

About Tools

ASRMicro, Inc.

Revision 2.0

2020/09/28

Table of Contents

1. Introduction	1
1.1. Tool for linux environment	2
1.2. Tool for windows environment	3
1.2.1. windows 64 bits env	3
1.2.2. windows 32 bits env	3
2. DKB board	5
2.1. Install required USB ACM or USB-to-Serial converter drivers	6
2.1.1. USB driver	6
2.1.2. UART driver	7
2.2. start Putty to monitor UART trace	8
3. Prepare generate release package	13
3.1. The information of product	16
3.2. The information of flash	21
3.3. The information of images	22
3.4. Binary files introduction:	22
4. Generate release package	27
4.1. Arelease.exe	27
4.2. Aboot.exe	31
4.3. How to change custom images	33
5. Burn program mode	37
5.1. Enter burn program mode normally	37
5.2. Enter burn program mode via XDB(crane Z1)	37
5.2.1. Burn Steps via XDB	37
6. Burn package to target	43
6.1. Download release package by USB	43
6.2. Adownload.exe	43
6.3. Aboot.exe	44
6.4. Download release package by UART	53
6.5. Trace for burn program	56
6.6. Trace for boot	63
7. upload function	67
7.1. Generate upload package	67
7.1.1. Arelease.exe	67
7.1.2. Adownload.exe	70

7.2. Burn upload package	72
7.2.1. Adownload.exe	72
7.2.2. Aboot.exe	73
8. erase all function	77
8.1. Generate erase all package	77
8.1.1. Adownload.exe	77
8.1.2. Aboot.exe	80
8.2. Burn erase all package	81
8.2.1. Adownload.exe	81
8.2.2. Aboot.exe	82
9. Boot in production mode	85
9.1. Enter production mode	85
9.2. Exit production mode	89

1

Introduction

The ASR Aboot Tools enables you to and generate release packages for products and download images to the target.

Aboot Tools are included as below :

1. arelease.exe
2. adownload.exe

The stable tools are shared at

```
\\sh2-filer02\Release\LTE\Crane\Tools.
```

Currently the latest version is **2020-09-22**.

win-x64 : used for windows 64bits.

win-x86 : used for windows 32bits.

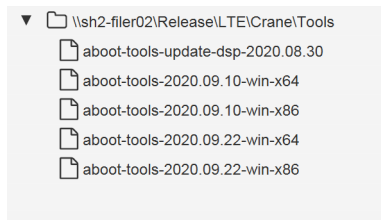


Figure 1.1. Aboot tools folder filesystem Tree

Latest Aboot Tool can be downloaded from jenkins link.








Build Artifacts		
	aboot-tools-2020.09.25-linux-x64.sfx	157.91 MB  view
	aboot-tools-2020.09.25-win-x64.exe	143.42 MB  view
	aboot-tools-2020.09.25-win-x86.exe	140.81 MB  view

Figure 1.2. aboot-tools-2020.09.25

- `aboot-tools-2020.09.25-linux-x64.sfx`

the zip of tool for linux environment

- `aboot-tools-2020.09.25-win-x64.exe`

the zip of tool for windows 64bits environment

- `aboot-tools-2020.09.25-win-x86.exe`

the zip of tool for windows 32bits environment

1.1. Tool for linux environment

support `aboot`, `arelease` or `adownload` function operated on ubuntu terminal.
Firstly uncompress files from sfx file.

- download `aboot-tools-2020.09.25-linux-x64.sfx` from jenkins aboot tool's link.
- `chmod +x aboot-tools-2020.09.25-linux-x64.sfx`
- `./aboot-tools-2020.09.25-linux-x64.sfx`
- `cd aboot-tools-2020.09.25-linux-x64`
- `./aboot`
- `./arelease`
- `./adownload`

1) `aboot`

In UI mode, include `BurnPage`, `Release Page` and `Misc Page`.

- `BurnPage`: flash data from image zip to flash.
- `ReleasePage`: generate image zip file.

- Misc Page: if Click "Enter production mode" and reset board, it will enter production mode.

2) arelease

In cmdline mode, generate release image zip file for these target.

- Normal mode: burn image to flash.
- EraseAll mode: erase all bytes of flash
- Upload mode: dump data from flash
- Fuse mode: write fuse bits of chip.

output file generated named with function keywords.

3) adownload

In cmdline mode, support burn image data to flash.

1.2. Tool for windows environment

next chapters will detail the introduction based on windows 64 bits environment.

1.2.1. windows 64 bits env

double click about-tools-2020.09.25-win-x64.exe, files are uncompressed to folder about-tools-2020.09.25-win-x64.

1.2.2. windows 32 bits env

double click about-tools-2020.09.25-win-x86.exe, files are uncompressed to folder about-tools-2020.09.25-win-x86.

2

DKB board

Peripherals: XDB JTAG, USB port, UART port, power on/off key, USB_DOWNLOAD key, UART_DOWNLOAD key, RST key.

Refer to the Figure, board and peripherals, the nor flash on DKB board is GD25LQ128D (id: U602). If target is Crane Phone, the nor flash is GD25LQ256C.

To start download steps, please connect all the Peripherals at the first.



If pc is a laptop or host pc connects too many USB devices. USB insufficient power supply makes USB connecting always failed. Suggest adding a USB-Port HUB to solve the problem.

Install required USB ACM or
USB-to-Serial converter drivers

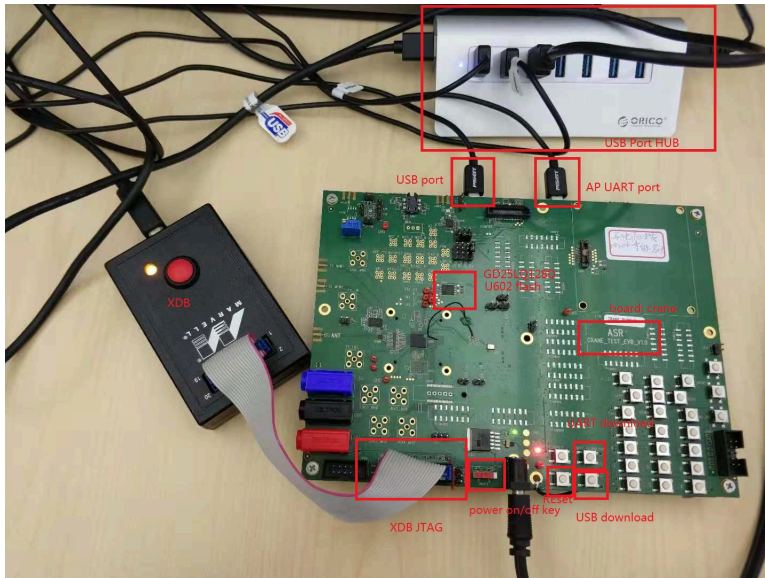


Figure 2.1. board and peripherals

2.1. Install required USB ACM or USB-to-Serial converter drivers

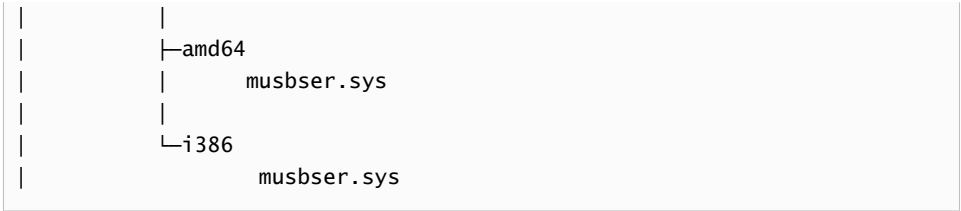
2.1.1. USB driver

Install USB ACM driver for USB downloading mode

If unrecognized USB device is found since USB driver is not installed. Upload USB driver from folder: \\YourAbootToolFolder\driver.

[new version's directory]\driver.

```
├─drivers
|   |   DrvInstaller.exe①
|   |   DrvInstaller_x64.exe②
|   |
|   └─Drv
|       |   musbser.inf
|       |   version.cfg
|       |
|       └─MUSBSer
|           |   musbser_amd64.cat
|           |   musbser_i386.cat
```



- ❶ usb driver for windows 32bits
- ❷ usb driver for windows 64bits

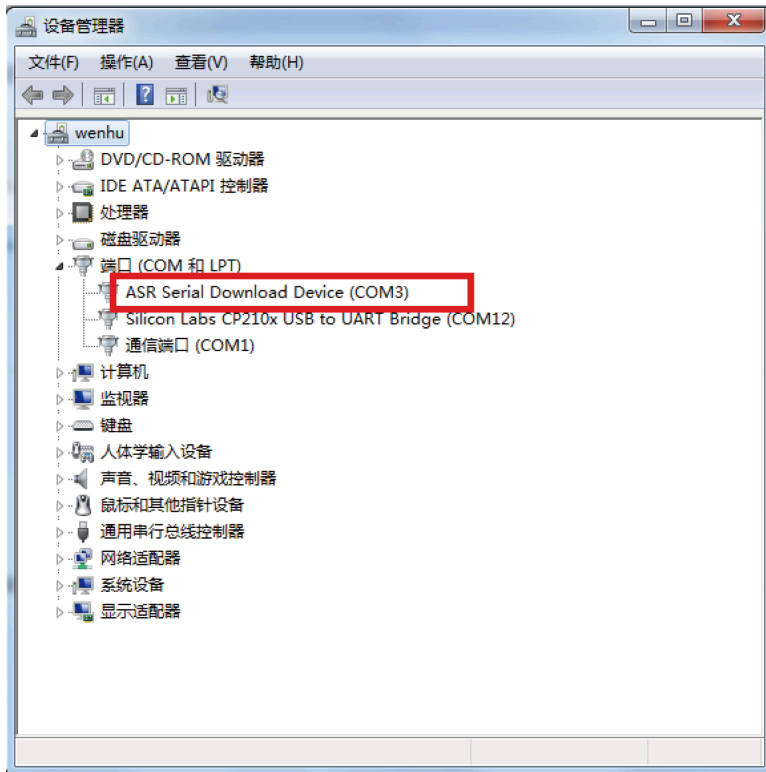


Figure 2.2. ASR USB com

2.1.2. UART driver

If UART's driver is never uploaded in your pc, download it from link:

```
\\fileserver\public-share\USER\uartdriver
```

After driver uploaded, the UART serial com is shown as below:

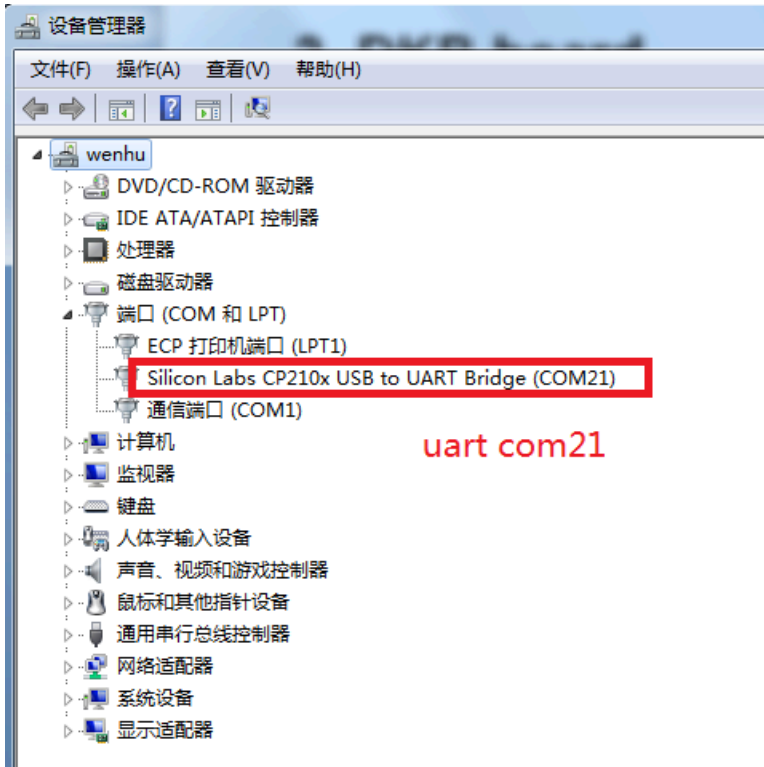


Figure 2.3. UART-to-serial com

2.2. start Putty to monitor UART trace

For example the UART serial line is com21 shown in device manager.

If use putty to monitor com21. Baud rate speed is 115200, add 'Implicit CR in every LF' and 'Implicit LF in every CR' in terminal options. Or there is no enter in the UART trace. If use secure CRT instead of putty, add "New line mode" in session options.

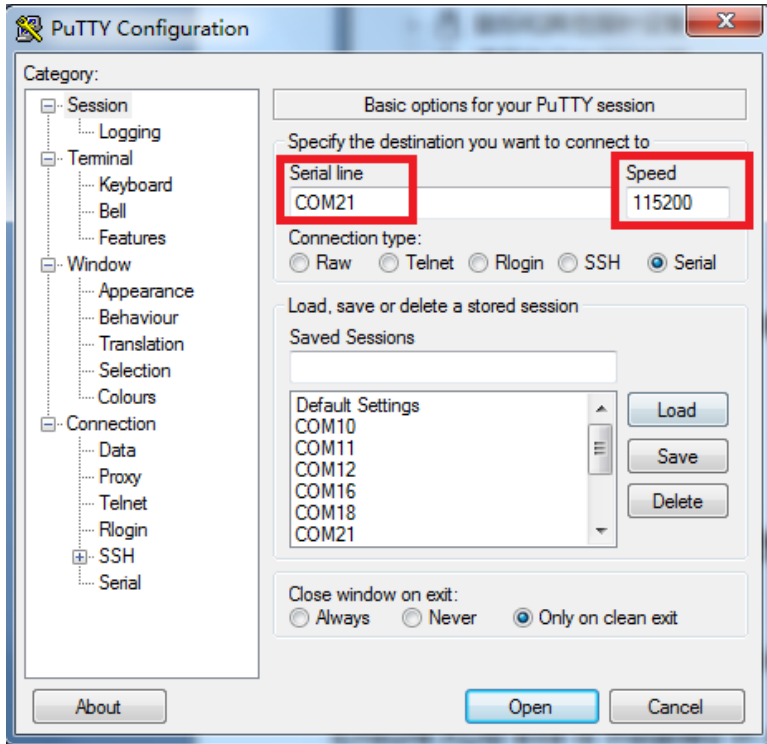


Figure 2.4. putty configuration for UART serial com(1)

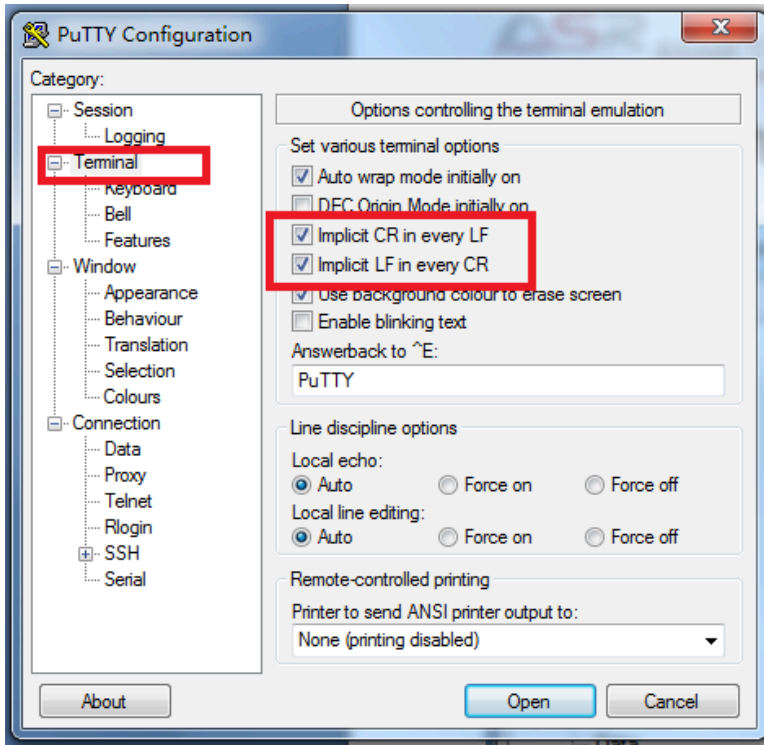


Figure 2.5. putty configuration for UART serial com(2)

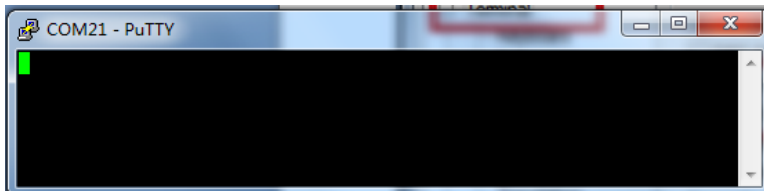


Figure 2.6. putty UART serial com

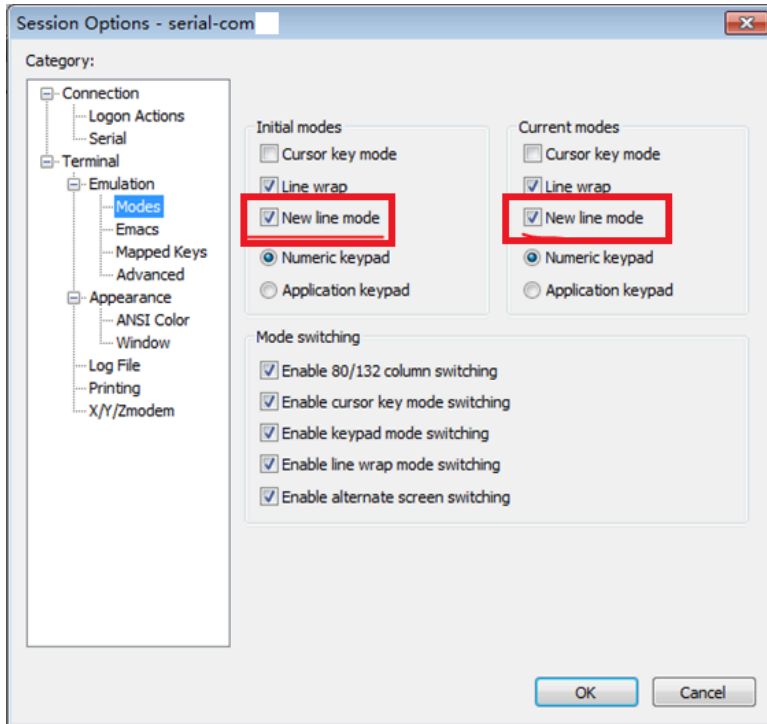


Figure 2.7. secure CRT UART serial com

Prepare generate release package

Abboot.exe(GUI)'release page or **arelease.exe**(Cmdline) is used to generate new release package. The tools are located at \\sh2-filer02\Release\LTE\Crane\Tools.

Abboot.exe or **arelease.exe**

- get config json files from **config** directory.
- get binary files from **images** directory.

It will get products list and support function to generate release package for each product.

The tree list of [new version's directory] is shown as below:

[new version's directory].

```
|  abboot.exe(GUI burn/release tool)
|  adownload.exe(cmdline burn tool)
|  arelease.exe(cmdline release tool)
|  |
|  |---config
|  |   |---flash
|  |   |   MEM_SIMULATION_FLASH.json
|  |   |   QSPI_NAND_128MB_B128KB_S128KB_P2KB.json
|  |   |   QSPI_NAND_256MB_B128KB_S128KB_P2KB.json
|  |   |   QSPI_NOR_16MB_B64KB_S4KB_P256.json
|  |   |   QSPI_NOR_32MB_B64KB_S4KB_P256.json
|  |   |   QSPI_NOR_8MB_B64KB_S4KB_P256.json
|  |   |   SPI_NOR_8MB_B64KB_S4KB_P256.json
|  |   |
|  |   |---partition
|  |   |   CRANE_DUAL_FLASH_LAYOUT.json
```

```

| | CRANE_SINGLE_FLASH_LAYOUT.json
| | JACANA_MEM_FLASH_LAYOUT.json
| |
| |├product
| | | ASR_CRANE_EVB.json
| | | ASR_JACANA_EVB.json
| | |
| |├security
| | | ECDSA_KEYS.json
| | | RSA_KEYS.json
| | |
| | |├key
| | | | ecdsa_non_trusted_world_key.pem
| | | | ecdsa_nt_fw_key.pem
| | | | ecdsa_nt_os_key.pem
| | | | ecdsa_rootpk_sha256.bin
| | | | ecdsa_root_key.pem
| | | | ecdsa_scp_fw_key.pem
| | | | ecdsa_soc_fw_key.pem
| | | | ecdsa_tos_fw_key.pem
| | | | ecdsa_trusted_world_key.pem
| | | | rsa_non_trusted_world_key.pem
| | | | rsa_nt_fw_key.pem
| | | | rsa_nt_os_key.pem
| | | | rsa_rootpk_sha256.bin
| | | | rsa_root_key.pem
| | | | rsa_scp_fw_key.pem
| | | | rsa_soc_fw_key.pem
| | | | rsa_tos_fw_key.pem
| | | | rsa_trusted_world_key.pem
| | |
| | |├template
| | | | CRANE_EVB.json
| | | | JACANA_EVB.json
| | |
| |├docs
| | | AbootPG.pdf
| | | AbootUG.pdf
| | | SecBootUG.pdf
| | |
| |├drivers
| | | DrvInstaller.exe
| | | DrvInstaller_x64.exe
| | |
| | |├Drv
| | | | musbser.inf

```

```
| | | Version.cfg
| | |
| | | └─MUSBSer
| | | | musbser_amd64.cat
| | | | musbser_i386.cat
| | | |
| | | | └─amd64
| | | | | musbser.sys
| | | | |
| | | | └─i386
| | | | | musbser.sys
| | |
| | └─images
| | | apn.bin
| | | arom.bin
| | | boot2.bin
| | | boot33.bin
| | | cp.bin
| | | dsp.bin
| | | flasher-jacana.bin
| | | flasher.bin
| | | jacana.bin
| | | preboot.bin
| | | pvt.bin
| | | ReliableData.bin
| | | rf.bin
| | | updater.bin
| | |
| | └─2019.01.15
| | | boot2.bin
| | | flasher.bin
| | | fuse.bin
| | | fuse.img
| | | preboot.bin
| | |
| | └─2020.04.08
| | | boot2.bin
| | | flasher.bin
| | | fuse.bin
| | | preboot.bin
| | |
| | └─swiftshader
| | | libEGL.dll
| | | libGLESv2.dll
```

└─upload

3.1. The information of product

All supported products are defined under

config/product/ASR_CRANE_EVB.json

config/product/ASR_CRANE_EVB.json.

```
{
  /* comment: default parameters defined for each product, each
  parameter for one product can be defined with specail value to replace
  default one. */
  "name": "ASR_CRANE_EVB",
  "version": "0.5",
  "version-bootrom": "2019.01.15",
  "template": "CRANE_EVB",
  "flashes": [
    {
      "name": "external",
      "port": "QSPI",
      "flash": "QSPI_NOR_16MB_B64KB_S4KB_P256"
    }
  ],
  "layout": "CRANE_SINGLE_FLASH_LAYOUT",
  "keyAlg": "rsa",
  "hashAlg": "sha256",
  "secureBoot": false,
  "fota": [
    "system"
  ],
  "variants": [
    {
      "name": "CRANE_A0_08MB", ❶
      "flashes": [
        {
          "name": "external",
          "port": "QSPI",
          "flash": "QSPI_NOR_8MB_B64KB_S4KB_P256"
        }
      ],
      "partitions": [
        {
```



```

        "name": "nvm",
        "size": "512KiB"
    }
]
},
{
    "name": "CRANE_A0_16MB"②
},
{
    "name": "CRANE_A0_16+8MB",③
    "flashes": [
        {
            "name": "external",
            "port": "QSPI",
            "flash": "QSPI_NOR_16MB_B64KB_S4KB_P256"
        },
        {
            "name": "internal",
            "port": "SSP2",
            "flash": "SPI_NOR_8MB_B64KB_S4KB_P256"
        }
    ],
    "layout": "CRANE_DUAL_FLASH_LAYOUT",
    "fota": [
        "system",
        "system2"
    ]
},
{
    "name": "CRANE_A0_32MB",④
    "flashes": [
        {
            "name": "external",
            "port": "QSPI",
            "flash": "QSPI_NOR_32MB_B64KB_S4KB_P256"
        }
    ]
},
{
    "name": "CRANEG_Z2_16+8MB",⑤
    "version-bootrom": "2020.04.08",
    "flashes": [
        {
            "name": "external",
            "port": "QSPI",

```

```

        "flash": "QSPI_NOR_16MB_B64KB_S4KB_P256"
    },
    {
        "name": "internal",
        "port": "SSP2",
        "flash": "SPI_NOR_8MB_B64KB_S4KB_P256"
    }
],
"layout": "CRANE_DUAL_FLASH_LAYOUT",
"fota": [
    "system",
    "system2"
]
},
{
    "name": "CRANEG_Z2_32MB", ❹
    "version-bootrom": "2020.04.08",
    "flashes": [
        {
            "name": "external",
            "port": "QSPI",
            "flash": "QSPI_NOR_32MB_B64KB_S4KB_P256"
        }
    ]
},
{
    "name": "CRANEG_Z2_32+8MB", ❺
    "version-bootrom": "2020.04.08",
    "flashes": [
        {
            "name": "external",
            "port": "QSPI",
            "flash": "QSPI_NOR_32MB_B64KB_S4KB_P256"
        },
        {
            "name": "internal",
            "port": "SSP2",
            "flash": "SPI_NOR_8MB_B64KB_S4KB_P256"
        }
    ],
    "layout": "CRANE_DUAL_FLASH_LAYOUT",
    "fota": [
        "system",
        "system2"
    ]
}

```

```
    },  
    {  
      "name": "CRANEGM_A0_16MB", ❸  
      "product": "arom-tiny",  
      "version-bootrom": "2020.07.30"  
    },  
    {  
      "name": "CRANEGM_A0_32MB", ❹  
      "product": "arom-tiny",  
      "version-bootrom": "2020.07.30",  
      "flashes": [  
        {  
          "name": "external",  
          "port": "QSPI",  
          "flash": "QSPI_NOR_32MB_B64KB_S4KB_P256"  
        }  
      ]  
    }  
  ]  
}
```

- ❶ CRANE_A0_08MB, QSPI flash is 8M, so use QSPI_NOR_8MB_B64KB_S4KB_P256 as flash layout
- ❷ CRANE_A0_16MB, same as default parameters.
- ❸ CRANE_A0_16+8MB, has internal spi flash 8M, so use CRANE_DUAL_FLASH_LAYOUT layout.
- ❹ CRANEG_Z2_32MB, bootrom's version is "2020.04.08"
- ❺ CRANEGM_A0_16MB, bootrom's version is "2020.07.30"

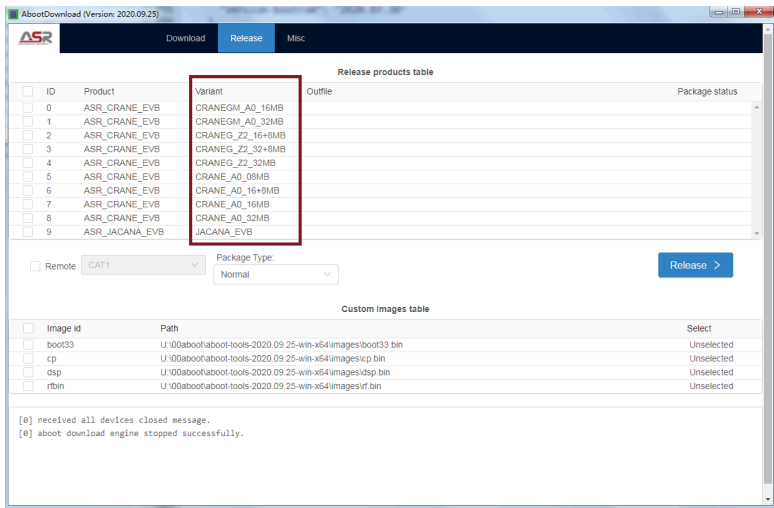


Figure 3.1. supported production list in release page

All supported flashes are defined in flash json.

```
[new version's directory]
├─config
│   └─flash
│       │   MEM_SIMULATION_FLASH.json
│       │   QSPI_NAND_128MB_B128KB_S128KB_P2KB.json
│       │   QSPI_NAND_256MB_B128KB_S128KB_P2KB.json
│       │   QSPI_NOR_16MB_B64KB_S4KB_P256.json
│       │   QSPI_NOR_32MB_B64KB_S4KB_P256.json
│       │   QSPI_NOR_8MB_B64KB_S4KB_P256.json
│       │   SPI_NOR_8MB_B64KB_S4KB_P256.json
```

Key words for flash info:

- Flash interface type :QSPI
- Flash type: NAND/ NOR
- Memory size: 128Mbytes /256Mbytes/16Mbytes/32Mbytes
- Block size: 128Kbytes/ 64Kbytes
- Sector size: 128Kbytes / 4Kbytes
- Page size: 2Kbytes/ 256bytes

3.2. The information of flash

If add a new flash. For example, GD25LQ128D, which is a norflash, memory size is 16Mbytes, block size is 64Kbytes, sector size is 4Kbytes, page size is 256bytes, add the new one in

```
config/flash/ QSPI_NOR_16MB_B64KB_S4KB_P256.json
```

Add the name/manufacture/vendorId/deviceId of the flash in devices.

QSPI_NOR_16MB_B64KB_S4KB_P256.json.

```
{
  "name": "QSPI_NOR_16MB_B64KB_S4KB_P256",
  "type": "QSPI_NOR",
  "minIoSize": 1,
  "pageSize": 256,
  "pebSize": "4KiB",
  "sectorSize": "4KiB",
  "blockSize": "64KiB",
  "flashSize": "16MiB",
  "erasedValue": "0xFF",
  "devices": [
    {
      "name": "FM25M4AA",
      "manufacturer": "DosiSilicon",
      "vendorId": "0xF8",
      "deviceId": "0x4218"
    },
    {
      "name": "GD25LQ128D",
      "manufacturer": "GigaDevice",
      "vendorId": "0xC8",
      "deviceId": "0x6018"
    },
    {
      "name": "DQ25Q128AL",
      "manufacturer": "DouQiTech",
      "vendorId": "0x54",
      "deviceId": "0x6018"
    },
    {
      "name": "W25Q128JW",
      "manufacturer": "Winbond",
```

```
    "vendorId": "0xEF",  
    "deviceId": "0x6018"  
  },  
  {  
    "name": "MX25U12835F",  
    "manufacturer": "Macronix",  
    "vendorId": "0xC2",  
    "deviceId": "0x2538"  
  }  
]  
}
```

3.3. The information of images

- CRANE_A0 products :

input binary files read from images/2019.01.15, bootrom version is 2019.01.15

- CRANEG_Z2 products :

input binary files read from images/2020.04.08, bootrom version is 2020.04.08.

- CRANEM_A0 products :

input binary files read from images/2020.07.30, bootrom version is 2020.07.30.

3.4. Binary files introduction:

- preboot.bin:

Run in ATCM, initializes PMIC, initialize PSRAM. Preboot will be called when burning program or booting.

- flasher.bin:

Run in ATCM, initializes flash in 4x quad mode and 104MHz, supply block erase and 4x PP write flash functions. Only called when burning program.

- boot2.bin:

Run in ATCM, initializes flash in 4x quad mode and 104MHz, supply 4x fast read in LUT, enable XIP read. Only called when booting. Read partition table to BTCM, read boot33.bin from bootload.ubi to PSRAM. Transfer control to boot33.

- boot33.bin:

Compiled independently. initialize flash XIP. run in PSRAM when booting, set mpu for XIP flash, run seagull on XIP.

- seagull.bin:

The binary for CP. Compiled independently. Run on XIP.

- dsp.bin:

The binary for DSP. Compiled independently. Run on XIP.

Note:

1. preboot.bin, flasher.bin and boot2.bin must consistent to arom.bin. If inconsistent, when burn program , kernel panic will be happen when call binary compiled with wrong bootrom version.
2. if PSRAM initialization is not right, flasher.info is failed to be parsed when downloading. It depends on the PSRAM code in preboot. Please check burn log to know what psram type is used, winbond or AP, 8+8M or 16M. The info is fused in block3's bits 168~170.
3. flasher.bin only called when burning program, supply 4x quad mode write fuction for fast download. Flasher.bin will not be used when booting.
4. flasher and boot2 use same flash driver in Qspi_nor_flash.c/ Qspi_nand_flash.c.
5. boot33 has own flash driver, if flash driver updated, need merge modification to boot33 code.

images for each bootrom version.

```
[new version's directory]
└─images❶
```

Binary files introduction:

```
| | apn.bin
| | arom.bin
| | boot2.bin
| | boot33.bin
| | cp.bin
| | dsp.bin
| | flasher-jacana.bin
| | flasher.bin
| | jacana.bin
| | preboot.bin
| | pvt.bin
| | ReliableData.bin
| | rf.bin
| | updater.bin
| |
| | 2019.01.15②
| |     boot2.bin
| |     flasher.bin
| |     fuse.bin
| |     fuse.img
| |     preboot.bin
| |
| | 2020.04.08③
| |     boot2.bin
| |     flasher.bin
| |     fuse.bin
| |     preboot.bin
```

- ❶ CRANEGM_A0_XXMB, bootrom's version is "2020.07.30"
- ❷ CRANE_A0_XXMB, bootrom's version is "2019.01.15"
- ❸ CRANEG_Z2_XXMB, bootrom's version is "2020.04.08"

3 Fuse bits for psram types:

- PSRAM Edition bank3 bit170
- PSRAM Edition bank3 bit169
- PSRAM Edition bank3 bit168

Psram types, bank3 bit[170~168]:

- AP 16MB : 000
- Winbond 16MB : 001

- AP 8MB+8MB : 010
- Winbond 8MB+8MB : 100
- AP 8MB : 101
- Winbond 8MB : 110

Chips with different psram types. Code read psram type id form fuse bits:

- crane A0 : psram type is AP 16MB.
- craneG Z2: psram type is AP 8MB+8MB.
- craneM a0: psram type is Winbond 8MB+8MB.

psram log shown when burm program mode.

```

17:20:18.669 <COM3> #=> [INFO: Psram      ] PSRAM PHY frequency changed
to 78
17:20:18.669 <COM3> #=> [PRI : Psram      ] [PSRAM]
psram_init_crane_a0.❶
17:20:18.670 <COM3> #=> [INFO: Psram      ] #####
17:20:18.670 <COM3> #=> [INFO: Psram      ] Version ID : 0
17:20:18.670 <COM3> #=> [PRI : Psram      ] AP 16MB❷
17:20:18.671 <COM3> #=> [PRI : Psram      ] No embedded flash.❸
17:20:18.728 <COM3> #=> [INFO: Psram      ] #####
17:20:18.728 <COM3> #=> [INFO: Psram      ] #####
17:20:18.729 <COM3> #=> [INFO: Psram      ] MP DEVICE
17:20:18.729 <COM3> #=> [INFO: Psram      ] MR[2] VALUE = 0x5
17:20:18.729 <COM3> #=> [INFO: Psram      ] #####
17:20:18.729 <COM3> #=> [INFO: Psram      ] mmap psram address spaces:
reg[0xc0100004] = 0x7e080001
17:20:18.729 <COM3> #=> [INFO: Psram      ] psram cache
enabled !!! :cache size=[128B], @[0xc0104000]=[0x000A3B11]
17:20:18.729 <COM3> #=> [INFO: Psram      ] Do Global Reset
17:20:18.729 <COM3> #=> [INFO: Psram      ] Enable fast_miss_acc
17:20:18.729 <COM3> #=> [INFO: Psram      ] RB: MR0x0_CS0:
@[0xc0108034]=[0x00000000]
17:20:18.729 <COM3> #=> [INFO: Psram      ] RB: MR0x0_CS0:
@[0xc0108038]=[0x8D138D13]
17:20:18.729 <COM3> #=> [INFO: Psram      ] RB: MR0x0_CS1:
@[0xc0108034]=[0x00800000]
17:20:18.729 <COM3> #=> [INFO: Psram      ] RB: MR0x0_CS1:
@[0xc0108038]=[0x8D138D13]
17:20:18.729 <COM3> #=> [INFO: Psram      ] RB: MR0x1_CS0:
@[0xc0108034]=[0x00000001]

```

Binary files introduction:

```
17:20:18.729 <COM3> #=> [INFO: Psram      ] RB: MR0x1_CS0:
@0xc0108038]=[0x958D958D]
17:20:18.729 <COM3> #=> [INFO: Psram      ] RB: MR0x1_CS1:
@0xc0108034]=[0x00800001]
17:20:18.729 <COM3> #=> [INFO: Psram      ] RB: MR0x1_CS1:
@0xc0108038]=[0x958D958D]
17:20:18.729 <COM3> #=> [INFO: Psram      ] PSRAM PHY frequency changed
to 175
```

- ❶ chip is crane A0, psram driver use the one for crane A0. crane and craneG's psram driver is different.
- ❷ psram type is AP 16MB, read from fuse bits[170~168] in bank3.
- ❸ no 8M SPI flash in craneA0, read from fuse bit[28] in bank0.

Generate release package

Two methods are supplied to generate release package:

1. Arelease.exe(Cmdline)
2. Aboot.exe(GUI)

4.1. Arelease.exe

In cmdline mode, enter Arelease.exe in windows's cmd.exe, the help information is shown as below:

execute arelease.exe in windows's cmd.exe.

```
about-tools-2020.09.25-win-x64>arelease.exe
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
```

```
-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=rsa
arelease.exe -c . -l❶
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.zip❷
arelease.exe -c config -I images -g -p ASR_CRANE_EVB -v CRANE_A0_16MB
```

1) Get products list

execute arelease.exe -l in windows's cmd.exe.

```
about-tools-2020.09.25-win-x64>arelease.exe -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+8MB
  0.7         CRANE_A0_16MB❶
  0.8         CRANE_A0_32MB
  1.          ASR_JACANA_EVB
  1.0         JACANA_EVB
```

2) Generate release package with default name

If add **-q**, the arelease.exe will exit automatically after process is completed.

execute arelease.exe with -q in windows's cmd.exe.

```
aboot-tools-2020.09.25-win-x64> arelease.exe -c . -q -g -p ASR_CRANE_EVB
-v CRANE_A0_16MB ASR_CRANE_EVB.zip

Parsing command line paramters ...
Release package file name is "ASR_CRANE_EVB.zip"
Finished parsing command line paramters.
Processing all images needed ...
    Generating fip image "preboot" with id "preboot.img" ...
        Opening image "preboot_bin" from "U:\00aboot\aboot-tools-2020.09.25-
win-x64\images\2019.01.15\preboot.bin" ...
        Done.
    Done.
    Generating fip image "flasher" with id "flasher.img" ...
        Opening image "flasher_bin" from "U:\00aboot\aboot-tools-2020.09.25-
win-x64\images\2019.01.15\flasher.bin" ...
        Done.
    Done.
    Generating finf image "flashinfo" with id "flashinfo.bin" ...
    Done.
    Opening image "rd" from "U:\00aboot\aboot-tools-2020.09.25-win-
x64\images\ReliableData.bin" ...
    Done.
    Opening image "apn" from "U:\00aboot\aboot-tools-2020.09.25-win-
x64\images\apn.bin" ...
    Done.
    Opening image "cp" from "U:\00aboot\aboot-tools-2020.09.25-win-
x64\images\cp.bin" ...
    Done.
    Opening image "dsp" from "U:\00aboot\aboot-tools-2020.09.25-win-
x64\images\dsp.bin" ...
    Done.
    Opening image "rfbin" from "U:\00aboot\aboot-tools-2020.09.25-win-
x64\images\rf.bin" ...
    Done.
    Opening image "updater" from "U:\00aboot\aboot-tools-2020.09.25-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 "U:\00aboot\aboot-tools-2020.09.25-
win-x64\images\2019.01.15\boot2.bin" ...
    Done.
    Generating lzma image "boot33_lzma" with id "boot33.lzma" ...
    Opening image "boot33" from "U:\00aboot\aboot-tools-2020.09.25-
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.
```

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

4.2. Abboot.exe

In GUI mode, release page supplys the function to generate release package.

1) Get products list

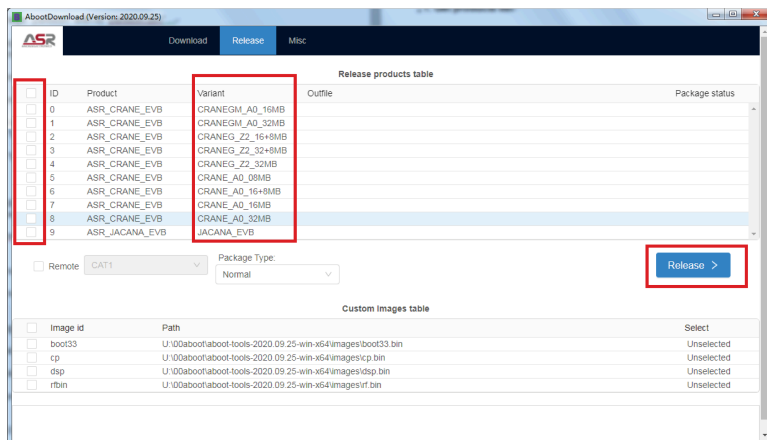


Figure 4.1. Abboot's Release page

2) Select products

User can select one or more products to create release zip file.

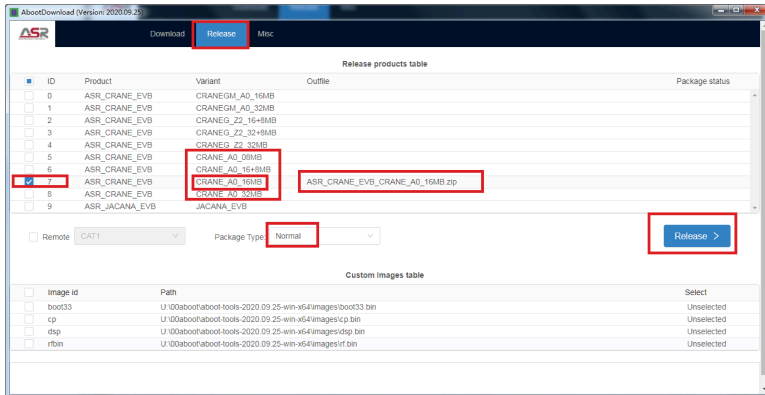


Figure 4.2. select the products for crane A0_16M

3) Click Release Button

When about engine is generating a release product, status is shown as **Generating**.

Release button is disabled till all the products' package are finished.

In log console, user can see the path of input binary files:

```
preboot.bin/flasher.bin/boot33.bin/seagall.bin/msa.bin.
```

It is good to confirm path for each binary file.

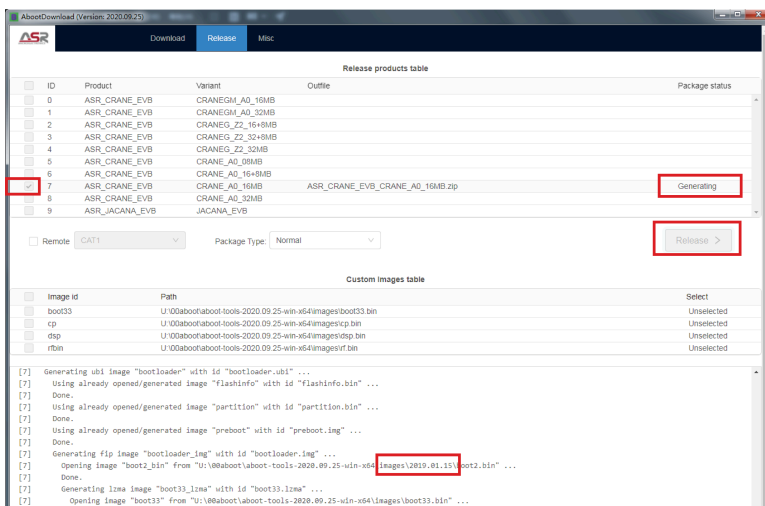


Figure 4.3. generating zip for crane A0_16M

If a product's package is finished, it is unselected automatically. Output file and status are shown as below:

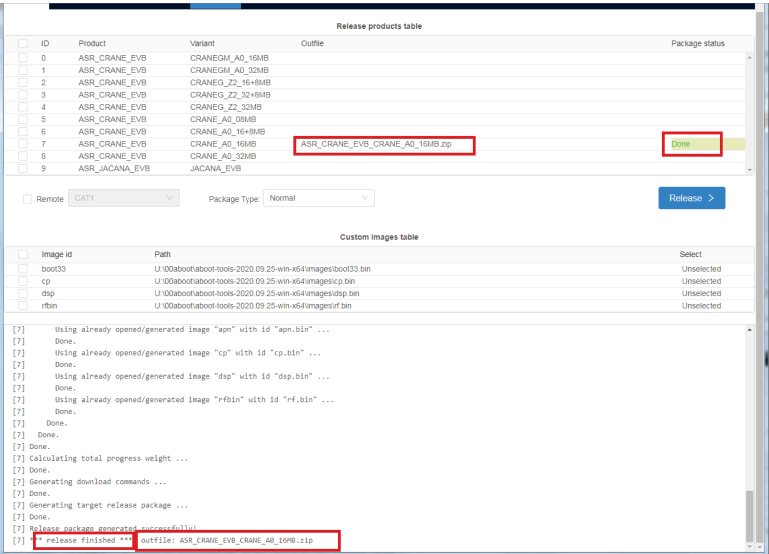


Figure 4.4. packages are released

4) Check the release package under [new version's directory].

[new version's directory]\ASR_CRANE_EVB_CRANE_A0_16MB.zip

4.3. How to change custom images

Custom images includes: **boot33.bin**, **cp.bin**, **dsp.bin**, **rf.bin**. When release package, the files are input from **images** folder defaultly, shown as below:

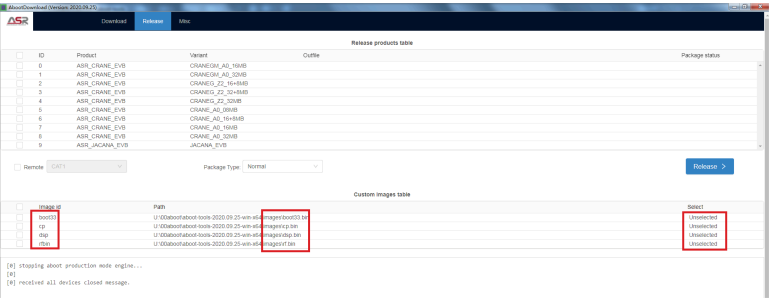


Figure 4.5. custom images table

User can change the input source for a custom image. eg: boot33.bin

- select a product type
- select a image id
- click the hand magnifier Button to update boot33 path.

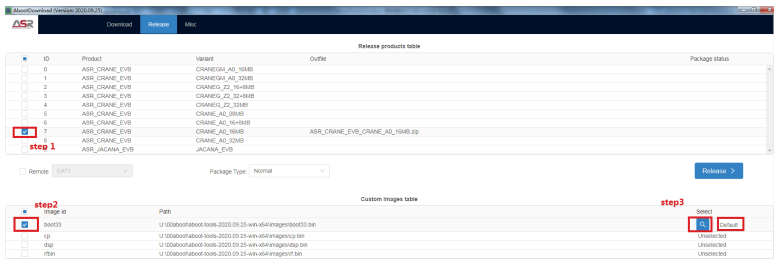


Figure 4.6. change boot33 input path

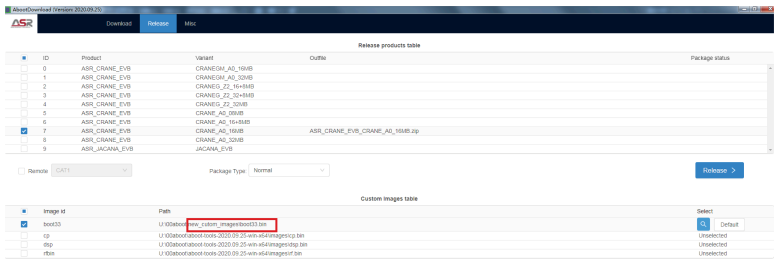


Figure 4.7. boot33 input from user own folder

- If user wants to use default patch, click default button.



The selected new path will be recorded in tool code no matter switch UI pages from release page to burn page. So user should confirm the path of files before release package. If about tool is closed, file path will use default one next time.

If 4 files are input from user own folder, user can check the path from log console when generating release package.

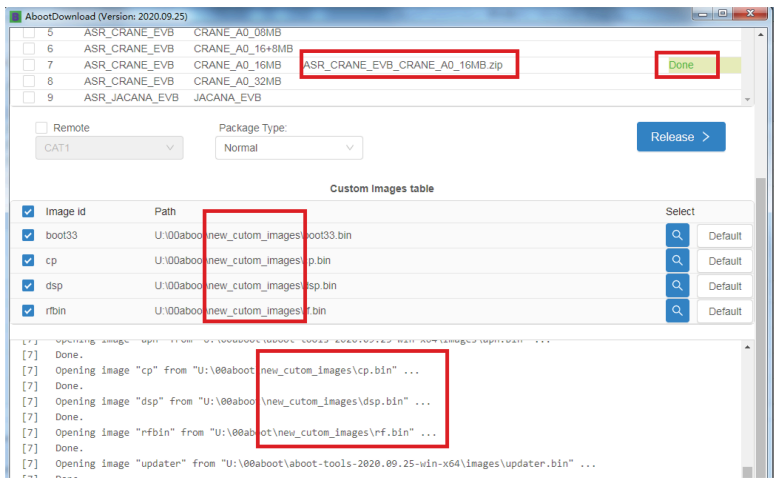


Figure 4.8. 4 files are input from user own folder

5

Burn program mode

5.1. Enter burn program mode normally

Press **USB_DOWNLOAD** or **UART_DOWNLOAD** key, not release, meanwhile power on or reset board. The board will enter burn program mode.

5.2. Enter burn program mode via XDB(crane Z1)

Enter burn program mode via XDB at first time only for crane Z1, other chip no need.

Prepare for entering burn program mode at first time, please connect XDB tool to target by JTAG. Arom.bin will be burned in flash. Run the arom.bin to enter downloading mode. If arom.bin exists and no need to be updated, use USB_DOWNLOAD/UART_DOWNLOAD key + RST key to enter downloading mode. If arom.bin is updated, please use XDB tool to burn new arom.bin again.

Ensure XDB exe is installed in your WINPC. Install link:

```
\\fileserver\public-share\USER\xdb w_XDB_u_5.7.039.exe
```

Copy **\\sh2-filer02\Release\LTE\Crane\Tools\[lastest release version]** to your WINPC.

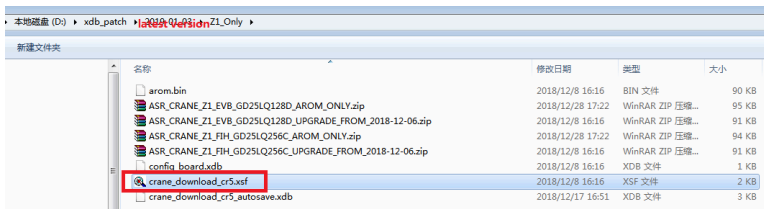
Suppose it is named as **\\YourAbootToolFolder** in the document.

5.2.1. Burn Steps via XDB

Enter directory **\\YourAbootToolFolder\Z1_Only**.

Files under the folder:

- ASR_CRANE_Z1_EVB_XXXXX_AROM_ONLY.zip: used to write arom binary in 512K on Z1 board. It need to be done after erase all on Z1.
- ASR_CRANE_Z1_EVB_XXXXX_UPGRADE_FROM_XXXX-X-XX.zip: used to upgrade Z1 board from latest version to new version.
- arom.bin: xdb will burn it in flash. bootloader.ubi is consistent with the arom.bin.
- config_board.xdb: configuration for XDB.
- crane_download_cr5.xsf: double click it to start XDB tool.



名称	修改日期	类型	大小
arom.bin	2018/12/8 16:16	BIN 文件	90 KB
ASR_CRANE_Z1_EVB_GD25LQ128D_AROM_ONLY.zip	2018/12/28 17:22	WinRAR ZIP 压缩...	95 KB
ASR_CRANE_Z1_EVB_GD25LQ128D_UPGRADE_FROM_2018-12-06.zip	2018/12/8 16:16	WinRAR ZIP 压缩...	91 KB
ASR_CRANE_Z1_FIH_GD25LQ256C_AROM_ONLY.zip	2018/12/28 17:22	WinRAR ZIP 压缩...	94 KB
ASR_CRANE_Z1_FIH_GD25LQ256C_UPGRADE_FROM_2018-12-06.zip	2018/12/8 16:16	WinRAR ZIP 压缩...	91 KB
config_board.xdb	2018/12/8 16:16	XDB 文件	1 KB
crane_download_cr5.xsf	2018/12/8 16:16	XSF 文件	2 KB
crane_download_cr5_autosave.xdb	2018/12/17 16:51	XDB 文件	3 KB

Figure 5.1. Z1_only file list

Double click **crane_download_cr5.xsf** to start XDB, do not click the “**Start**” button at once.

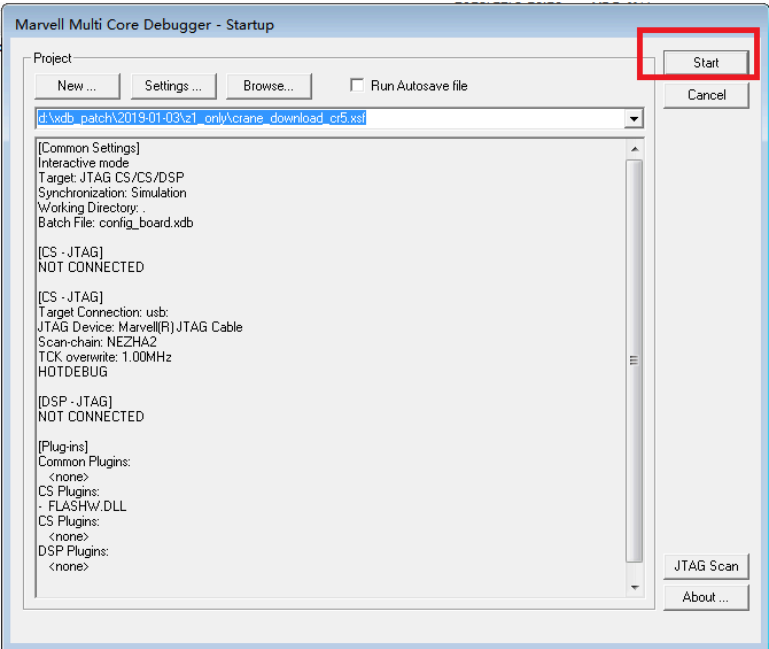


Figure 5.2. XDB startup

Press **USB_DOWNLOAD** or **UART_DOWNLOAD** key in the EVB board and not release, then click **Start** button to start XDB, the board will enter the selected burn program mode after a while.

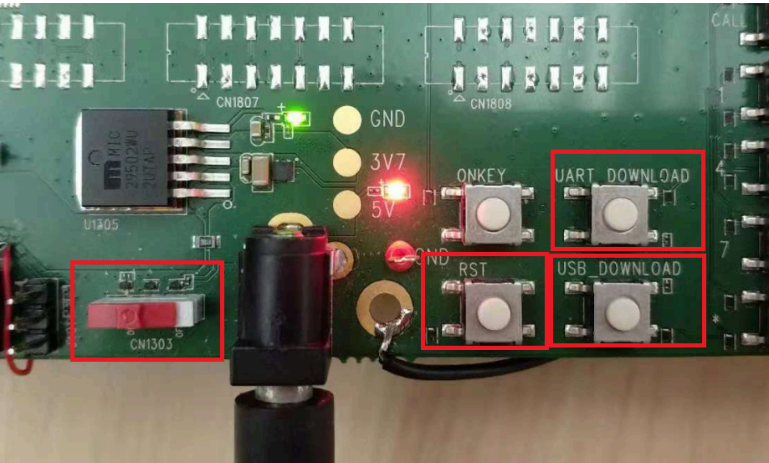


Figure 5.3. USB/UART download key on board

When the USB-to-Serial com is found in device manager. Then the device had been entered into the downloading mode.

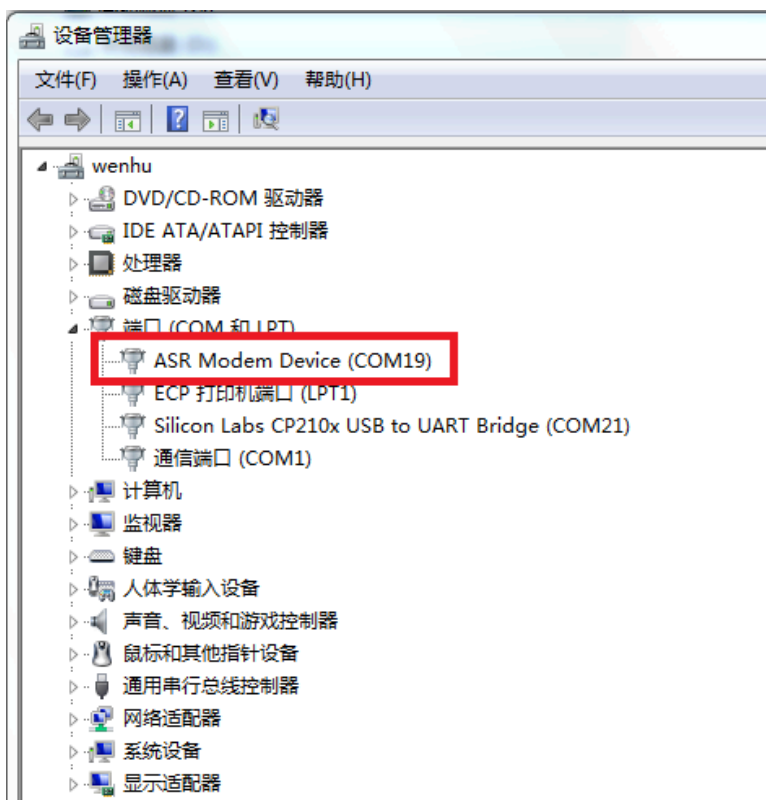


Figure 5.4. USB serial line shown when in burn program mode

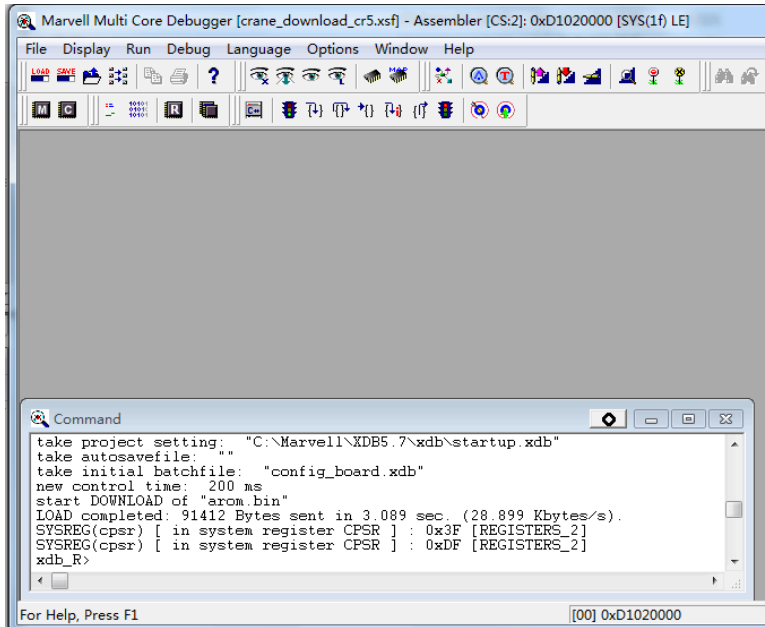


Figure 5.5. XDB window shown when in burn program mode

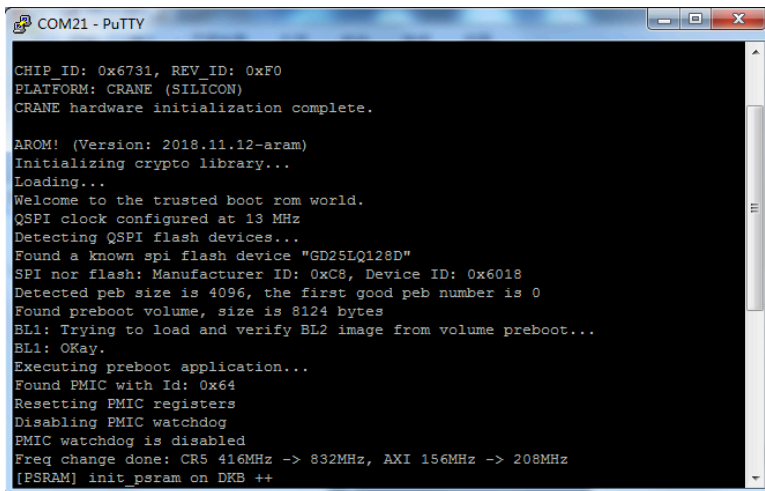


Figure 5.6. putty trace shown when in burn program mode

6

Burn package to target

The image package(zip file) is generated by aboot.exe or arelease.exe. User will use the zip file to burn images to flash of board.

6.1. Download release package by USB

Two methods are supplied to burn image package:

1. Adownload.exe(Cmdline)
2. Aboot.exe(GUI)

6.2. Adownload.exe

In cmdline mode, enter adownload.exe in windows's cmd.exe, the help information is shown as below:

execute adownload.exe in windows's cmd.exe.

```
about-tools-2020.09.25-win-x64>adownload.exe

Asrmicro 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 device①
  -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 production mode, default
is upgrade mode
-f, --at-fallback          Send AT cmd to fallback to download mode
-r, --reboot               Reboot device after finished
-q, --quit                 Quit application after any port finished②
```

Example:

```
adownload.exe -p COM1,COM2 -s 115200 aboot.zip
adownload.exe -u -a -s 115200 aboot.zip③
adownload.exe -p COM1 -a -s 115200 aboot.zip
```

- ① adownload example for burn normal images. It is good to add -q.
Download images and burn to flash.

Command:

```
adownload.exe -q -u -a -s 115200 aboot.zip
```



-q: quit application after any port finished.

6.3. Abboot.exe

GUI application if windows.



If Some PC's graphics card on PC is too old, they may met problem when open Abboot.exe. It may show black and exit automatically. If user meet the same problem, copy **libEGL.dll** and **libGLESv2.dll** from child folder **swiftshader** to the folder which **adownload.exe** is in.

software graphics driver files.

```
[new version's directory]
| aboot.exe
| adownload.exe
| arelease.exe
|—swiftshader
|   libEGL.dll①
```

| libGLv2.dll^②

① libEGL.dll: software graphics drivers

② libGLv2.dll: software graphics drivers

Download Steps:

1) Start exe

Double click \\YourAboutToolFolder\\About.exe on your win pc. A burn page will be shown.

There are some input options:

- Release package:

Need select a released packaged zip file from \\YourAboutToolFolder\\. It can be a normal zip(eg: ASR_CRANE_EVB_CRANE_A0_16MB.zip) or erase-all/fuse/upload image zip(refer to other chapters), depend on your target.

- Production mode:

Checkbox is checked: burn program in production mode. Display log is disabled.

Checkbox is unchecked: burn program in development mode. Display log is enabled.

- save log:

Only enabled in development mode.

Checkbox is checked: dump log will be saved to **about.log**.

Checkbox is unchecked: no log file.

- Display log:

Only enabled in development mode.

Checkbox is checked: after click "Start >", the burning logs are printed.

Checkbox is unchecked: no log.

- USB device only:

Checkbox is checked: only USB COMs are shown.

Checkbox is unchecked: USB and UART COMS are all shown.

- Auto enable for USB:

Checkbox is checked: the USB COMs is enabled to burn program automatically after click "Start >".

Checkbox is unchecked: USB COMs or UART COMs need be enabled by click button manually.

- AT fallback

Checkbox is checked: after click **Start** button, tool will send AT command to the board which already booted successfully. It realizes that board can enter burn program mode remotely.

Checkbox is unchecked: must press **USB_DOWNLOAD** or **UART_DOWNLOAD** key, not release, meanwhile power on or reset board locally. Then board will enter burn program mode.

- Reboot

Checkbox is checked: has reboot function. If burn program process is finished, board automatically reboot, board will boot in normal mode.

Checkbox is unchecked: no reboot function.

- Baud rate:

It is used for burn program by UART. By default , it is 115200bps. Max rate is 1842000bps for uart.

It is not used for burn program by USB. It is automatically adjusted.

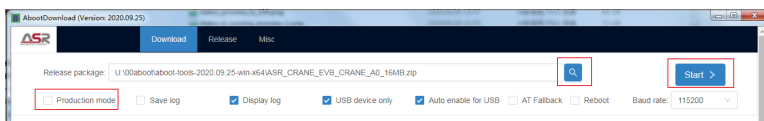


Figure 6.1. Aboot.exe's input and checkboxes in Burn page

2) Select a package

click the hand magnifier to select a package zip file.

3) Enable development/production mode

3.1) Enable development mode (developer need)

After select a package file, development mode is enabled defaultly.

Partition Table and Download command table from the package are shown as below.

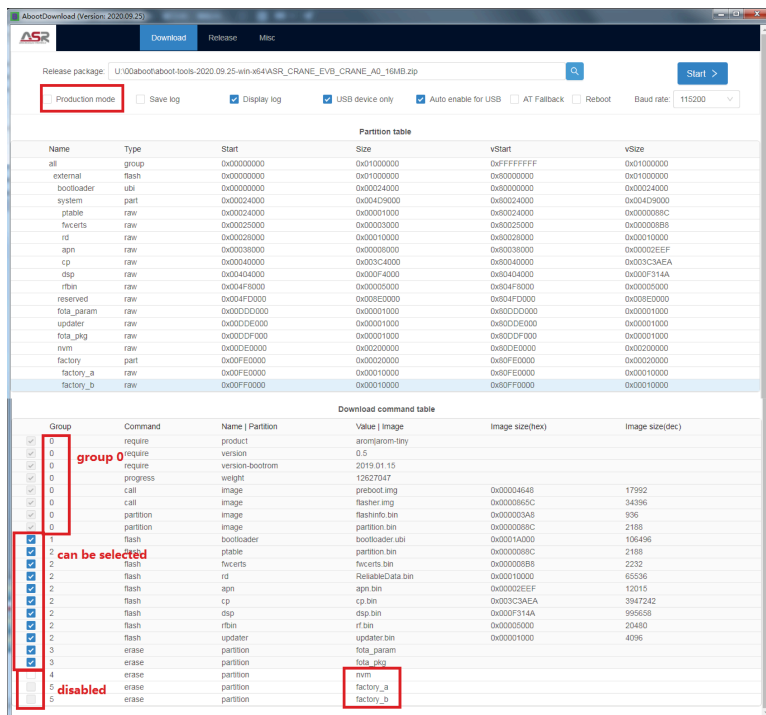


Figure 6.2. development mode

- The images are grouped. select/unselect one command, others with the same group id will be selected/unselected too.
- Commands with group id=0(grey selection box without tick) will be forced to execute.
- Commands with blue selection box. They are selected defaultly, means that commands will be executed defaultly. They can be unselected by user manually.

- Commands with white selection box. They are not be executed defaultly.They can be unselected by user manually. eg: NVM partition is disabled defaultly. CP saves lasting nvm files which has information about configuration in NVM partition. So user should not erase NVM every time. Nvm only should be erased when burn program in factory in production mode or NVM partition is destructured by other problem in development mode. After **NVM** is erased, the nvm files will be recovered by reliable data in **Rd** partition.
- Commands with grey selection box without tick, they are forced to be disabled. factory_a and factory_b is written in factory by calibration tool. They cannot never be erased by developer.

3.2) Enable production mode (factory need)

In production mode, download command table cannot be selected. All commands need to be executed. The log will not be printed out. Production burn program only can be executed in factory before calibration. It erase the factory_a and factory_b partitions which used for save calibration data.

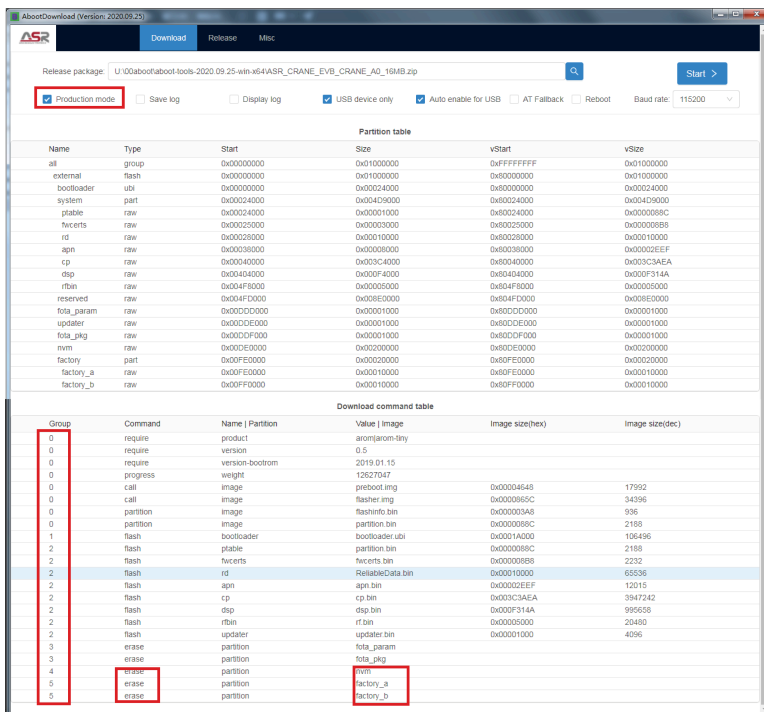


Figure 6.3. production mode

4) Start Abboot engine

Click **Start >** button, Abboot engine will start.

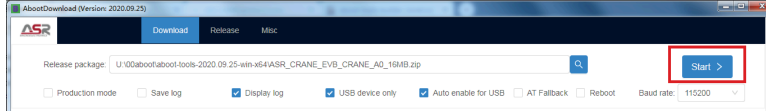


Figure 6.4. start Abboot engine

If board is not in USB download mode (no AT command or key pressed), the screen is shown like below.

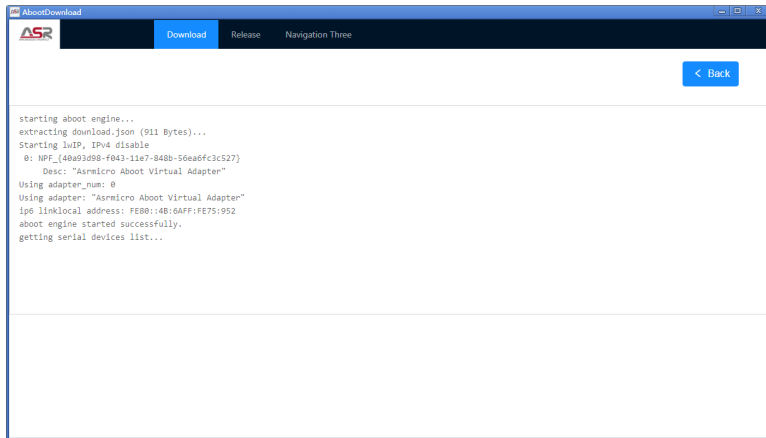


Figure 6.5. Abboot engine is started without USB download mode

5) Set target in USB burn program mode

If target is in USB burn program mode via **XDB** or press **USB_DOWNLOAD** key + **RST** key or **AT fallback box** is checked, the USB ACM com will be found in device manager and the USB com will be shown in Abboot download screen.

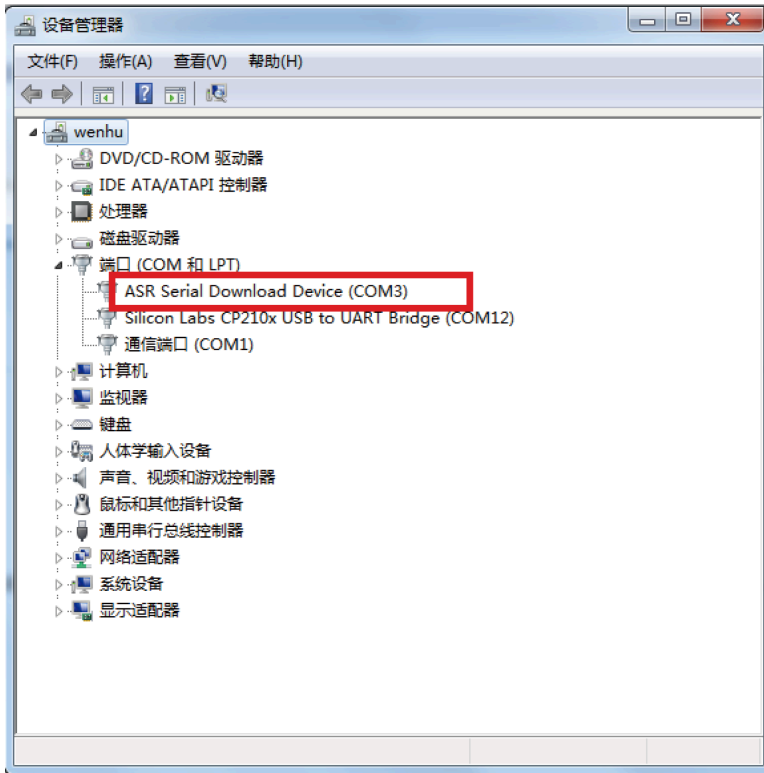


Figure 6.6. target enter USB download mode

Since Auto enable for USB is enabled. If target connects in USB burn program mode, About engine will enable USB port to execute commands for burn images. Progress bar value begins from 0% to 100%, show as below:

- Status is connecting, progress bar value is 0%.

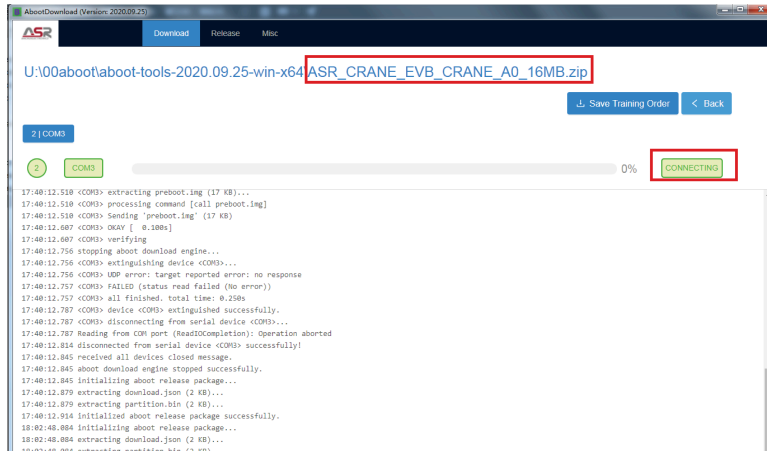


Figure 6.7. status is connecting

- Status is running, progress bar value is 2%.

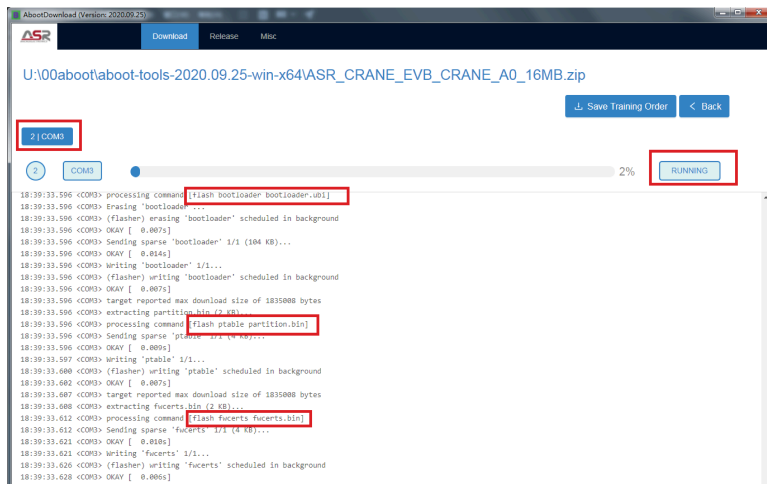


Figure 6.8. status is running, process is 2%

- Status is succeeded, progress bar value is 100%. Download finished.

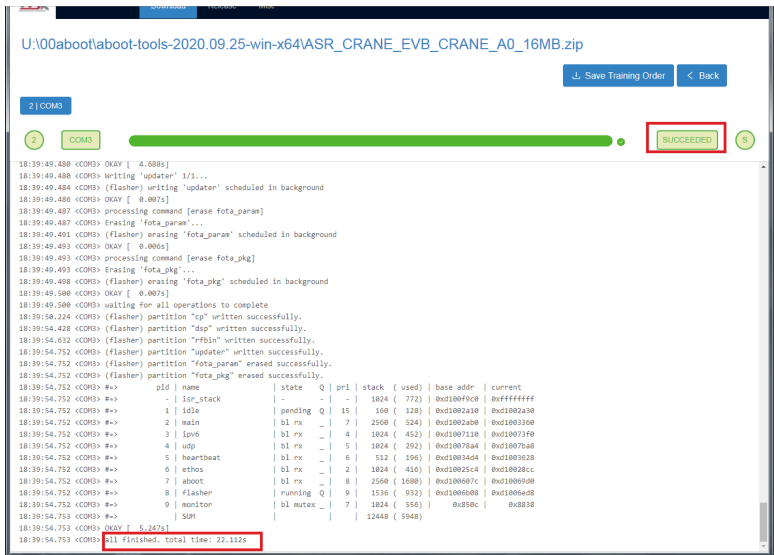


Figure 6.9. status is completed, process is 100%

6) Save training order



save training order is necessary when downloading several devices with production mode in factory.

If factory customer wants to save the training order for all connected usb ports. click "**Save Training Order**" button after download process is completed.

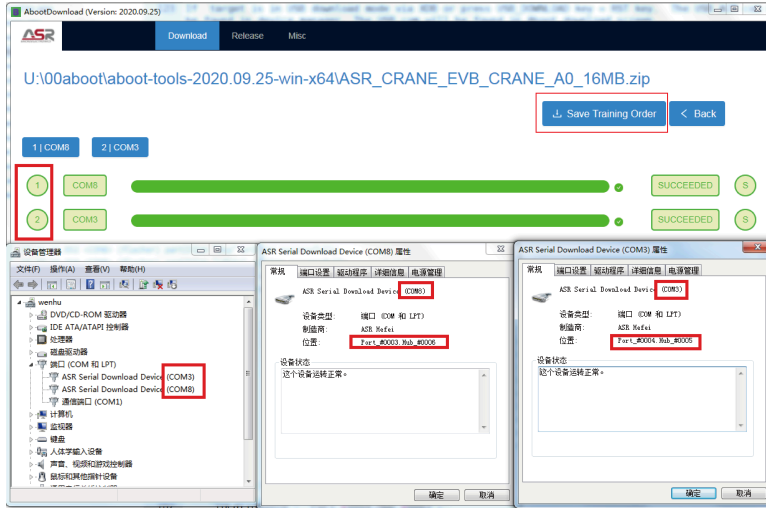


Figure 6.10. save training order

A json file named **training.json** will be written in the tool's folder. For example:

training.json.

```
[
  {
    "locationInfo": "Port_#0003.Hub_#0006",
    "order": 1
  },
  {
    "locationInfo": "Port_#0004.Hub_#0005",
    "order": 2
  }
]
```

If training.json exists, tool will read the training orders from json file when tool is opened. The order of usb port's com is fixed each time, same as the order saved in **training.json**.

It is useful when downloading with several usb devices in factory.

6.4. Download release package by UART

- If download via UART, uncheck USB device only button.

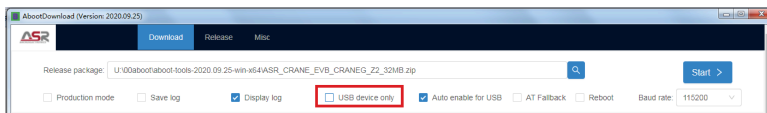


Figure 6.11. uncheck USB device only button

- press **UART_DOWND** key + **RST** key, target enters UART downloading mode.

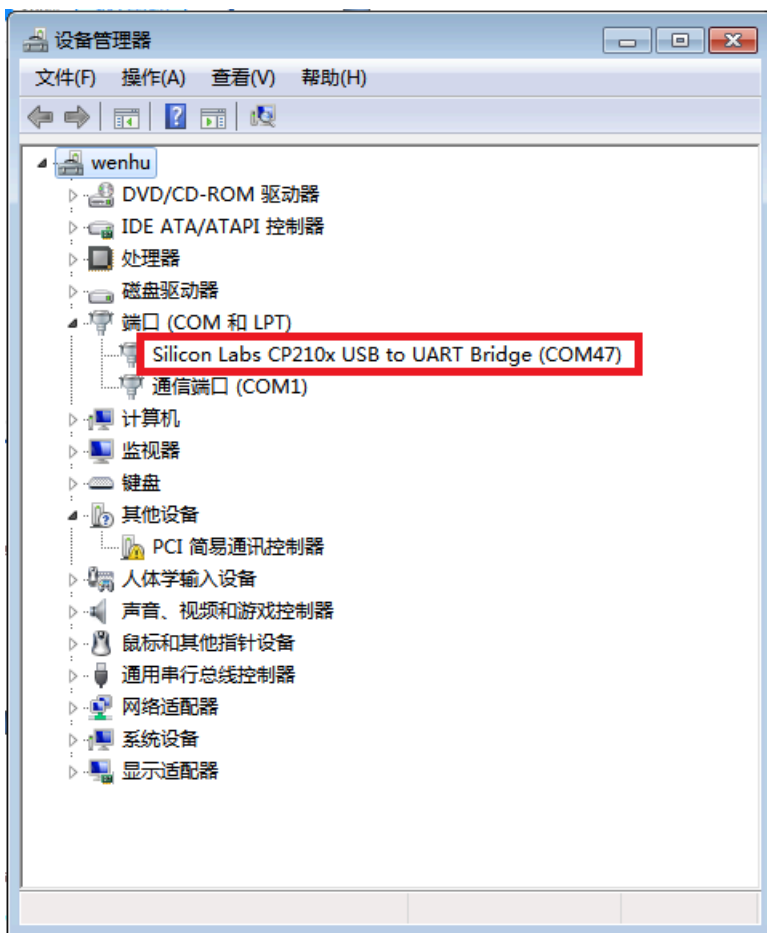


Figure 6.12. UART-to-Serial com

- Click **Start >** button, find **COM47** in device manager. Click **0|COM47** to enable Abboot engine for com47.

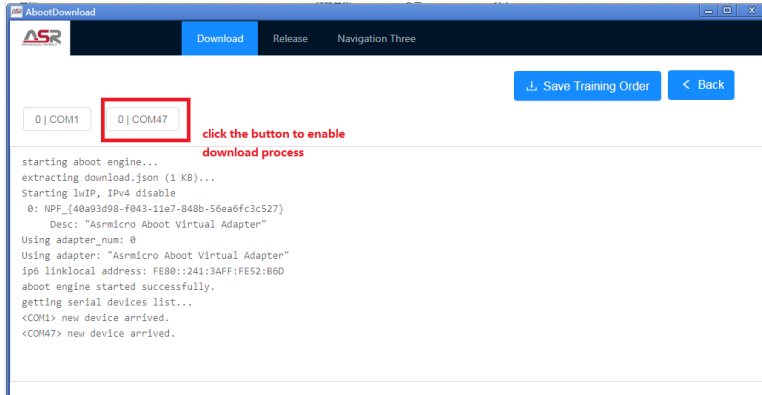


Figure 6.13. UART serial com shown in UART burning program mode

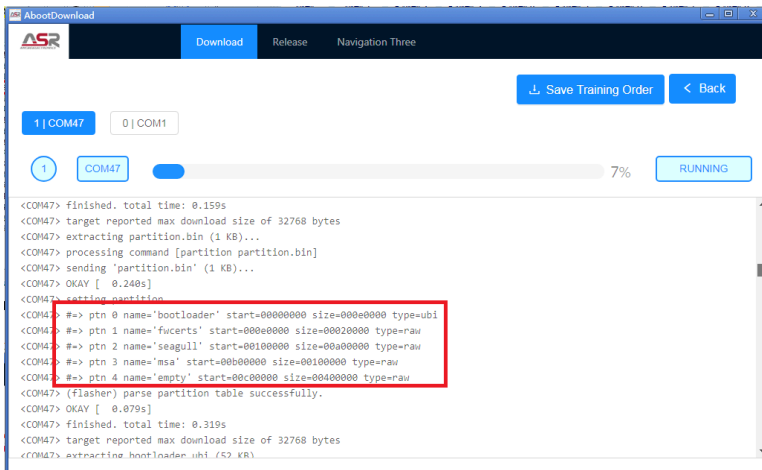


Figure 6.14. burning program with UART

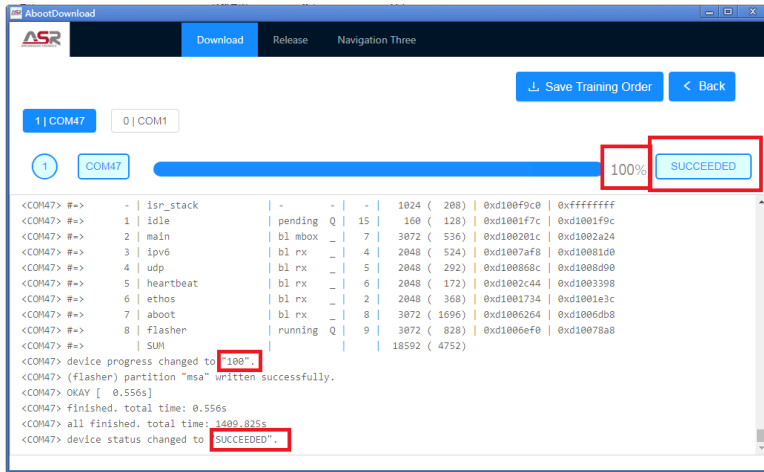


Figure 6.15. burn program finished with UART

6.5. Trace for burn program

Burn program trace.

```

17:20:12.933 starting about download engine...
17:20:12.933 extracting download.json (2 KB)...
17:20:12.933 download engine running in upgrade mode!
17:20:12.933 about download engine started successfully.
17:20:12.933 getting serial devices list...
17:20:13.025 <COM8> new device arrived.
17:20:13.025 enabling device <COM8> into downloading mode...
17:20:13.025 device <COM8> enabled successfully.
17:20:13.025 connecting to serial device <COM8>...
17:20:17.152 Open (SetCommState): Access denied
17:20:17.152 connected to serial device <COM8> failed!
17:20:17.192 <COM8> device offline
17:20:17.264 <COM3> device online
17:20:17.264 connecting to serial device <COM3>...
17:20:17.264 <COM3> connected to serial device <COM3> successfully!
17:20:17.264 <COM3> starting to fire device <COM3>...
17:20:17.264 <COM3> device <COM3> fired successfully.
17:20:18.211 <COM3> device <COM3> is ready to be enabled manually
17:20:18.269 <COM3> processing command [get bootrom info]
17:20:18.269 <COM3> processing command [require product arom|arom-tiny]
17:20:18.269 <COM3> checking product
17:20:18.269 <COM3> OKAY [ 0.005s]
17:20:18.269 <COM3> processing command [require version 0.5]
17:20:18.269 <COM3> checking version

```



```

17:20:18.269 <COM3> OKAY [ 0.006s]
17:20:18.269 <COM3> processing command [require version-bootrom
2019.01.15]❶
17:20:18.269 <COM3> Checking version-bootrom
17:20:18.270 <COM3> OKAY [ 0.006s]
17:20:18.270 <COM3> processing command [progress weight 10398823]
17:20:18.270 <COM3> setting total progress
17:20:18.270 <COM3> OKAY [ 0.005s]
17:20:18.271 <COM3> target reported max download size of 64768 bytes
17:20:18.271 <COM3> extracting preboot.img (17 KB)...
17:20:18.272 <COM3> processing command [call preboot.img]❷
17:20:18.272 <COM3> Sending 'preboot.img' (17 KB)
17:20:18.419 <COM3> OKAY [ 0.147s]
17:20:18.419 <COM3> verifying
17:20:18.648 <COM3> OKAY [ 0.229s]
17:20:18.648 <COM3> calling
17:20:18.652 <COM3> #=> [PRI : Preboot ] Executing preboot
application...
17:20:18.653 <COM3> #=> [PRI : Preboot ] Preboot version: 2020.09.22
17:20:18.655 <COM3> #=> [INFO: Preboot ] fuse: sbe = 0,
sys_boot_ctrl=0x8001
17:20:18.655 <COM3> #=> [PRI : Preboot ] ### Non-trusted boot mode.
###
17:20:18.656 <COM3> #=> [INFO: Preboot ] usb phy reg 28 0x1811 ->
0x3811
17:20:18.657 <COM3> #=> [PRI : Preboot ] Found PMIC with Id: 0x3b
17:20:18.658 <COM3> #=> [INFO: Preboot ] Disabling PMIC watchdog
17:20:18.659 <COM3> #=> [INFO: Preboot ] set prcharge current to 150mA
17:20:18.660 <COM3> #=> [INFO: Preboot ]
PM813_PRE_CHARGE_CURRENT_SETTING_REG: [0x21] = 0xf
17:20:18.662 <COM3> #=> [INFO: Preboot ] PMIC watchdog is disabled
17:20:18.663 <COM3> #=> [tRI : Preboot ] power_up_reason=0x40.
17:20:18.663 <COM3> #=> [INFO: Preboot ] Disable PMIC fault wakeup.
17:20:18.665 <COM3> #=> [INFO: Preboot ] CRANE: set PM813_DVC_SET_REG
= 0x0, set bit7 to 0.
17:20:18.667 <COM3> #=> [INFO: Preboot ] CRANE: set cpu core voltage
PM813_BUCK_VOLTAGE_SET_REG = 0xa4
17:20:18.668 <COM3> #=> [INFO: Preboot ] Freq change done: CR5 416MHZ
-> 624MHZ, AXI 156MHZ -> 208MHZ
17:20:18.669 <COM3> #=> [INFO: Psram ] PSRAM PHY frequency changed
to 78
17:20:18.669 <COM3> #=> [PRI : Psram ] [PSRAM] psram_init_crane_a0.
17:20:18.670 <COM3> #=> [INFO: Psram ] #####
17:20:18.670 <COM3> #=> [INFO: Psram ] Version ID : 0
17:20:18.670 <COM3> #=> [PRI : Psram ] AP 16MB

```

```

17:20:18.671 <COM3> #=> [PRI : Psram      ] No embedded flash.
17:20:18.728 <COM3> #=> [INFO: Psram      ] #####
17:20:18.728 <COM3> #=> [INFO: Psram      ] #####
17:20:18.729 <COM3> #=> [INFO: Psram      ] MP DEVICE
17:20:18.729 <COM3> #=> [INFO: Psram      ] MR[2] VALUE = 0x5
17:20:18.729 <COM3> #=> [INFO: Psram      ] #####
17:20:18.729 <COM3> #=> [INFO: Psram      ] mmap psram address spaces:
    reg[0xc0100004] = 0x7e080001
17:20:18.729 <COM3> #=> [INFO: Psram      ] psram cache
    enabled !!! :cache size=[128B], @[0xc0104000]=[0x000A3B11]
17:20:18.729 <COM3> #=> [INFO: Psram      ] Do Globe1 Reset
17:20:18.729 <COM3> #=> [INFO: Psram      ] Enable fast_miss_acc
17:20:18.729 <COM3> #=> [INFO: Psram      ] RB: MR0x0_CS0:
    @[0xc0108034]=[0x00000000]
17:20:18.729 <COM3> #=> [INFO: Psram      ] RB: MR0x0_CS0:
    @[0xc0108038]=[0x8D138D13]
17:20:18.729 <COM3> #=> [INFO: Psram      ] RB: MR0x0_CS1:
    @[0xc0108034]=[0x00800000]
17:20:18.729 <COM3> #=> [INFO: Psram      ] RB: MR0x0_CS1:
    @[0xc0108038]=[0x8D138D13]
17:20:18.729 <COM3> #=> [INFO: Psram      ] RB: MR0x1_CS0:
    @[0xc0108034]=[0x00000001]
17:20:18.729 <COM3> #=> [INFO: Psram      ] RB: MR0x1_CS0:
    @[0xc0108038]=[0x958D958D]
17:20:18.729 <COM3> #=> [INFO: Psram      ] RB: MR0x1_CS1:
    @[0xc0108034]=[0x00800001]
17:20:18.729 <COM3> #=> [INFO: Psram      ] RB: MR0x1_CS1:
    @[0xc0108038]=[0x958D958D]
17:20:18.729 <COM3> #=> [INFO: Psram      ] PSRAM PHY frequency changed
    to 175
17:20:18.729 <COM3> #=> [INFO: Preboot   ] set fip image load address
    spaces to 0x7e000000
17:20:18.729 <COM3> OKAY [ 0.033s]
17:20:18.729 <COM3> target reported max download size of 64768 bytes
17:20:18.729 <COM3> extracting flasher.img (33 KB)...
17:20:18.730 <COM3> processing command [call flasher.img]
17:20:18.730 <COM3> Sending 'flasher.img' (33 KB)
17:20:18.875 <COM3> OKAY [ 0.186s]
17:20:18.875 <COM3> verifying
17:20:19.037 <COM3> OKAY [ 0.162s]
17:20:19.037 <COM3> calling
17:20:19.039 <COM3> #=> Executing flasher application...③
17:20:19.040 <COM3> #=> Flasher version: 2020.09.22
17:20:19.041 <COM3> #=> (monitor_thread) monitor thread started
17:20:19.042 <COM3> OKAY [ 0.005s]

```

```
17:20:19.046 <COM3> target reported max download size of 1835008 bytes
17:20:19.047 <COM3> extracting flashinfo.bin (936 Bytes)...
17:20:19.048 <COM3> processing command [partition flashinfo.bin]
17:20:19.048 <COM3> Sending 'flashinfo.bin' (0 KB)
17:20:19.060 <COM3> OKAY [ 0.012s]
17:20:19.061 <COM3> setting partition
17:20:19.063 <COM3> #=> QSPI clock configured at 104 MHz
17:20:19.064 <COM3> #=> Manufacturer ID: 0xC8, Device ID: 0x6018
17:20:19.065 <COM3> #=> Found a known spi flash device "GD25LQ128D"
17:20:19.065 <COM3> #=> qspi flash, cancel_block_protected: status2 =
    0x2, status1 = 0x0
17:20:19.066 <COM3> OKAY [ 0.006s]
17:20:19.071 <COM3> target reported max download size of 1835008 bytes
17:20:19.072 <COM3> extracting partition.bin (2 KB)...
17:20:19.072 <COM3> processing command [partition partition.bin]
17:20:19.073 <COM3> Sending 'partition.bin' (2 KB)
17:20:19.091 <COM3> OKAY [ 0.019s]
17:20:19.091 <COM3> setting partition4
17:20:19.095 <COM3> #=> ptn 0 name='all' start=00000000 size=01000000
    type=group depth=0
17:20:19.096 <COM3> #=> ptn 1 name='external' start=00000000
    size=01000000 type=flash depth=1
17:20:19.163 <COM3> #=> ptn 2 name='bootloader' start=00000000
    size=00024000 type=ubi depth=2
17:20:19.163 <COM3> #=> ptn 3 name='system' start=00024000 size=004d9000
    type=part depth=2
17:20:19.164 <COM3> #=> ptn 4 name='ptable' start=00024000 size=00001000
    type=raw depth=3
17:20:19.164 <COM3> #=> ptn 5 name='fwcerts' start=00025000
    size=00003000 type=raw depth=3
17:20:19.164 <COM3> #=> ptn 6 name='rd' start=00028000 size=00010000
    type=raw depth=3
17:20:19.164 <COM3> #=> ptn 7 name='apn' start=00038000 size=00008000
    type=raw depth=3
17:20:19.164 <COM3> #=> ptn 8 name='cp' start=00040000 size=003c4000
    type=raw depth=3
17:20:19.164 <COM3> #=> ptn 9 name='dsp' start=00404000 size=000f4000
    type=raw depth=3
17:20:19.164 <COM3> #=> ptn 10 name='rfbin' start=004f8000 size=00005000
    type=raw depth=3
17:20:19.164 <COM3> #=> ptn 11 name='reserved' start=004fd000
    size=008e0000 type=raw depth=2
17:20:19.164 <COM3> #=> ptn 12 name='fota_param' start=00ddd000
    size=00001000 type=raw depth=2
```

```
17:20:19.164 <COM3> #=> ptn 13 name='updater' start=00dde000
size=00001000 type=raw depth=2
17:20:19.164 <COM3> #=> ptn 14 name='fota_pkg' start=00ddf000
size=00001000 type=raw depth=2
17:20:19.164 <COM3> #=> ptn 15 name='nvm' start=00de0000 size=00200000
type=raw depth=2
17:20:19.164 <COM3> #=> ptn 16 name='factory' start=00fe0000
size=00020000 type=part depth=2
17:20:19.164 <COM3> #=> ptn 17 name='factory_a' start=00fe0000
size=00010000 type=raw depth=3
17:20:19.164 <COM3> #=> ptn 18 name='factory_b' start=00ff0000
size=00010000 type=raw depth=3
17:20:19.165 <COM3> (flasher) parse partition table successfully.
17:20:19.165 <COM3> OKAY [ 0.020s]
17:20:19.165 <COM3> target reported max download size of 1835008 bytes
17:20:19.165 <COM3> extracting bootloader.ubi (104 KB)...
17:20:19.165 <COM3> processing command [flash bootloader
bootloader.ubi]⑤
17:20:19.165 <COM3> Erasing 'bootloader'...
17:20:19.165 <COM3> (flasher) erasing 'bootloader' scheduled in
background
17:20:19.165 <COM3> OKAY [ 0.007s]
17:20:19.165 <COM3> Sending sparse 'bootloader' 1/1 (104 KB)...
17:20:19.165 <COM3> OKAY [ 0.014s]
17:20:19.165 <COM3> Writing 'bootloader' 1/1...
17:20:19.165 <COM3> (flasher) writing 'bootloader' scheduled in
background
17:20:19.165 <COM3> OKAY [ 0.006s]
17:20:19.165 <COM3> target reported max download size of 1835008 bytes
17:20:19.165 <COM3> extracting partition.bin (2 KB)...
17:20:19.165 <COM3> processing command [flash ptable partition.bin]⑥
17:20:19.165 <COM3> Sending sparse 'ptable' 1/1 (4 KB)...
17:20:19.172 <COM3> OKAY [ 0.010s]
17:20:19.172 <COM3> Writing 'ptable' 1/1...
17:20:19.176 <COM3> (flasher) writing 'ptable' scheduled in background
17:20:19.178 <COM3> OKAY [ 0.006s]
17:20:19.183 <COM3> target reported max download size of 1835008 bytes
17:20:19.183 <COM3> extracting fwcerts.bin (2 KB)...
17:20:19.188 <COM3> processing command [flash fwcerts fwcerts.bin]⑦
17:20:19.188 <COM3> Sending sparse 'fwcerts' 1/1 (4 KB)...
17:20:19.197 <COM3> OKAY [ 0.009s]
17:20:19.197 <COM3> Writing 'fwcerts' 1/1...
17:20:19.202 <COM3> (flasher) writing 'fwcerts' scheduled in background
17:20:19.204 <COM3> OKAY [ 0.007s]
17:20:19.208 <COM3> target reported max download size of 1835008 bytes
```

```
17:20:19.209 <COM3> extracting ReliableData.bin (64 KB)...
17:20:19.214 <COM3> processing command [flash rd ReliableData.bin]8
17:20:19.214 <COM3> Sending sparse 'rd' 1/1 (4 KB)...
17:20:19.223 <COM3> OKAY [ 0.009s]
17:20:19.260 <COM3> writing 'rd' 1/1...
17:20:19.260 <COM3> (flasher) writing 'rd' scheduled in background
17:20:19.260 <COM3> OKAY [ 0.007s]
17:20:19.260 <COM3> target reported max download size of 1835008 bytes
17:20:19.260 <COM3> extracting apn.bin (11 KB)...
17:20:19.260 <COM3> processing command [flash apn apn.bin]9
17:20:19.261 <COM3> Sending sparse 'apn' 1/1 (12 KB)...
17:20:20.211 <COM3> (flasher) partition "bootloader" erased
successfully.
17:20:20.473 <COM3> (flasher) partition "bootloader" written
successfully.
17:20:20.480 <COM3> OKAY [ 1.241s]
17:20:20.481 <COM3> writing 'apn' 1/1...
17:20:20.485 <COM3> (flasher) writing 'apn' scheduled in background
17:20:20.487 <COM3> OKAY [ 0.007s]
17:20:20.491 <COM3> target reported max download size of 1835008 bytes
17:20:20.492 <COM3> extracting cp.bin (3 MB)...
17:20:20.550 <COM3> processing command [flash cp cp.bin]10
17:20:20.550 <COM3> Sending sparse 'cp' 1/3 (1788 KB)...
17:20:20.562 <COM3> (flasher) partition "ptable" written successfully.
17:20:20.650 <COM3> OKAY [ 0.100s]
17:20:20.650 <COM3> writing 'cp' 1/3...
17:20:20.655 <COM3> (flasher) writing 'cp' scheduled in background
17:20:20.657 <COM3> OKAY [ 0.007s]
17:20:20.657 <COM3> Sending sparse 'cp' 2/3 (1788 KB)...
17:20:20.685 <COM3> (flasher) partition "fwcerts" written successfully.
17:20:20.773 <COM3> OKAY [ 0.116s]
17:20:20.773 <COM3> writing 'cp' 2/3...
17:20:20.778 <COM3> (flasher) writing 'cp' scheduled in background
17:20:20.780 <COM3> OKAY [ 0.007s]
17:20:20.780 <COM3> Sending sparse 'cp' 3/3 (280 KB)...
17:20:21.193 <COM3> (flasher) partition "rd" written successfully.
17:20:21.213 <COM3> OKAY [ 0.433s]
17:20:21.213 <COM3> writing 'cp' 3/3...
17:20:21.218 <COM3> (flasher) writing 'cp' scheduled in background
17:20:21.220 <COM3> OKAY [ 0.007s]
17:20:21.224 <COM3> target reported max download size of 1835008 bytes
17:20:21.225 <COM3> extracting dsp.bin (972 KB)...
17:20:21.241 <COM3> processing command [flash dsp dsp.bin]
17:20:21.241 <COM3> Sending sparse 'dsp' 1/1 (976 KB)...
17:20:21.572 <COM3> (flasher) partition "apn" written successfully.
```

```

17:20:21.626 <COM3> OKAY [ 0.385s]
17:20:21.627 <COM3> writing 'dsp' 1/1...
17:20:21.631 <COM3> (flasher) writing 'dsp' scheduled in background
17:20:21.633 <COM3> OKAY [ 0.007s]
17:20:21.638 <COM3> target reported max download size of 1835008 bytes
17:20:21.638 <COM3> extracting rf.bin (20 KB)...
17:20:21.642 <COM3> processing command [flash rfbin rf.bin]
17:20:21.643 <COM3> Sending sparse 'rfbin' 1/1 (20 KB)...
17:20:40.014 <COM3> (flasher) partition "cp" written successfully.
17:20:40.022 <COM3> OKAY [ 18.381s]
17:20:40.022 <COM3> writing 'rfbin' 1/1...
17:20:40.026 <COM3> (flasher) writing 'rfbin' scheduled in background
17:20:40.028 <COM3> OKAY [ 0.007s]
17:20:40.034 <COM3> target reported max download size of 1835008 bytes
17:20:40.034 <COM3> extracting updater.bin (4 KB)...
17:20:40.039 <COM3> processing command [flash updater updater.bin]
17:20:40.039 <COM3> Sending sparse 'updater' 1/1 (0 KB)...
17:20:44.746 <COM3> (flasher) partition "cp" written successfully.
17:20:44.751 <COM3> OKAY [ 4.705s]
17:20:44.751 <COM3> writing 'updater' 1/1...
17:20:44.755 <COM3> (flasher) writing 'updater' scheduled in background
17:20:44.757 <COM3> OKAY [ 0.007s]
17:20:44.758 <COM3> processing command [erase fota_param]
17:20:44.758 <COM3> Erasing 'fota_param'...
17:20:44.763 <COM3> (flasher) erasing 'fota_param' scheduled in
background
17:20:44.764 <COM3> OKAY [ 0.007s]
17:20:44.764 <COM3> processing command [erase fota_pkg]
17:20:44.764 <COM3> Erasing 'fota_pkg'...
17:20:44.769 <COM3> (flasher) erasing 'fota_pkg' scheduled in background
17:20:44.771 <COM3> OKAY [ 0.006s]
17:20:44.771 <COM3> waiting for all operations to complete
17:20:45.499 <COM3> (flasher) partition "cp" written successfully.
17:20:52.614 <COM3> (flasher) partition "dsp" written successfully.
17:20:52.989 <COM3> (flasher) partition "rfbin" written successfully.
17:20:53.119 <COM3> (flasher) partition "updater" written successfully.
17:20:53.119 <COM3> (flasher) partition "fota_param" erased
successfully.
17:20:53.119 <COM3> (flasher) partition "fota_pkg" erased successfully.
17:20:53.119 <COM3> #=> pid | name | state Q | pri |
stack ( used) | base addr | current
17:20:53.119 <COM3> #=> - | isr_stack | - - | - |
1024 ( 772) | 0xd100f9c0 | 0xffffffff
17:20:53.119 <COM3> #=> 1 | idle | pending Q | 15 |
160 ( 128) | 0xd1002a10 | 0xd1002a30

```

```

17:20:53.122 <COM3> #=> 2 | main | b1 rx _ | 7 |
2560 ( 524) | 0xd1002ab0 | 0xd1003360
17:20:53.122 <COM3> #=> 3 | ipv6 | b1 rx _ | 4 |
1024 ( 452) | 0xd1007110 | 0xd10073f0
17:20:53.122 <COM3> #=> 4 | udp | b1 rx _ | 5 |
1024 ( 292) | 0xd10078a4 | 0xd1007ba8
17:20:53.123 <COM3> #=> 5 | heartbeat | b1 rx _ | 6 |
512 ( 196) | 0xd10034d4 | 0xd1003628
17:20:53.124 <COM3> #=> 6 | ethos | b1 rx _ | 2 |
1024 ( 416) | 0xd10025c4 | 0xd10028cc
17:20:53.125 <COM3> #=> 7 | aboot | b1 rx _ | 8 |
2560 ( 1680) | 0xd100607c | 0xd10069d0
17:20:53.126 <COM3> #=> 8 | flasher | running Q | 9 |
1536 ( 932) | 0xd1006b08 | 0xd1006f58
17:20:53.127 <COM3> #=> 9 | monitor | b1 mutex _ | 7 |
1024 ( 556) | 0x850c | 0x8838
17:20:53.127 <COM3> #=> | SUM | | |
12448 ( 5948)
17:20:53.128 <COM3> OKAY [ 8.359s]
17:20:53.128 <COM3> all finished. total time: 34.904s11
17:21:10.306 stopping aboot download engine...
17:21:10.307 <COM3> extinguishing device <COM3>...
17:21:10.307 <COM3> device <COM3> extinguished successfully.
17:21:10.307 <COM3> disconnecting from serial device <COM3>...
17:21:10.307 Reading from COM port (ReadIOCompletion): Operation aborted
17:21:10.346 disconnected from serial device <COM3> successfully!
17:21:10.374 received all devices closed message.
17:21:10.374 aboot download engine stopped successfully.
17:21:10.374 initializing aboot release package...
17:21:10.375 extracting download.json (2 KB)...
17:21:10.375 extracting partition.bin (2 KB)...
17:21:10.455 initialized aboot release package successfully.

```

- ❶ bootrom version for the product is 2019.01.15.
- ❷ call preboot.bin.
- ❸ call flasher.bin.
- ❹ show partition table.
- ❺ download process is finished

6.6. Trace for boot

- If usb is connected, bootloader info log will be printed.

- If usb is unconnected when booting, bootloader info log will not be printed.
- preboot's INFO log will not be printed when booting. It only be printed when burning program mode.

When booting, preboot's info log will not be printed. If user what to check preboot's info log, check log when downloading process.

Bootting trace with usb is unconnected.

```
[18:14:14.836]收←◆CHIP_ID: 0x6731, REV_ID: 0xA0❶
PLATFORM: CRANE (SILICON)
CRANE hardware initialization complete.

AROM! (Version: 2019.01.15)❷
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"❸
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 17992 bytes
BL1: Trying to load and verify BL2 image from volume preboot...

[18:14:15.097]收←◆BL1: Okay.
[PRI : Preboot   ] Executing preboot application...
[PRI : Preboot   ] Preboot version: 2020.09.22❹
[PRI : Preboot   ] ### Non-trusted boot mode. ###
[PRI : Preboot   ] Found PMIC with Id: 0x3b❺
[PRI : Preboot   ] power_up_reason=0x1.❻
[PRI : Psram     ] [PSRAM] psram_init_crane_a0.❼
[PRI : Psram     ] AP 16MB❽
[PRI : Psram     ] No embedded flash.❾

[18:14:15.151]收←◆Found bootloader volume, size is 66936 bytes
BL1: Trying to load and verify BL2 image from volume bootloader...

[18:14:15.307]收←◆BL1: Okay.

[18:14:15.389]收←◆Starting BOOT2 (Version: 2020.09.22)❿
B33 START
```



```

[18:14:15.434]收←◆LastPowerOff: 0x4 @0xE5, 0x0 @0xE6

[18:14:15.459]收←◆SECBOOT_SUPPORT:NON-SECBOOT
ver:boot33.bin 20200521_ver111
PowerOn: 0x1
OnkeyPowerOnCheck

[18:14:16.498]收←◆NO FOTA FLAG
PS:LTEGSM
IMG:CRANE_DS_LTEGSM_DKB_Z2_A0_XIP_LWIP_MODULEONLY_16M16M
LDT_RW_CPZ_INFO
[RW_CPZ_1][DDR_RW_][7ea00000][80051508][00023a10][80051508]
[RW_CPZ_3][PS_NCAH][7ef20000][80074f18][000052c0][80058d96]
[RW_CPZ_4][USBNAH][7ef92400][8007a1d8][00000244][8005928c]
[RW_CPZ_5][CODE_PS][7e200000][8007a41c][0054320c][800593ed]
Region CPZ struct detected from LDT
decompress [ DDR_RW_] from [80051508] to [7ea00000]

[18:14:16.534]收←◆decompress [ PS_NCAH] from [80058d96] to [7ef20000]
decompress [ USBNAH] from [8005928c] to [7ef92400]
decompress [ CODE_PS] from [800593ed] to [7e200000]

[18:14:18.114]收←◆stop decompress as no further RW_CPZ_ detected
logo.bin not exist in PTB
B33 DONE
PC:0x80040000

[18:14:18.183]收←◆Cinit 112
srand(1)
CRANE_CUST_VER_INFO = N/A
file:X:\hop\BSP\src\cinit1.c,function:Cinit1,line:675

```

- ❶ chip id is CRANE A0.
- ❷ bootrom version is 2019.01.15.
- ❸ QSPI flash is GD25LQ128D.
- ❹ Preboot version.
- ❺ PMIC with Id: 0x3b, is PM813 A3.
- ❻ power_up_reason read from PMIC.
- ❼ PSRAM driver is crane_a0.
- ❽ PSRAM type is AP 16MB. It is read from fuse bits.

- 9 whether has internal SPI flash. CRANE has no embedded flash. it is read from fuse bits.
- 10 BOOT2 version.
- 11 Start BOOT33.
- 12 Start CP.

upload function

The tool supplies function that data can be uploaded from external/internal flash to PC, saved as binary files. It supplies user a debug method for consistency verification manually.

1. Uncompressed source files from released normal package(eg: ASR_CRANE_EVB_CRANEG_Z2_32MB.zip)
2. Release a package for upload fuction.
3. Burn program with upload package. the dump data is written in binary files.
4. Compare uploaded binary files with source files. The data must be same.

When burning program process, bootloader has mechanism for consistency verification when downloding data from PC to psram and writing data from psram to flash.

If flash's data is updated in future by other reason, user can use the upload function to dump data from flash to check.

7.1. Generate upload package

Two methods are supplied to generate upload package:

1. arelase.exe(cmdline)
2. Aboot.exe(GUI)

7.1.1. Arelease.exe

In cmdline mode, enter arelease.exe, the help information is shown as below:

execute arelease.exe in windows's cmd.exe.

```
about-tools-2020.09.25-win-x64>arelease.exe
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=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_16MB❶
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 the upload command example.

1) Get products list

execute arelease.exe -l in windows's cmd.exe.

```
about-tools-2020.09.25-win-x64>arelease.exe -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+8MB
0.7		CRANE_A0_16MB
0.8		CRANE_A0_32MB
1.	ASR_JACANA_EVB	
1.0		JACANA_EVB

2) Generate upload package zip file

For example (product is CRANE_A0_16MB):

copy the command to in windows's cmd.exe, execute it. The output file ASR_CRANE_EVB_CRANE_A0_16MB_UPLOAD.zip will be generated under the tool's folder.

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

execute arelease.exe in windows's cmd.exe.

```
about-tools-2020.09.25-win-x64>arelease.exe -c . -g --upload-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 ...
```

```
Generating fip image "preboot_2019.01.15" with id "preboot.img" ...
```

```
Opening image "preboot_bin_2019.01.15" from "U:\00about\about-
tools-2020.09.25-win-x64\images\2019.01.15\preboot.bin" ...
```

```
Done.
```

```
Done.
```

```
Generating fip image "flasher_2019.01.15" with id "flasher.img" ...
```

```
Opening image "flasher_bin_2019.01.15" from "U:\00aboot\aboot-
tools-2020.09.25-win-x64\images\arom.bin" ...
Done.
Done.
Generating fip image "preboot_upload" with id "preboot.img" ...
Opening image "preboot_bin_upload" from "U:\00aboot\aboot-
tools-2020.09.25-win-x64\images\preboot.bin" ...
Done.
Done.
Generating fip image "flasher_upload" with id "flasher.img" ...
Opening image "flasher_bin_upload" from "U:\00aboot\aboot-
tools-2020.09.25-win-x64\images\flasher.bin" ...
Done.
Done.
Generating finf image "flashinfo" with id "flashinfo.bin" ...
Done.
Generating aptb image "partition" with id "partition.bin" ...
Done.
Done.
Calculating total progress weight ...
Done.
Generating download commands ...
Done.
Generating target release package ...
Done.
Release package generated successfully!
```

7.1.2. Adownload.exe

1) Get products list

- open Release page, the products list is shown, see **Variant** column.
- Click **Package Type** selection frame, change string element from **Normal** to **"Upload"**.
- select the product id for generate a upload package. eg: CRANE_A0_16MB

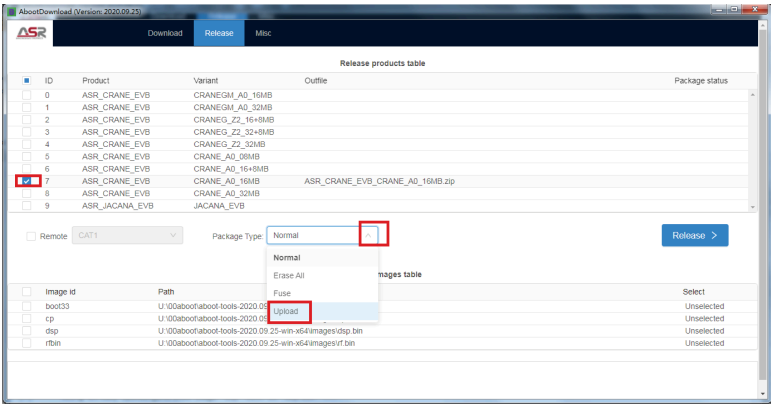


Figure 7.1. select upload function

- After selected, the default output file name is changed, has the key words **UPLOAD**. eg: **ASR_CRANE_EVB_CRANE_A0_16MB_UPLOAD.zip**. If user want to updated output name, click name and enter new words manually.

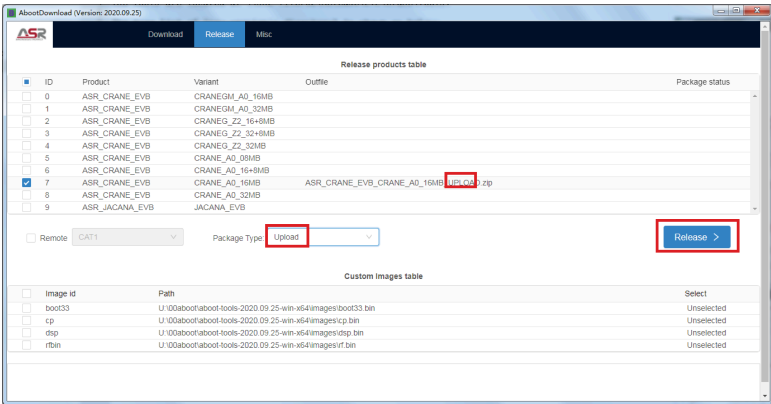


Figure 7.2. upload function is selected

- Click **Release** Button, the key words are shown in log console. output zip file is generated in the folder of tool.

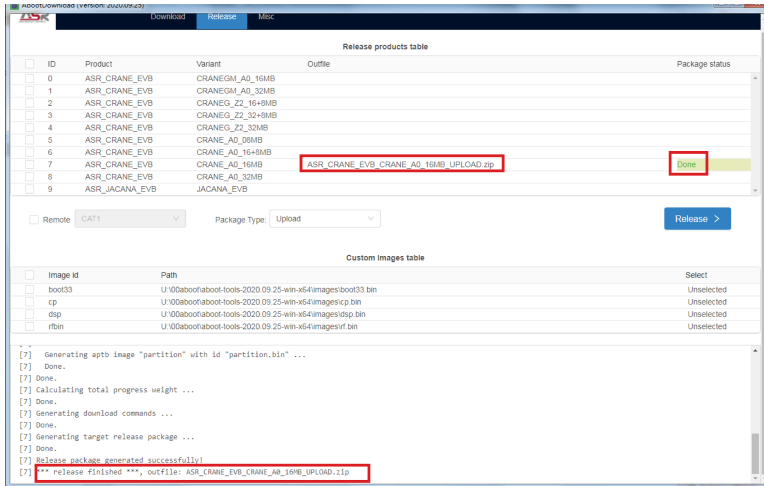


Figure 7.3. upload package is generated

7.2. Burn upload package

Two methods are supplied to burn upload package:

1. Adownload.exe(cmdline)
2. Aboot.exe(GUI)

7.2.1. Adownload.exe

1) In cmdline mode, enter **adownload.exe** in windows cmd.exe, the help information is shown as below:

[new version's directory]\adownload.exe help information.

about-tools-2020.09.25-win-x64>adownload.exe

Arsmicro 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 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,
1686400)
-m, --production  Running in mass production mode, default
is upgrade mode
-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.zip❶
adownload.exe -p COM1 -a -s 115200 aboot.zip

```

❶ burn upload package can follow the command. It is good to add -q.

2) execute burn program command:

Need add -q, if adownload.exe is finished, exe will be automatically exit.

```

aboot-tools-2020.09.25-win-x64> adownload.exe -u -a -s -q 115200
ASR_CRANE_EVB_CRANE_A0_16MB_UPLOAD.zip

```

7.2.2. Abboot.exe

- Open **Burn** page, the latest zip is loaded automatically. If the loaded zip by automatically is not what you want(eg: **ASR_CRANE_EVB_CRANE_A0_16MB_UPLOAD.zip**), you can select upload package zip file by click the hand magnifier Button.

Information of upload package:

1. Download Command Table:

It shows that a upload command will be executed.

1. Partition table:

It shows the flash space for dump data.

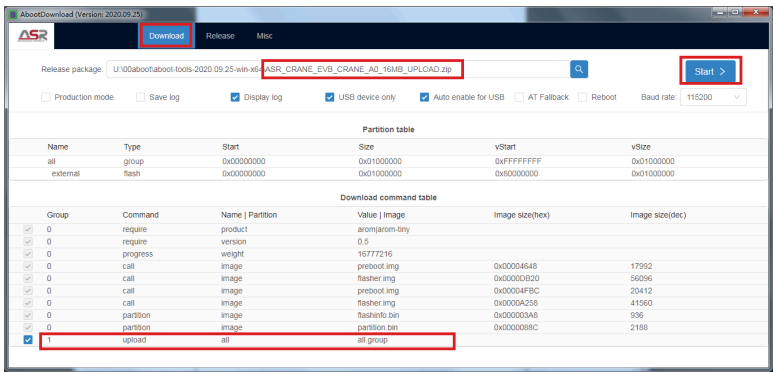


Figure 7.4. upload package information

- Click **Start** Button to execute commands of upload function. User can check which command is executed from log console.

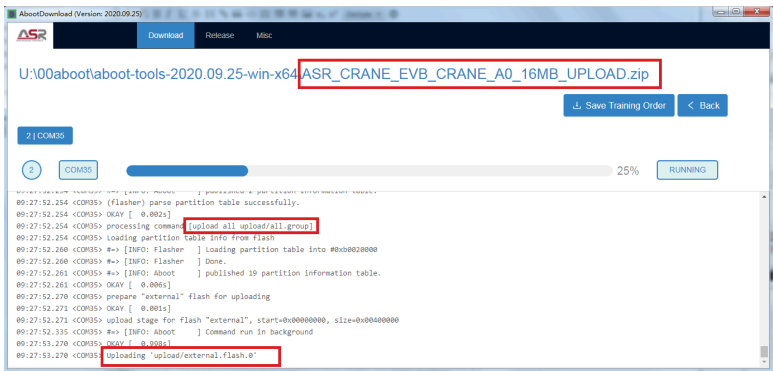


Figure 7.5. upload package is burning

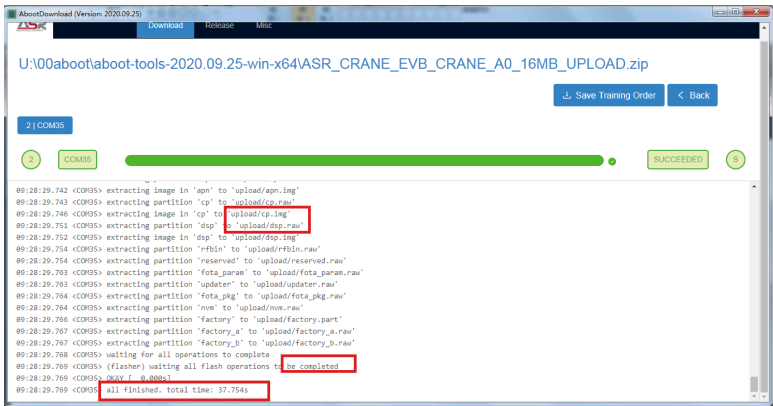


Figure 7.6. upload package is burned

- After completed, the dump data from flash is written to many binary files under the folder **upload**, shown as below:

[new version's directory]\upload.

```
about-tools-2020.09.25-win-x64\upload>dir

2020/09/29  09:28          16,777,216 all.group
2020/09/29  09:28           12,015 apn.img
2020/09/29  09:28          32,768 apn.raw
2020/09/29  09:28          147,456 bootloader.ubi❶
2020/09/29  09:28       3,947,242 cp.img
2020/09/29  09:28       3,948,544 cp.raw
2020/09/29  09:28          995,658 dsp.img
2020/09/29  09:28          999,424 dsp.raw
2020/09/29  09:28       16,777,216 external.flash❷
2020/09/29  09:28          131,072 factory.part
2020/09/29  09:28           65,536 factory_a.raw
2020/09/29  09:28           65,536 factory_b.raw
2020/09/29  09:28           4,096 fota_param.raw
2020/09/29  09:28           4,096 fota_pkg.raw
2020/09/29  09:28           2,232 fwcerts.img
2020/09/29  09:28          12,288 fwcerts.raw
2020/09/29  09:28       2,097,152 nvme.raw
2020/09/29  09:28           2,188 ptable.img
2020/09/29  09:28           4,096 ptable.raw
2020/09/29  09:28           65,536 rd.raw
2020/09/29  09:28       9,306,112 reserved.raw
2020/09/29  09:28          20,480 rfbn.raw
2020/09/29  09:28       5,083,136 system.part❸
2020/09/29  09:28           4,096 updater.raw
```

- ❶ bootloader.ubi: dump from bootloader partition. Since it is in UBI format, it will not same as the bootloader.ubi uncompressed from source zip file. Ubi pebs is dynamically selected when written flash, peb id is not increased from 0. But bootload.ubi also can be useful to see details manually. preboot.bin is one volume, boot2.bin and boot33.bin is another volume.
- ❷ external.flash: the whole data dumped from external.flash
- ❸ system.part: the data dumped from system partition. We alway use it to compare with system.bin which is uncompressed from source zip file. The data must be same. System partition includes data of cp.raw and dsp.raw

The source files uncompressed from zip file(eg: ASR_CRANE_EVB_CRANE_A0_16MB.zip), is shown as below:

[new version's directory]\[the folder which is uncompressed from zip].

```
about-tools-2020.09.25-win-x64\ASR_CRANE_EVB_CRANE_A0_16MB>dir
```

```
2020/09/25 17:11          12,015 apn.bin
2020/09/28 18:56       106,496 bootloader.ubi❶
2020/09/25 17:11     3,947,242 cp.bin
2020/09/28 18:56        2,739 download.json
2020/09/25 17:11     995,658 dsp.bin
2020/09/28 18:56       34,396 flasher.img
2020/09/28 18:56        936 flashinfo.bin
2020/09/28 18:56        101 fota.json
2020/09/28 18:56       2,232 fwcerts.bin
2020/09/28 18:56       2,188 partition.bin
2020/09/28 18:56      17,992 preboot.img
2020/09/25 17:11      65,536 ReliableData.bin
2020/09/25 17:11      20,480 rf.bin
2020/09/28 18:56     5,083,136 system.img❷
2020/09/25 17:11        4,096 updater.bin
```

- ❶ bootloader.ubi: data starts from lowest address in the image, eg: peb0. When burning bootloader.ubi to flash, code will choose a lowest written counter's peb to save data.
- ❷ system.img: the data of system partition. It should be same as system.part dumped from upload function.

8

erase all function

The tool supplies function for erase all space of flash.

Burn program with released normal package only erases these space automatically:

1. Blocks or sectors which will be written data in.
2. partition is forced erased by erase command in **Download Command Table**.

In some special case, user can use the erase all function to erase all space of flash. Reset all bytes of flash to 0xFF.

8.1. Generate erase all package

Two methods are supplied to generate erase all package:

1. Adownload.exe
2. Aboot.exe

8.1.1. Adownload.exe

In cmdline mode, enter arelease.exe in windows's cmd.exe, the help information are shown as below:

execute arelease.exe in windows's cmd.exe.

```
aboot-tools-2020.09.25-win-x64>arelease.exe
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
-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=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_16MB❶
arelease.exe -c . -g --erase-all-only -p ASR_CRANE_EVB -v
CRANE_A0_16MB❷
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 before writing flash, erase all space of flash at firstly.

❷ only erase all space of flash. Image data will not be written to flash.

1) Get products list

excute arelease.exe -l in windows's cmd.exe.

```
about-tools-2020.09.25-win-x64>arelease.exe -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+8MB
0.7		CRANE_A0_16MB
0.8		CRANE_A0_32MB
1.	ASR_JACANA_EVB	
1.0		JACANA_EVB

2) Generate erase all package zip file

For example (product is CRANE_A0_16MB):

Copy the command to execute in windows's cmd.exe. The output file **ASR_CRANE_EVB_CRANE_A0_16MB_ERASEALL.zip** will be generated in the tool's folder.

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

excute arelease.exe in windows's cmd.exe for erase all.

```
about-tools-2020.09.25-win-x64>arelease.exe -c . -g --erase-all-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 ...
    Generating fip image "preboot" with id "preboot.img" ...
        Opening image "preboot_bin" from "U:\00aboot\about-tools-2020.09.25-
win-x64\images\2019.01.15\preboot.bin" ...
    Done.
Done.
    Generating fip image "flasher" with id "flasher.img" ...
        Opening image "flasher_bin" from "U:\00aboot\about-tools-2020.09.25-
win-x64\images\2019.01.15\flasher.bin" ...
```

```
Done.
Done.
Generating finf image "flashinfo" with id "flashinfo.bin" ...
Done.
Generating aptb image "partition" with id "partition.bin" ...
Done.
Done.
Calculating total progress weight ...
Done.
Generating download commands ...
Done.
Generating target release package ...
Done.
Release package generated successfully!
```

8.1.2. Abboot.exe

1) Get products list

- open Release page, the products list is shown, see **Variant** column.
- Click **Package Type** selection frame, change string element from **Normal** to **Erase All**.
- select the product id for generate a upload package. eg: CRANE_A0_16MB.

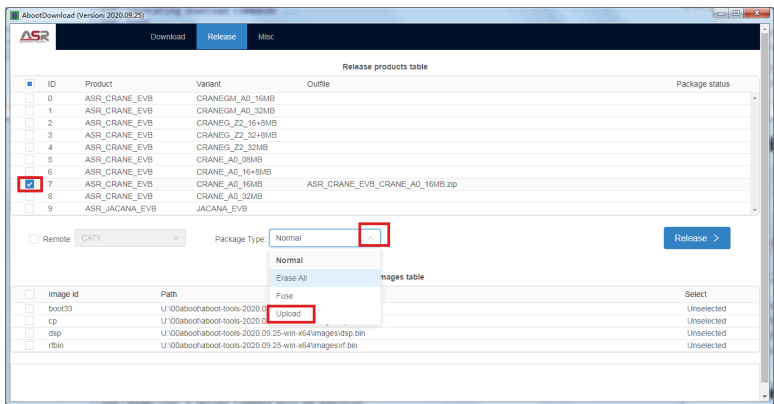


Figure 8.1. select erase all function

- After selected, the default output file name is changed, has the key words **ERASE ALL**. eg: **ASR_CRANE_EVB_CRANE_A0_16MB_ERASEALL.zip**. If user wants to update name, click it and update it manually.

Burn erase all package

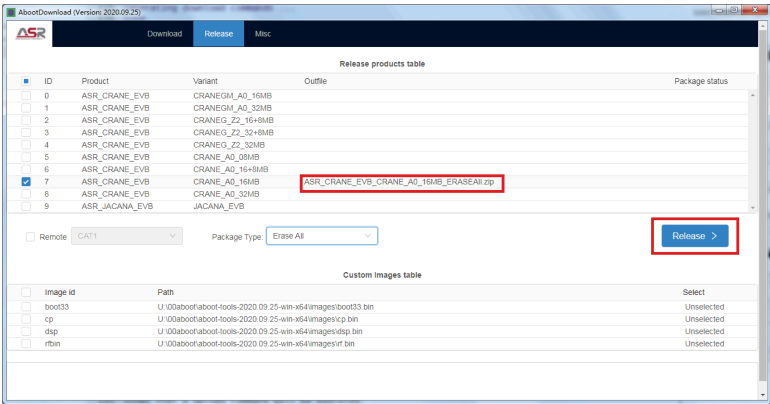


Figure 8.2. erase all function is selected

- Click **Release** Button, the key words can be seen in log console.

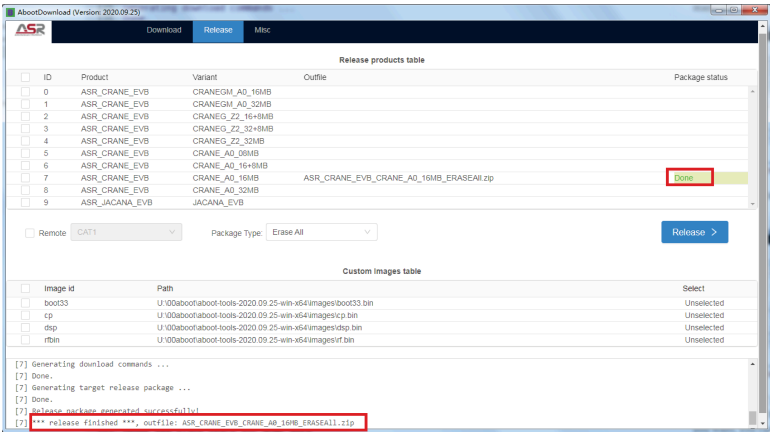


Figure 8.3. erase all package is generated

8.2. Burn erase all package

Two methods are supplied to burn erase all package:

1. Adownload.exe(cmdline)
2. Aboot.exe(GUI)

8.2.1. Adownload.exe

1) In cmdline mode, enter **adownload.exe** in windows cmd.exe, the help information is shown as below:

[new version's directory]\adownload.exe help information.

```

aboot-tools-2020.09.25-win-x64>adownload.exe

Arsmicro 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 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 mode
  -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.zip❶
  adownload.exe -p COM1 -a -s 115200 aboot.zip

```

- ❶ burn erase all package can follow the commond. It is good to add -q.
 2) execute burn program commond:

Need add -q, if adownload.exe is finished, exe will be automatically exit.

```

aboot-tools-2020.09.25-win-x64> adownload.exe -u -a -s -q 115200
ASR_CRANE_EVB_CRANE_A0_16MB_ERASE_ALL.zip

```

8.2.2. Aboot.exe

- In **Burn** page, the latest zip is loaded automatically. If the loaded zip by automatically is not what you want(eg: **ASR_CRANE_EVB_CRANE_A0_16MB_ERASE_ALL.zip**), you can select erase all package zip file by click the hand magnifier Button.

Information of erase all package:

1. Download Command Table:

It shows that a erase command will be executed.

1. Partition table:

It shows the flash space for erase all.

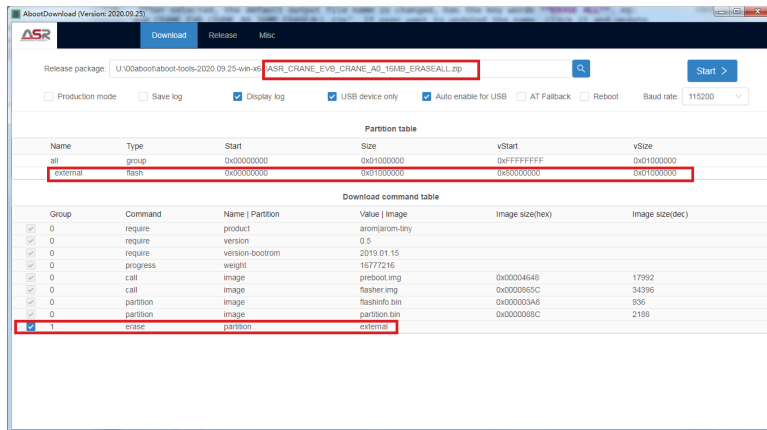


Figure 8.4. erase all package information

- Click **"Start"** Button to execute commands of erase all function. User can check which command is executed from log console.

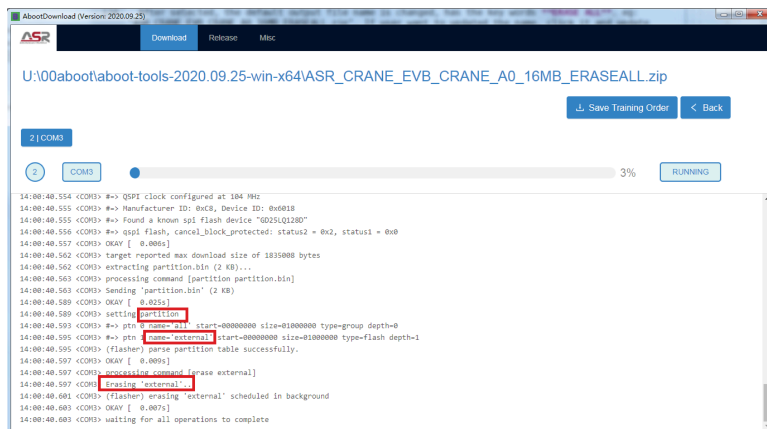


Figure 8.5. erase all package is burning

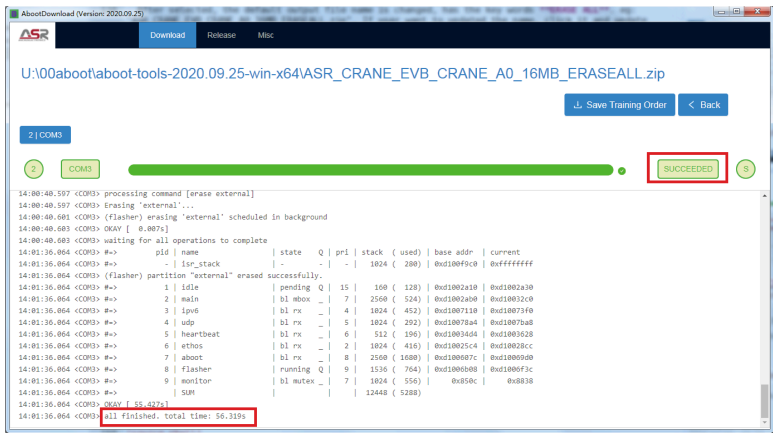


Figure 8.6. erase all package is burned

- After it is completed, all bytes of flash are 0xFF. User also can dump data from flash via upload function to check the result.

Boot in production mode

About.exe(GUI) Misc page supply Button to let board boot in production mode. When board boots in production mode, board can receive AT commands from PC. Board can response to AT commands for audio or RF calibration without GUI.



Boot in production mode is different with normal mode which will start GUI.

9.1. Enter production mode

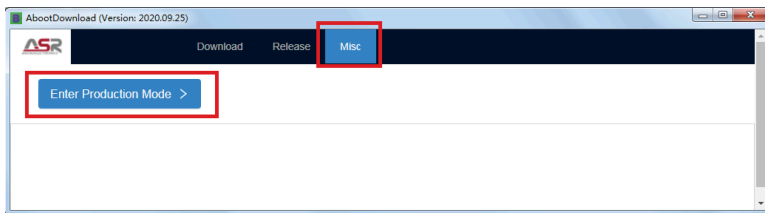


Figure 9.1. board boot as production mode.

- Start about production mode engine

Click button **Enter Procution Mode** to start about production mode engine. Button change to contrary status **Stop Procution Mode**.

- Reset board with button **Reset**.

It is valid only when about production mode engine is running.

The key words **PROD CMD ACCEPTED!** is shown in log console. The power up reason is recorded in RTC register. RTC will not be clear by reset. RTC only be cleared by power down.

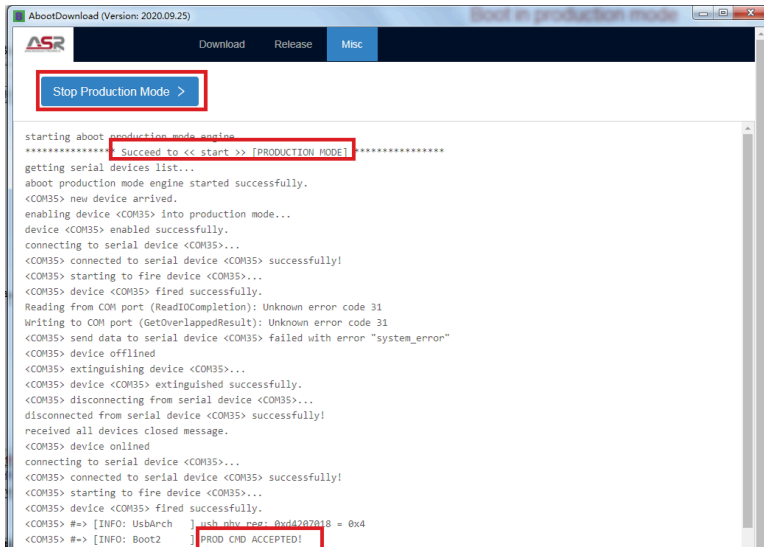


Figure 9.2. board boot as production mode.

- Stop about production mode engine.

Click button **Stop Procution Mode** to stop about production mode engine. Button change to contrary status **Enter Procution Mode**.

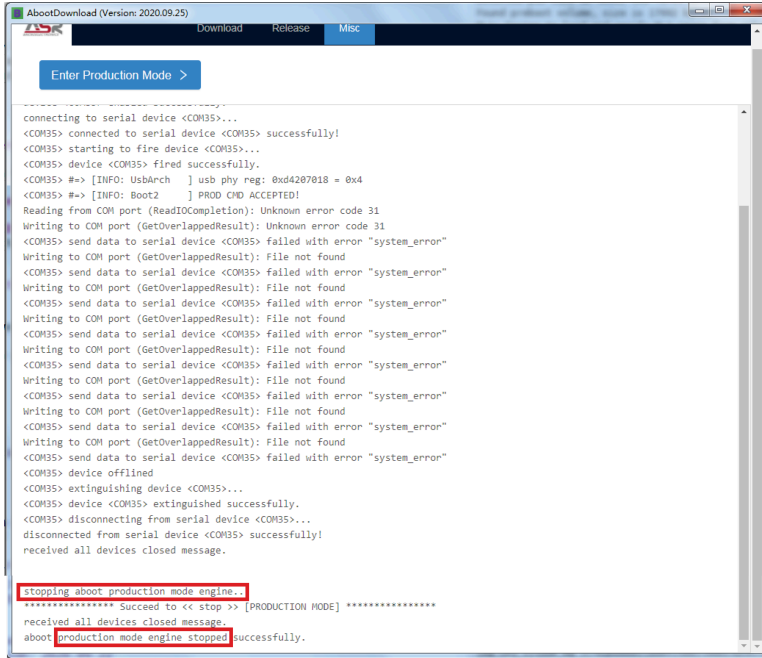


Figure 9.3. stop about production mode engine

Uart traces show board start in production mode

boot in production mode.

```
CHIP_ID: 0x3602, REV_ID: 0x00
PLATFORM: CRANE (SILICON)
CRANE hardware initialization complete.

AROM! (Version: 2020.04.08)
Initializing crypto library...
Loading...
welcome to the trusted boot rom world.
QSPI clock configured at 13 MHz
Detecting QSPI flash devices...
Found an unknown spi flash device which has same manufacture id with
"w25q128jw"
SPI nor flash: Manufacturer ID: 0xEF, Device ID: 0x6019
Detected peb size is 4096, the first good peb number is 0
Found preboot volume, size is 17992 bytes
BL1: Trying to load and verify BL2 image from volume preboot...

[16:04:37.481] 收←BL1: Okay.
[PRI : Preboot  ] Executing preboot application...
```

```
[PRI : Preboot ] Preboot version: 2020.09.22
[PRI : Preboot ] ### Trusted boot mode. ###
[PRI : Preboot ] Found PMIC with Id: 0x3b
[PRI : Preboot ] power_up_reason=0x48.
[PRI : Psram ] [PSRAM] psram_init_craneg_z1.
[PRI : Psram ] AP 8+8MB
[PRI : Psram ] Has embedded flash.
```

```
[16:04:37.533]收←◆Found bootloader volume, size is 66828 bytes
BL1: Trying to load and verify BL2 image from volume bootloader...
```

```
[16:04:37.695]收←◆BL1: Okay.
```

```
[16:04:37.819]收←◆[INFO: Main ]
[INFO: Main ] Starting BOOT2 (Version: 2020.09.22)
[INFO: PMIC ] Get restart cmd = 0x52
[INFO: PMIC ] Set restart status = 0x52
[INFO: QspiNor ] Manufacturer ID: 0xEF, Device ID: 0x6019.
[INFO: QspiNor ] Found a known spi flash device "w25q256jw".
[ERR : QspiNor ] [qspi_nor_flash_probe] update SMPR bit5=1, use
falling edge.
[INFO: QspiNor ] flash_cancel_block_protected_dq: status3 = 0x61,
status2 = 0x2, status1 = 0x0
[INFO: Boot2 ] parse flash info table succes
[16:04:37.854]收←◆sfully.
[INFO: Boot2 ] Loading partition table into #0xb0020000
[INFO: Boot2 ] Done.
[INFO: Boot2 ] Running in production mode.❶
[INFO: Boot2 ] Load and verify boot33 from bootloader volume.
[INFO: Boot2 ] Done.
[INFO: Boot2 ] Decompressing boot33...
[INFO: Boot2 ] Done.
[INFO: Boot2 ] Booting boot33...
[INFO: Boot2 ] boot to 0x7E000000, cpsr should be 0x0x1D3
B33 START
```

```
[16:04:37.889]收←◆LastPowerOff: 0x80 @0xE5, 0x0 @0xE6
```

```
[16:04:37.952]收←◆SECBOOT_SUPPORT:NON-SECBOOT
ver:boot33.bin 20200521_ver1
PowerOn: 0x48
NO FOTA FLAG
PS:LTEGSM
IMG:CRANE_DS_LTEGSM_DKB_Z2_A0_XIP_LWIP_MODULEONLY_16M16M
LDT_RW_CPZ_INFO
```



```
[RW_CPZ_1][DDR_RW_][7ea00000][80051508][00023a10][80051508]
[RW_CPZ_3][PS_NCAH][7ef20000][80074f18][000052c0][80058d96]
[RW_CPZ_4][USBNAH][7ef92400][8007a1d8][00000244][8005928c]
[RW_CPZ_5][CODE_PS][7e200000][8007a41c][0054320c][800593ed]
Region CPZ struct detected from LDT
decompress [ DDR_RW_] from [80051508] to [7ea00000]

[16:04:37.988]收←◆decompress [ PS_NCAH] from [80058d96] to [7ef20000]
decompress [ USBNAH] from [8005928c] to [7ef92400]
decompress [ CODE_PS] from [800593ed] to [7e200000]

[16:04:39.569]收←◆stop decompress as no further RW_CPZ_ detected
logo.bin not exist in PTB
B33 DONE
PC:0x80040000

[16:04:39.638]收←◆Cinit 1
```

- ❶ key words for booting in production mode.

9.2. Exit production mode

Power down the board. The power up reason in RTC is cleared. Board will boot in normal mode next time.

