,0x0A,0x30,0x30,0xA0,
PSRAM PHY frequency changed to 78
I will use MPU 3
Freq change done: CR5 624MHz -> 416MHz, AXI 208MHz -> 156MHz
boot to 0X7E000000, cpsr should be 0x1d3
pid | name | state | Q | pri | stack (used) | base
addr | current
- | isr_stack | - | - | - | 1024 ( 144) |
0xd100f9c0 | 0xffffffff
1 | idle | pending | Q | 15
[16:56:23.866] 收←♦ | 160 ( 128) | 0xd1002a10 | 0xd1002a30
2 | main | running | Q | 7 | 2560 ( 1628) |
0xd1002ab0 | 0xff00
3 | ipv6 | b1 rx | _ | 4 | 1024 ( 288) |
0xd1007110 | 0xd10073f0
4 | udp | b1 rx | _ | 5 | 1024 ( 252) |
0xd10078a4 | 0xd1007ba8
| SUM | | | | 5792 ( 2440)

[16:56:23.919] 收←♦Boot33 UART INIT DONE⑤

```



```

Built by DS-5.
boot33 MPU config
=====DUMP MPU REGION=====
[ NUM ] [ BASE_ADDR ] [ ACCES_CTRL ] [ REG_SIZE ]
[ 0 ] [ 0x00000000 ] [ 0x00000308 ] [ 0x0000001e ]
[ 1 ] [ 0xd1000000 ] [ 0x00000308 ] [ 0x00000013 ]
[ 2 ] [ 0x80000000 ] [ 0x00000308 ] [ 0x0000001a ]
[ 3 ] [ 0x80000000 ] [ 0x0000030b ] [ 0x00000019 ]
[ 4 ] [ 0xc0000000 ] [ 0x00001301 ] [ 0x0000001d ]
[ 5 ] [ 0x00000000 ] [ 0x00000000 ] [ 0x00000000 ]
[ 6 ] [ 0x00000000 ] [ 0x00000000 ] [ 0x00000000 ]
[ 7 ] [ 0x00000000 ] [ 0x00000000 ] [ 0x00000000 ]
Func:Keypad_OTA_Flag_Check_B,Line:832
[-----Keypad_OTA_Flag_Check_B-----]
Keypad_init_clk
Keypad_init_clk done

[16:56:24.152]收←◆[2_2]After polling 1000 times, not get a key value
[OTA] NO FOTA FLAG DETECTED
[-----]
[LOADTABLE AREA ] [LOADTABLE_AREA_HEADER ]
[INIT_ROUTINE ] [7e6d2104 ]
[-----]
[LOADTABLE AREA ] [LOADTABLE_AREA_VER_INFO ]
[ANTI_RB_VER ] [20190904 ]
[EXECUTE_MODE ] [XIP ]
[PS
[16:56:24.197]收←◆_MODE ] [LTEGSM
]
[IMAGE_DESC ] [CRANE_DS_LTEGSM_DKB_Z2_A0_XIP_LWIP_MODULEONLY_16M16M]
[-----]
[LOADTABLE AREA ] [LOADTABLE_AREA_RW_CPZ_INFO ]
[RW_CPZ_1] [DDR_RW_] [7ea00000] [80051508] [00023a10] [80051508]
[RW_CPZ_3] [PS_NCAH] [7ef20000] [80074f18] [000052c0] [80058d96]
[RW_CPZ_4] [USBNCAH] [7ef92400] [8007a1d8] [00000244] [8005928c]
[RW_CPZ_5] [CODE_PS] [7e200000] [8007a41c] [0054320c] [800593ed]
[-----]
[16:56:24.242]收←◆-----]
[LOADTABLE AREA ] [LOADTABLE_AREA_ARMLINK_SYMBOL ]
[00.CP_EXEC ] [0x80040000] [AVAILABLE ]
[01.CP_LOAD ] [0x4c4c554e] [INVALID ]
[02.IMG_END ] [0x4c4c554e] [INVALID ]
[03.BIN_SIZE ] [0x003c3aea] [AVAILABLE ]
[04.DSP_BEGIN ] [0x7e000000] [AVAILABLE ]
[05.DSP_END ] [0x7e1ffffc] [AVAILABLE ]
[06.RF_ADDR_Z2 ] [0x7e090000] [AVAILABLE ]

```

```

[16:56:24.287]收←♦          ]
[07.RF_ADDR_A0  ][0x7e090000][AVAILABLE          ]
[08.RD_BEGIN    ][0x80028000][AVAILABLE          ]
[09.RD_END      ][0x80038000][AVAILABLE          ]
[10.APN_BEGIN   ][0x80038000][AVAILABLE          ]
[11.APN_END     ][0x80040000][AVAILABLE          ]
[12.FOTA_PRMS   ][0x80dbf000][AVAILABLE          ]
[13.FOTA_PRME   ][0x80dc0000][AVAILABLE          ]
[14.UPDATER_S   ][0x80dc0000][AVAILABLE          ]
[1
[16:56:24.333]收←♦5.UPDATER_E  ][0x80de0000][AVAILABLE
]
[16.FOTA_PKG_S  ][0x80de0000][AVAILABLE          ]
[17.FOTA_PKG_E  ][0x80de0000][AVAILABLE          ]
[18.NVM_BEGIN   ][0x80de0000][AVAILABLE          ]
[19.NVM_END     ][0x80fe0000][AVAILABLE          ]
[20.F_A_START   ][0x80fe0000][AVAILABLE          ]
[21.F_A_END     ][0x80ff0000][AVAILABLE          ]
[22.F_B_START   ][0x80ff0000][AVAILABLE          ]
[23.F_B_END     ][0x810000
[16:56:24.378]收←♦00][AVAILABLE          ]
[-----]
[LOADTABLE AREA ][LOADTABLE_AREA_FUNC_VAL          ]
[00.VERGIN      ][ffffffff          ]
[01.BOOTCOUNT  ][ffffffff          ]
[02.UART_PRT    ][1          ]
[03.FATAL_PRT   ][1          ]
[04.DEF_FREQ    ][ffffffff          ]
[05.CORE_VOL    ][ffffffff          ]
[-----]
BOOT33_VERSION : 20190904[20190904]
CP_ANTI_RB_VER : 20190904[20190904]
[CP ] CHIP_ID=[00a06731] CRANE_A0
[CP ] Region CPZ struct detected from loadtable
[CP ] decompress [ DDR_RW_] from [80051508] to [7ea00000]

[16:56:24.449]收←♦[CP ] decompress [ PS_NCAH] from [80058d96] to
[7ef20000]
[CP ] decompress [ USBNCAH] from [8005928c] to [7ef92400]
[CP ] decompress [ CODE_PS] from [800593ed] to [7e200000]

[16:56:26.791]收←♦[CP ] stop decompress as no further RW_CPZ_ detected
[DSP] SKIP DSP LOAD
[RF ] SKIP RF LOAD

*****

```

```

** BOOTLOADER DONE JUMP TO CP IMAGE⑥
** VERSION : 20190904
** PC      : 0x80040000
*****

```

[16:56:26.873] 收←◆Cinit 1

- ① preboot卷签名验证中。
- ② preboot签名验证通过，preboot检测到Trusted Boot。
- ③ bootloader卷签名验证中。
- ④ bootloader卷签名验证通过，boot2执行。
- ⑤ boot33执行。
- ⑥ CP执行。



启动各个镜像文件，如果签名验证不通过。都会打印出错log，终止启动。

### 8.2.2. 使能boot33验签功能，启动时boot33的log

成功运行的开机log如下，会打印被验证目标image，验证结果与耗时，验证具体消耗时间与被验证image的size相关。

**sscom**窗口。

```

//boot33开启了secboot验签功能
[20200402-16:25:01.987][BOOT33]VB_VERSION_DATE           :
[20200320]
[20200402-16:25:01.987][BOOT33]VB_OEM_LCD_TYPE           :
[NO_LCD]
[20200402-16:25:01.987][BOOT33]VB_SECBOOT_SUPPORT        :
[SECBOOT]
[20200402-16:25:01.988][BOOT33]VB_COMPRESSED_LOGO_SUPPORT :
[SUPPORT_COMPRESSED_LOGO]
[20200402-16:25:01.988][BOOT33]build_info_string         :
[boot33.bin 2020320_ver1]
[20200308-13:31:26.380][SECBOOT]fh->entry_count = [4]

//对CP.bin进行安全校验，耗时5s
[20200402-16:25:03.287][SECBOOT]fh->entry_count = [6]
[20200402-16:25:03.287][SECBOOT]sb_item item_id = [0] check

```

```
[20200402-16:25:03.287][SECB00T]sb_item [0][CP][31][30]
[1.3.6.1.4.1.4128.2100.8001]
[20200402-16:25:03.287][SECB00T]bingo entry_index = [3]
[20200402-16:25:03.295][SECB00T]bingo entry_index = [0]
[20200402-16:25:03.295][SECB00T][CP]cert check_integrity start
[20200402-16:25:03.295][SECB00T][CP]cert check_integrity done rc = 0
[20200402-16:25:03.350][SECB00T][CP]img_content verify_hash start
[20200402-16:25:09.502][SECB00T][CP]img_content verify_hash done rc = 0
[20200402-16:25:09.502][SECB00T][CP]secboot check done SUCCEED
[20200402-16:25:09.502][SECB00T][CP]secboot time consumption = 5202 ms

//对DSP.bin进行安全校验耗时2s
[20200402-16:25:09.502][SECB00T]sb_item item_id = [1] check
[20200402-16:25:09.502][SECB00T]sb_item [1][DSP][33][32]
[1.3.6.1.4.1.4128.2100.8002]
[20200402-16:25:09.503][SECB00T]bingo entry_index = [4]
[20200402-16:25:09.503][SECB00T]bingo entry_index = [1]
[20200402-16:25:09.503][SECB00T][DSP]cert check_integrity start
[20200402-16:25:09.507][SECB00T][DSP]cert check_integrity done rc = 0
[20200402-16:25:09.564][SECB00T][DSP]img_content verify_hash start
[20200402-16:25:11.141][SECB00T][DSP]img_content verify_hash done rc = 0
[20200402-16:25:11.141][SECB00T][DSP]secboot check done SUCCEED
[20200402-16:25:11.141][SECB00T][DSP]secboot time consumption = 1632 ms

//对customer_app.bin安全校验耗时121ms
[20200402-16:25:11.141][SECB00T]sb_item item_id = [2] check
[20200402-16:25:11.142][SECB00T]sb_item [2][APP][35][34]
[1.3.6.1.4.1.4128.2100.8003]
[20200402-16:25:11.142][SECB00T]bingo entry_index = [5]
[20200402-16:25:11.142][SECB00T]bingo entry_index = [2]
[20200402-16:25:11.142][SECB00T][APP]cert check_integrity start
[20200402-16:25:11.146][SECB00T][APP]cert check_integrity done rc = 0
[20200402-16:25:11.204][SECB00T][APP]img_content verify_hash start
[20200402-16:25:11.283][SECB00T][APP]img_content verify_hash done rc = 0
[20200402-16:25:11.283][SECB00T][APP]secboot check done SUCCEED
[20200402-16:25:11.283][SECB00T][APP]secboot time consumption = 121 ms

//skip没有配置在fwcert fip中的item，boot33阶段fwcert.bin的安全已经由前置运行保证
[20200402-16:25:11.284][SECB00T]sb_item item_id = [3] check
[20200402-16:25:11.284][SECB00T]sb_item [3][USER1][37][36]
[1.3.6.1.4.1.4128.2100.8101]
[20200402-16:25:11.284][SECB00T]fip doesn't contain image(id=37)
[20200402-16:25:11.284][SECB00T]fip_open_image [37] with
    FIP_ERR_IMG_MISS, just skip

[20200402-16:25:11.284][SECB00T]sb_item item_id = [4] check
```

```
[20200402-16:25:11.285][SECB00T]sb_item [4][USER2][39][38]
[1.3.6.1.4.1.4128.2100.8102]
[20200402-16:25:11.285][SECB00T]fip doesn't contain image(id=39)
[20200402-16:25:11.327][SECB00T]fip_open_image [39] with
  FIP_ERR_IMG_MISS, just skip

[20200402-16:25:11.327][SECB00T]sb_item item_id = [5] check
[20200402-16:25:11.327][SECB00T]sb_item [5][USER3][41][40]
[1.3.6.1.4.1.4128.2100.8103]
[20200402-16:25:11.328][SECB00T]fip doesn't contain image(id=41)
[20200402-16:25:11.328][SECB00T]fip_open_image
```



## 工厂双工位烧写

---

Secboot使能步骤在工厂模式的烧写环境中，为加快烧写速度可以采用双工位（station）烧写。

- 工位1只负责烧写fuse。
- 工位2只负责烧写固件包。

流水线同时操作。工位1烧好的芯片做为工位2的烧写芯片。

### 9.1. 为两个工位准备一套版本

1. 生成私钥和证书。
2. 制作烧写OEM public key的fuse only的下载包A。
3. 制作烧写镜像签名的下载包B。

### 9.2. 工位1烧写fuse only下载包A

1. 插上没有烧过fuse的芯片，上电烧写fuse only下载包A。Trusted模式需要的Root key和OEM公钥hash被烧录进Fuse。
2. 下电取出芯片给工位2。

### 9.3. 工位2烧写镜像签名的下载包B

1. 插上从工位1拿来的烧过fuse的芯片，上电烧写镜像签名的下载包B。
2. 固件烧录，烧录过程中自动对固件进行数字签名。
3. 下电完成。

