# UUU (Universal Update Utility)

Frank Li

**REVISION HISTORY**

| NUMBER | DATE | DESCRIPTION | NAME |
|---|---|---|---|
| 02508ba | Fri Jun 16 10:46:21 2023 -0500 | Updated _Sidebar (markdown) | nxpfrankli |
| cf1f381 | Fri Jun 16 10:46:39 2023 -0500 | Updated _Sidebar (markdown) | nxpfrankli |
| 1c16de1 | Fri Jun 16 11:16:23 2023 -0500 | Updated _Sidebar (markdown) | nxpfrankli |
| 79a9c81 | Thu Jun 22 14:31:33 2023 -0500 | Updated Release Notes (asciidoc) | nxpfrankli |
| 2d31142 | Wed Aug 2 11:34:51 2023 -0500 | Updated Release Notes (asciidoc) | nxpfrankli |
| 2d49b33 | Wed Aug 30 09:29:07 2023 -0500 | Updated FAQ (asciidoc) | nxpfrankli |
| 107eba5 | Wed Jan 24 16:14:50 2024 -0600 | Updated Release Notes (asciidoc) | nxpfrankli |
| 7208854 | Wed Jan 24 16:15:03 2024 -0600 | Updated Release Notes (asciidoc) | nxpfrankli |
| fa62563 | Fri Oct 18 17:30:37 2024 -0400 | Updated Home (asciidoc) | nxpfrankli |
| 32e1d97 | Fri Oct 18 17:46:13 2024 -0400 | Updated Home (asciidoc) | nxpfrankli |
| 13cbb56 | Fri Oct 18 17:55:00 2024 -0400 | Updated Home (asciidoc) | nxpfrankli |
| f68a5aa | Fri Oct 18 17:56:07 2024 -0400 | Updated Home (asciidoc) | nxpfrankli |
| a54ddd5 | Fri Oct 18 17:56:36 2024 -0400 | Updated Home (asciidoc) | nxpfrankli |
| f12e600 | Thu Oct 31 16:23:21 2024 -0400 | Updated cmdhelp (asciidoc) | nxpfrankli |
| e81e81c | Thu Oct 31 16:24:21 2024 -0400 | Updated cmdhelp (asciidoc) | nxpfrankli |
| d7aa969 | Thu Dec 19 13:21:10 2024 -0600 | Updated FAQ (asciidoc) | nxpfrankli |
| 5176569 | Thu Dec 19 13:30:54 2024 -0600 | Updated FAQ (asciidoc) | nxpfrankli |

**REVISION HISTORY**

| NUMBER | DATE | DESCRIPTION | NAME |
|---|---|---|---|
| 5679a42 | Thu Jan 9 16:53:39 2025 -0500 | Fix doc build failure | Frank Li |
| 9a10210 | Tue Apr 1 15:38:30 2025 -0500 | Remove duplicate line | nxpfrankli |
| 414b145 | Tue Apr 1 15:39:31 2025 -0500 | frf -> frs | nxpfrankli |
| 8710164 | Wed Aug 20 09:10:33 2025 -0500 | Created Buildin emmc_all script (asciidoc) | nxpfrankli |
| f94bb52 | Wed Aug 20 09:12:15 2025 -0500 | Updated Buildin emmc_all script (asciidoc) | nxpfrankli |
| c6dda43 | Wed Aug 20 09:13:56 2025 -0500 | Updated UUU (asciidoc) | nxpfrankli |
| ff51e14 | Tue Nov 4 13:47:31 2025 -0600 | Updated Release Notes (asciidoc) | nxpfrankli |
| e32aed8 | Tue Nov 4 13:48:47 2025 -0600 | Updated Release Notes (asciidoc) | nxpfrankli |

# Contents

**1 Introduction** **1**
1.1 Running environment . . . 1
1.2 Common Usage . . . 1
1.3 UUU Script . . . 2
1.4 Firmware(default it is uboot) Requirements . . . 3
1.5 Setup auto parameter complete . . . 3
1.5.1 windows . . . 3
1.5.2 linux . . . 3

**2 Syntactic** **3**

**3 Usage Example** **5**
3.1 Basic . . . 5
3.1.1 Download boot loader for imx6\imx7 . . . 5
3.1.2 Download boot loader for imx8qxp . . . 5
3.1.3 Download SPL and uboot, such as imx8mq. . . . 5
3.1.4 Burn Android Image to eMMC . . . 5
3.1.5 Burn yocto Image to eMMC . . . 5
3.2 Built-in script . . . 6
3.3 multi boards support . . . 6
3.3.1 For the same boards . . . 6
3.3.2 For the difference boards . . . 6
3.4 Talk with fastboot . . . 6
3.4.1 boot linux kernel . . . 6
3.4.2 write image to emmc . . . 7
3.5 Talk with linux kenrel, transfer file between board and pc . . . 7

**4 Sample scripts** **7**

**5 Built-in scripts** **13**

**6 Supported protocol** **13**
6.1 SDP: i.MX6/7 ROM download protocol . . . 14
6.1.1 Supported command: . . . 14
6.2 HABv4 closed chip support . . . 14
6.3 SDPU\SDPV: uboot implement simplified ROM SDP protocol . . . 15
6.4 SDPS: i.MX8QXP and i.MX8QM ROM download protocol . . . 15
6.5 FB: Android fastboot protocol . . . 16
6.5.1 Support command: . . . 16
6.6 FBK: Android fastboot protocol, implement at initramfs. See project imx-uuc . . . 16
6.6.1 Support command: . . . 16
6.7 Common command for all protocol . . . 17

**7 Migration from mfgtool ucl2.xml** **17**

**8 Win 7 User Guide** **19**

8.1 Back Ground . . . 19

8.2 Install updated winusb inf file . . . 19

8.3 Use zadig to install winusb driver . . . 20

**9 FAQ** **21**

**10 Build Steps** **25**

10.1 windows . . . 25

10.2 linux . . . 25

**11 Uboot config requirement** **26**

**12 kernel config requirement** **27**

**13 Release Notes** **27**

13.1 1.5.233 (stable) . . . 27

13.1.1 New features . . . 27

13.2 uuu_1.5.222 . . . 27

13.2.1 bug fixes . . . 27

13.3 uuu_1.5.220 . . . 28

13.3.1 Bug fixes . . . 28

13.4 uuu_1.5.219 . . . 28

13.4.1 Bug fixes . . . 28

13.5 uuu_1.5.218 . . . 28

13.5.1 new feature . . . 28

13.5.2 Bug fixes . . . 28

13.6 uuu_1.5.201 (stable) . . . 28

13.6.1 new feature . . . 28

13.6.2 Bug fixes . . . 28

13.7 uuu_1.5.197 . . . 28

13.7.1 new feature . . . 28

13.8 uuu_1.5.191 . . . 29

13.8.1 new feature . . . 29

13.9 uuu_1.5.182 (stable) . . . 29

13.9.1 his is hot fix for 1.5.181. . . . 29

13.10uuu_1.5.181 . . . 29

13.11uuu_1.5.179 . . . 29

13.12New feature . . . 29

13.131.5.177 . . . 29
13.13.1 New features . . . 29
13.141.6.165 (stable) . . . 29
13.151.5.163 . . . 29
13.15.1 New features . . . 29
13.15.2 Bug fixes . . . 29
13.161.5.125 (stable) . . . 29
13.171.5.120 . . . 30
13.17.1 New features . . . 30
13.181.5.109 . . . 30
13.191.5.104 . . . 30
13.19.1 New fetaures . . . 30
13.19.2 Bug fixes . . . 30
13.201.5.21 (stable) . . . 30
13.20.1 New features . . . 30
13.20.2 Bug fixes . . . 30
13.211.4.243 . . . 31
13.21.1 New features . . . 31
13.21.2 Bug fixes . . . 31
13.221.4.193 . . . 31
13.22.1 New features . . . 31
13.22.2 Bug fixes . . . 31
13.231.4.165 . . . 31
13.23.1 New features . . . 31
13.23.2 Bug fixes . . . 31
13.241.4.139 . . . 32
13.24.1 New features . . . 32
13.24.2 Bug fixes . . . 32
13.251.4.72 . . . 32
13.25.1 New features . . . 32
13.25.2 Bug fixes . . . 32
13.261.4.43 . . . 32
13.26.1 New features . . . 32
13.26.2 Bug fixes . . . 32
13.271.3.191 . . . 33
13.27.1 New features . . . 33
13.27.2 Bug fixes . . . 33
13.281.3.154 . . . 33
13.28.1 New features . . . 33

13.28.2 Bug fixes . . . 33
13.29 1.3.134 . . . 33
13.29.1 Bug fixes . . . 33
13.30 1.3.130 . . . 33
13.30.1 New features . . . 33
13.30.2 Bug fixes . . . 34
13.31 1.3.102 . . . 34
13.31.1 New features . . . 34
13.31.2 Bug fixes . . . 34
13.32 1.3.82 . . . 34
13.32.1 New features . . . 34
13.32.2 Bug fixes . . . 35
13.33 1.2.135 . . . 35
13.33.1 New features . . . 35
13.33.2 Bug fixes . . . 35
13.34 1.2.91 . . . 35
13.34.1 New features . . . 35
13.34.2 Bug fixes . . . 36
13.35 1.2.0 . . . 36
13.35.1 New features . . . 36
13.35.2 Bug fixes . . . 36
13.36 1.1.81 . . . 36
13.36.1 New features . . . 36
13.36.2 Bug fixes . . . 36
13.37 1.1.41 . . . 37
13.37.1 New features . . . 37
13.37.2 Bug fixes . . . 37

**14 Known issue** **37**

# 1 Introduction

Welcome to the UUU (Universal Update Utility), the next evolution of MFGTools, also known as MFGTools v3.

UUU is a tool designed for deploying images on Freescale/NXP I.MX chips.

Over time, the demand for a cross-platform update utility that works seamlessly on both Linux and Windows has grown. UUU meets this need, offering identical functionality and usage on both operating systems, allowing the same scripts to run smoothly on either platform.

UUU is **command line tools**. look like

```
uuu (universal update utility) for nxp imx chips -- libuuu-1.0.1-gffd9837

Succues:0       Failure:3               Wait for Known USB Device Appear...

1:11     5/5 [                            ] SDP: jump -f u-boot-dtb.imx - ↩
    ivtinitramf....
2:1      1/5 [===>                        ] SDP: boot -f u-boot-imx7dsabresd_sd. ↩
    imx ....
```

UUU is designed as a common library and UI, making it easy for users to integrate the UUU library into their own tools. It also seamlessly runs within any scripts.

PDF version of the wiki content is available on the release page for convenience.

## 1.1 Running environment

- Windows 10, 64bit, early version(below 1.2.0) need install vs2017 redistribute package
- Ubuntu 16.14 or above, 64bit

**Windows 7 user please read WIN7-User-Guide**

## 1.2 Common Usage

Set the board's boot pin to USB serial download mode. In most cases, the iMX ROM will automatically fall back to USB serial download mode if there is a boot failure.

To download and boot U-Boot:

```
uuu bootloader
```

To burn U-Boot into eMMC:

```
uuu -b emmc bootloader
```

To burn the boot image into QSPI flash:

```
uuu -b qspi qspi_bootloader
```

To burn the root filesystem image into eMMC:

```
uuu -b emmc_all bootloader rootfs.wic
```

To decompress the root filesystem image and burn it into eMMC:

```
uuu -b emmc_all bootloader rootfs.wic.zstd
```

To decompress the root filesystem image and burn it into eMMC without the bootloader (since version 1.4.146 and after 2021 Q4 Yocto image):

```
uuu -b emmc_all rootfs.wic.zstd
```

**Notes:** "Bootloader" refers to a bootable image that includes the ROM-required header. For i.MX6 and i.MX7, this is typically `uboot.imx`, while for i.MX8QXP, i.MX8QM, i.MX8MM, and i.MX8MQ, it is `flash.bin`.

To burn the release image into eMMC, use the following command:

```
uuu L4.9.123_2.3.0_8mm-ga.zip
```

**Note:** Some releases bundle multiple board configurations into a single zip package, use uuu release.zip/uuu.auto-<boardname>

For more usage details, please refer to the Example.

## 1.3 UUU Script

UUU scripts are simple plain text files.

**The first line must be:**

```
uuu_version 1.0.1
```

The version number specifies the minimum UUU version required to parse and run the script.

After that, include the UUU commands.

UUU commands follow this format:

```
PROTOCOL: CMD
```

Here's an example script for booting U-Boot on i.MX6 and i.MX7:

```
uuu_version 1.0.1
SDP: dcd -f u-boot.imx
SDP: write -f u-boot.imx -ivt 0
SDP: jump -f u-boot.imx -ivt 0
```

For more sample scripts, see: Sample-script[].

The following table shows environment variables that may be useful when writing UUU scripts.

Table 2: Table Android Fastboot environment

| Variable | Description |
|---|---|
| `fastboot_dev` | fastboot flash device, support mmc and sata |
| `fastboot_buffer` | fastboot download buffer adddress |
| `fastboot_bytes` | fastboot download file size |
| `emmc_dev` | eMMC device number |
| `sd_dev` | sd slot device number |

| What you want | Required Firmware |
|---|---|

### 1.4 Firmware(default it is uboot) Requirements

| What you want | Required Firmware |
|---|---|
| Download bootloader | N/A |
| Burn Image to eMMC/SD | uboot with fastboot enable |
| Burn Image to qspi\spi\nor | uboot with fastboot enable |
| Burn Image into Nand flash | uboot(1), linux kernel\initramfs\uboot\dtb |
| Need linux shell cmd such as fdisk | uboot(1), linux kernel\initramfs\uboot\dtb |
| Boot linux kenrel with rootfs already in eMMC | uboot with fastboot enable |
| Boot Linux kernel with nfs over USB | uboot with fastboot enable, initramfs |

(1) It's recommended to enable fastboot. If the ROM HID supports writing additional images to DDR, you can load the kernel, DTB, and initramfs to DDR before jumping to U-Boot. Enabling fastboot provides more flexibility for modifying the kernel command line.

### 1.5 Setup auto parameter complete

#### 1.5.1 windows

Just power shell support customized auto complete

Powershell: Enjoy auto [tab] command complete by run below command or put into Documents\WindowsPowerShell\Microsoft.PowerS

```
Register-ArgumentCompleter -CommandName uuu -ScriptBlock {param($commandName, ↩
    $parameterName,$wordToComplete,$commandAst,$fakeBoundParameter); C:\Users\ ↩
    nxa23210\uuu\uuu\x64\Release\lib\uuu.exe -autocomplete $parameterName }
```

#### 1.5.2 linux

Enjoy auto [tab] command complete by put below script into /etc/bash_completion.d/uuu

```
_uuu_autocomplete()
{
     COMPREPLY=($(/home/lizhi/source/mfgtools/uuu/uuu $1 $2 $3))
}
complete -o nospace -F _uuu_autocomplete  uuu
```

## 2 Syntactic

```
uuu (Universal Update Utility) for nxp imx chips -- libuuu_1.5.191-1-g5d77a61

uuu [-d -m -v -V -bmap -no-bmap] <bootloader|cmdlists|cmd>

    bootloader      download bootloader to board by usb
    cmdlist         run all commands in cmdlist file
                    If it is path, search uuu.auto in dir
                    If it is zip, search uuu.auto in zip
    cmd             Run one command, use -H see detail
                    example: SDPS: boot -f flash.bin
```

```
    -d              Daemon mode, wait for forever.
    -v -V           verbose mode, -V enable libusb error\warning info
    -dry            Dry run mode, check if script or cmd correct
    -bmap           Try using .bmap files even if flash commands do not specify  ↩
       them
    -no-bmap        Ignore .bmap files even if flash commands specify them
    -m <usbpath>    Only monitor these paths.
                      -m 1:2 -m 1:3

    -ms <serial_no> Monitor the serial number prefix of the device using ' ↩
       serial_no'.
    -t <timeout>    Timeout second for wait known usb device appeared
    -T <timeout>    Timeout second for wait next known usb device appeared at  ↩
       stage switch
    -e <key=value>  set environment variable key=value
    -pp <ms>        usb polling period in milliseconds
    -dm             disable small memory
uuu -s              Enter shell mode. uuu.inputlog record all input commands
                    you can use "uuu uuu.inputlog" next time to run all commands

uuu -udev           linux: show udev rule to avoid sudo each time
uuu -lsusb          List connected know devices
uuu -IgSerNum       Set windows registry to ignore USB serial number for known uuu ↩
    devices
uuu -h              show general help
uuu -H              show general help and detailed help for commands

uuu [-d -m -v -bmap -no-bmap] -b[run] <emmc|emmc_all|fat_write|nand|nvme_all|qspi| ↩
   sd|sd_all|spi_nand|spl> arg...
        Run Built-in scripts
        emmc    burn boot loader to eMMC boot partition
                arg0: _flash.bin  bootloader
                arg1: _image[Optional]  image burn to emmc, default is the same as ↩
                    bootloader
        emmc_all        burn whole image to eMMC
                arg0: _flash.bin  bootloader, which can extract from wic image
                arg1: _image[Optional]  wic image burn to emmc.
        fat_write       update one file in fat partition, require uboot fastboot  ↩
           running in board
                arg0: _image  image, which cp to fat partition
                arg1: _device  storage device, mmc\sata
                arg2: _partition  fat partition number, like 1:1
                arg3: _filename[Optional]  file name in target fat partition, only ↩
                    support rootdir now
        nand    burn boot loader to NAND flash
                arg0: _flash.bin  bootloader
                arg1: _image[Optional]  image burn to nand, default is the same as ↩
                    bootloader
        nvme_all        burn whole image io nvme storage
                arg0: _flash.bin  bootloader, which can extract from wic image
                arg1: _image[Optional]  wic image burn to emmc.
        qspi    burn boot loader to qspi nor flash
                arg0: _flexspi.bin  bootloader
                arg1: _image[Optional]  image burn to flexspi, default is the same ↩
                    as bootloader
        sd      burn boot loader to sd card
                arg0: _flash.bin  bootloader
```

```
                arg1: _image[Optional]  image burn to emmc, default is the same as ↩
                    bootloader
        sd_all  burn whole image to sd card
                arg0: _flash.bin  bootloader, which can extract from wic image
                arg1: _image[Optional]  wic image burn to emmc.
        spi_nand        burn boot loader to spi nand flash
                arg0: _flexspi.bin  bootloader
                arg1: _image[Optional]  image burn to fspinand, default is the  ↩
                   same as bootloader
        spl     boot spl and uboot
                arg0: _flash.bin

uuu -bshow <emmc|emmc_all|fat_write|nand|nvme_all|qspi|sd|sd_all|spi_nand|spl>
        Show built-in script....

Notes:  Some board supports super speed (USB3.0). USB 3.0 port path is difference  ↩
   USB 2.0. If use -m to filter port, you need add USB 3.0 port number otherwise  ↩
   fastboot will not be detected.
```

# 3 Usage Example

## 3.1 Basic

### 3.1.1 Download boot loader for imx6\imx7

```
uuu uboot.imx
```

### 3.1.2 Download boot loader for imx8qxp

```
uuu flash.bin
```

### 3.1.3 Download SPL and uboot, such as imx8mq.

```
uuu sdp: boot -f flash.bin
uuu sdpu: delay 1000
uuu sdpu: write -f flash.bin -offset 0x57c00
uuu sdpu: jump
```

### 3.1.4 Burn Android Image to eMMC

```
uuu android.zip (not implement default yet)
```

### 3.1.5 Burn yocto Image to eMMC

```
uuu L4.9.123_2.3.0_8mm-ga.zip
```

## 3.2 Built-in script

```
uuu -b emmc bootloader                              Write bootloader to emmc
uuu -b emmc_all bootloader rootfs.sdcard            Write rootfs to emmc
uuu -b emmc_all bootloader rootfs.sdcard.bz2/*      Decompress rootfs and write  ↩
    rootfs to emmc
uuu -b sd bootloader                                Write bootloader to sd card
uuu -b sd_all bootloader rootfs.sdcard              Write rootfs to sd card
uuu -b sd_all bootloader rootfs.sdcard.bz2/*        Decompress rootfs and write  ↩
    rootfs to sd card
uuu -b qspi qspi_bootloader                         write bootloader to qspi
uuu -b qspi qspi_bootloader m4image                 write m4image to qpsi
uuu -b spl bootloader                               Download SPL and uboot
```

**Notes:**

```
Some boards have many sd slot. built-in script only work uboot environment ${ ↩
    sd_dev} point to slot
Some boards have not emmc chip, emmc build in script does not work for such boards
```

## 3.3 multi boards support

### 3.3.1 For the same boards

```
uuu -d uuu.auto                         The same boards connected
```

### 3.3.2 For the difference boards

```
uuu -d -m 1:1 -m 2:1 boardA_uuu.auto           monitor port 1:1 and 2:1 for  ↩
    boardsA.
uuu -d -m 1:3 -m 4:1 boardB_uuu.auto           monitor port 1:3 and 4:1 for  ↩
    boardsB.
```

**Note: please avoid monitor the same port by difference uuu instance, which cause unexpected result.**

## 3.4 Talk with fastboot

### 3.4.1 boot linux kernel

```
uuu FB: ucmd setenv fastboot_buffer ${loadaddr}
uuu FB: download -f Image
uuu FB: ucmd setenv fastboot_buffer ${fdt_addr}
uuu FB: download -f imx8qxp_mek.dtb
uuu FB: acmd booti ${loadaddr} - ${fdt_addr}
```

Notes: Linux/Mac user need add \ before $ to avoid shell replace environment variable.

Extended environment for fastboot

```
fastboot_buffer      Image download address
fastboot_bytes       reflect previous download image byte size
```

#### 3.4.2 write image to emmc

```
uuu FB: flash -raw2sparse all <image file>
```

### 3.5 Talk with linux kenrel, transfer file between board and pc

```
on board, run \linuxrc&
on pc,    uuu fbk:ucp file t:\file
```

## 4 Sample scripts

This pages list some examples on scripts to help UUU to make some tasks automatically.

Copy the source code to the file `uuu.auto` and execute the application executable file.

# Download spl and uboot for imx8mm and imx8mq

```
uuu_version 1.2.39

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

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

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

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

# Burn bootloader to eMMC boot partition

```
uuu_version 1.2.39

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

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

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

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

FB: ucmd setenv fastboot_dev mmc
FB: ucmd setenv mmcdev ${emmc_dev}
FB: ucmd mmc dev ${emmc_dev}
FB: flash bootloader _flash.bin
FB: ucmd mmc partconf ${emmc_dev} 0 1 0
FB: Done
```

## Burn android to eMMC

```
uuu_version 1.2.39

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

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

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

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

FB: ucmd setenv fastboot_dev mmc
FB: ucmd setenv mmcdev ${emmc_dev}
FB: ucmd mmc dev ${emmc_dev}
FB: flash gpt partition-table.img
FB: flash boot_a boot-imx8qxp.img
FB: flash system_a system.img
```

```
FB: flash vendor_a vendor.img
FB: flash vbmeta_a vbmeta-imx8qxp.img
#FB: ucmd setenv fastboot_buffer ${loadaddr}
#FB: download -f u-boot-imx8qxp.imx
#FB: ucmd setexpr fastboot_blk ${fastboot_bytes}
#FB: ucmd setexpr fastboot_blk ${fastboot_blk} + 0x1FF
#FB: ucmd setexpr fastboot_blk ${fastboot_blk} / 0x200
#FB: ucmd mmc partconf ${emmc_dev} 1 1 1
#FB: ucmd echo ${fastboot_buffer}
#FB: ucmd echo ${fastboot_blk}
#FB: ucmd mmc write ${fastboot_buffer} 0x40  ${fastboot_blk}
FB: flash bootloader u-boot-imx8qxp.imx
FB: ucmd mmc partconf ${emmc_dev} 0 1 0
FB: Done
```

## Burn yocto image to eMMC by linux kernel

```
uuu_version 1.2.39

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

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

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

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

FB: ucmd setenv fastboot_buffer ${loadaddr}
FB: download -f _Image
FB: ucmd setenv fastboot_buffer ${fdt_addr}
FB: download -f _board.dtb
FB: ucmd setenv fastboot_buffer ${initrd_addr}
FB: download -f _initramfs.cpio.gz.uboot
FB: ucmd setenv mfgtool_args ${mfgtool_args} mfg_mmcdev=${emmc_dev}
FB: ucmd run mfgtool_args
FB: acmd booti ${loadaddr} ${initrd_addr} ${fdt_addr}

# get mmc dev number from kernel command line
FBK: ucmd cmdline=`cat /proc/cmdline`;cmdline=${cmdline#*mfg_mmcdev=};cmds=( ↩
    $cmdline);echo ${cmds[0]}>/tmp/mmcdev
# Wait for mmc
FBK: ucmd mmc=`cat /tmp/mmcdev`; while [ ! -e /dev/mmcblk${mmc} ]; do sleep 1;  ↩
```

```
    echo "wait for /dev/mmcblk${mmc} appear"; done;
# create partition
FBK: ucmd mmc=`cat /tmp/mmcdev`; PARTSTR=$'10M,500M,0c\n600M,,83\n'; echo " ↩
    $PARTSTR" | sfdisk --force /dev/mmcblk${mmc}

FBK: ucmd mmc=`cat /tmp/mmcdev`; dd if=/dev/zero of=/dev/mmcblk${mmc} bs=1k seek ↩
    =4096 count=1
FBK: ucmd sync
FBK: ucmd mmc=`cat /tmp/mmcdev`; echo 0 > /sys/block/mmcblk${mmc}boot0/force_ro
FBK: ucp  _flash.bin t:/tmp
FBK: ucmd mmc=`cat /tmp/mmcdev`; dd if=/tmp/_flash.bin of=/dev/mmc${mmc}boot0 bs=1 ↩
    K seek=32
FBK: ucmd mmc=`cat /tmp/mmcdev`; echo 1 > /sys/block/mmcblk${mmc}boot0/force_ro
FBK: ucmd mmc=`cat /tmp/mmcdev`; while [ ! -e /dev/mmcblk${mmc}p1 ]; do sleep 1;  ↩
    done
FBK: ucmd mmc=`cat /tmp/mmcdev`; mkfs.vfat /dev/mmcblk${mmc}p1
FBK: ucmd mmc=`cat /tmp/mmcdev`; mkdir -p /mnt/fat
FBK: ucmd mmc=`cat /tmp/mmcdev`; mount -t vfat /dev/mmcblk${mmc}p1 /mnt/fat
FBK: ucp  _Image t:/mnt/fat
FBK: ucp  _board.dtb t:/mnt/fat
FBK: ucmd umount /mnt/fat
FBK: ucmd mmc=`cat /tmp/mmcdev`; mkfs.ext3 -F -E nodiscard /dev/mmcblk${mmc}p2
FBK: ucmd mkdir -p /mnt/ext3
FBK: ucmd mmc=`cat /tmp/mmcdev`; mount /dev/mmcblk${mmc}p2 /mnt/ext3
FBK: acmd export EXTRACT_UNSAFE_SYMLINKS=1; tar -jx -C /mnt/ext3
FBK: ucp  _rootfs.tar.bz2 t:-
FBK: Sync
FBK: ucmd umount /mnt/ext3
FBK: DONE
```

## Burn NAND flash by linux kernel

```
uuu_version 1.2.39

# Please replace below item with actual name
# @_flash_fw.bin                | boot loader firmware, for i.MX8QM/QX, it's  ↩
    different from _flash.bin, for all other platforms, it's same as _flash.bin
# @_flash.bin                   | boot loader file burn to NAND
# @_Image                       | linux kernel image, zImage for arm32, Image for  ↩
    arm64
# @_board.dtb                   | board dtb file
# @_initramfs.cpio.gz.uboot     | initramfs
# @_tee                         | optee image
# @_rootfs.tar.bz2              | rootfs

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

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

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

```
SDPU: jump
# }

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


FB: ucmd setenv fastboot_buffer ${loadaddr}
FB: download -f _Image
FB: ucmd setenv fastboot_buffer ${fdt_addr}
FB: download -f _board.dtb
FB: ucmd setenv fastboot_buffer ${initrd_addr}
FB: download -f _initramfs.cpio.gz.uboot
FB: ucmd setenv bootargs ${bootargs} ${mtdparts}
#FB: ucmd setenv bootargs console=ttymxc3,115200  ${mtdparts}
FB: acmd ${kboot} ${loadaddr} ${initrd_addr} ${fdt_addr}

FBK: ucmd cat /proc/mtd
FBK: ucmd cat /proc/mtd | while read dev size erase name; do mtd=${dev:3}; mtd=${ ↩
    mtd%:}; name=${name%\"}; name=${name#\"}; echo export $name=$mtd >> /tmp/mtd.sh ↩
    ; done;

FBK: ucmd chmod 777 /tmp/mtd.sh
FBK: ucmd mount -t debugfs debugfs /sys/kernel/debug

# write boot loader
FBK: ucmd source /tmp/mtd.sh; flash_erase /dev/mtd${nandboot} 0 0
FBK: ucp _flash.bin t:/tmp/boot
FBK: ucmd source /tmp/mtd.sh; cd /tmp; if ! [[ `cat /sys/devices/soc0/soc_id` = *" ↩
    MX8Q"* ]]; then pad="-x"; fi; kobs-ng init $pad -v --chip_0_device_path=/dev/ ↩
    mtd${nandboot} /tmp/boot

# burn FIT
FBK: ucmd if [[ `cat /tmp/mtd.sh` = *"nandfit"* ]]; then source /tmp/mtd.sh;  ↩
    flash_erase /dev/mtd${nandfit} 0 0; dd if=/tmp/boot of=/tmp/fit; nandwrite -p / ↩
    dev/mtd${nandfit} -p /tmp/fit; fi

# burn kernel
FBK: ucmd source /tmp/mtd.sh; flash_erase /dev/mtd${nandkernel} 0 0
FBK: ucp _Image t:/tmp/img
FBK: ucmd source /tmp/mtd.sh; nandwrite -p /dev/mtd${nandkernel} -p /tmp/img

# burn dtb
FBK: ucmd source /tmp/mtd.sh; flash_erase /dev/mtd${nanddtb} 0 0
FBK: ucp _board.dtb t:/tmp/dtb
FBK: ucmd source /tmp/mtd.sh; nandwrite -p /dev/mtd${nanddtb} -p /tmp/dtb

# burn uTee
FBK: ucmd source /tmp/mtd.sh; flash_erase /dev/mtd${nandtee} 0 0
FBK: ucp _tee t:/tmp/tee
FBK: ucmd source /tmp/mtd.sh; nandwrite -p /dev/mtd${nandtee} -p /tmp/tee
```

```
# burn rootfs
FBK: ucmd source /tmp/mtd.sh; flash_erase /dev/mtd${nandrootfs} 0 0
FBK: ucmd source /tmp/mtd.sh; ubiattach /dev/ubi_ctrl -m ${nandrootfs}
FBK: ucmd source /tmp/mtd.sh; ubimkvol /dev/ubi0 -Nnandrootfs -m
FBK: ucmd source /tmp/mtd.sh; mkdir -p /mnt/mtd
FBK: ucmd source /tmp/mtd.sh; mount -t ubifs ubi0:nandrootfs /mnt/mtd
FBK: acmd export EXTRACT_UNSAFE_SYMLINKS=1; tar -jx -C /mnt/mtd
FBK: ucp _rootfs.tar.bz2 t:-
FBK: sync
FBK: ucmd umount /mnt/mtd

FBK: done
```

## Download kernel and mount nfs by usb ncm

```
uuu_version 1.2.39

# Please replace below item with actual name
# @_flash.bin                   | boot loader
# @_Image                       | linux kernel image, zImage for arm32, Image for  ↩
    arm64
# @_board.dtb                   | board dtb file
# @_initramfs.cpio.gz.uboot     | initramfs
# @_nfspath                     | rootfs path without ip address, ip address will  ↩
    auto detect

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

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

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

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

FB: ucmd setenv fastboot_buffer ${loadaddr}
FB: download -f _Image
FB: ucmd setenv fastboot_buffer ${fdt_addr}
FB: download -f _board.dtb
FB: ucmd setenv fastboot_buffer ${initrd_addr}
FB: download -f _initramfs.cpio.gz.uboot
FB: ucmd setenv nfsroot _nfspath
FB: ucmd setenv bootargs console=${console},${baudrate} nfsroot=${nfsroot} init=/ ↩
```

```
    linuxrc root=/dev/nfs
#uncomment below line to stop at initramfs
#FB: ucmd setenv bootargs console=${console},${baudrate} nfsroot=${nfsroot}
FB: acmd ${kboot} ${loadaddr} ${initrd_addr} ${fdt_addr}
FB: done
```

# 5 Built-in scripts

include::Buildin emmc_all script[Built-in scripts]

# 6 Supported protocol

UUU is scripted base multi protocol system.

Built in config:

```
        Pctl     Chip            Vid     Pid     BcdVersion
         ================================================
         SDPS:    MX8QXP          0x1fc9  0x012f  [0x0002..0xffff]
         SDPS:    MX8QM           0x1fc9  0x0129  [0x0002..0xffff]
         SDPS:    MX8DXL          0x1fc9  0x0147
         SDPS:    MX28            0x15a2  0x004f
         SDPS:    MX815           0x1fc9  0x013e
         SDPS:    MX865           0x1fc9  0x0146
         SDPS:    MX8ULP          0x1fc9  0x014a
         SDPS:    MX8ULP          0x1fc9  0x014b
         SDPS:    MX93            0x1fc9  0x014e
         SDP:     MX7D            0x15a2  0x0076
         SDP:     MX6Q            0x15a2  0x0054
         SDP:     MX6D            0x15a2  0x0061
         SDP:     MX6SL           0x15a2  0x0063
         SDP:     MX6SX           0x15a2  0x0071
         SDP:     MX6UL           0x15a2  0x007d
         SDP:     MX6ULL          0x15a2  0x0080
         SDP:     MX6SLL          0x1fc9  0x0128
         SDP:     MX7ULP          0x1fc9  0x0126
         SDP:     MXRT106X        0x1fc9  0x0135
         SDP:     MX8MM           0x1fc9  0x0134
         SDP:     MX8MQ           0x1fc9  0x012b
         SDPU:    SPL             0x0525  0xb4a4  [0x0000..0x04ff]
         SDPV:    SPL1            0x0525  0xb4a4  [0x0500..0x9998]
         SDPV:    SPL1            0x1fc9  0x0151  [0x0500..0x9998]
         SDPU:    SPL             0x0525  0xb4a4  [0x9999..0x9999]
         SDPU:    SPL             0x3016  0x1001  [0x0000..0x04ff]
         SDPV:    SPL1            0x3016  0x1001  [0x0500..0x9998]
         FBK:                     0x066f  0x9afe
         FBK:                     0x066f  0x9bff
         FBK:                     0x1fc9  0x0153
         FB:                      0x0525  0xa4a5
         FB:                      0x18d1  0x0d02
         FB:                      0x3016  0x0001
         FB:                      0x1fc9  0x0152
```

Table 4: Table UUU Protocol to USB lower level Map

| UUU Protocol | USB Low level transfer |
|---|---|
| SDP | HID i.MX6/7, i.MX8 MM, i.MX8M |
| SDPU\SDPV | HID uboot implement of SDP, SPL download uboot |
| SDPS | HID i.MX8QXP i.MX8QM |
| FB | winusb (windows), raw transfer by libusb |
| FBK | winusb (windows), raw transfer by libusb |

## 6.1 SDP: i.MX6/7 ROM download protocol

### 6.1.1 Supported command:

Run DCD from image with ivt header

```
dcd   -f  <filename>
```

write image to address.

```
write -f  <filename> [-addr 0x000000] [-ivt 0]
```

ivt 0 means write to the address, which ivt pointer

jump to image with ivt header

```
jump  -f  <filename> [-ivt 0]
```

boot image, include (dcd, write and jump three commands)

```
boot  -f  <filename> [-nojump]
```

## 6.2 HABv4 closed chip support

For boot images not including a DCD table the same image used for SDCard/eMMC boot can be used with UUU tool.

For boot images including a DCD table, the DCD is loaded in OCRAM and must be properly signed.

Since U-Boot v2017.01 a build log containing the U-Boot and DCD addresses and lengths is available just after building U-Boot:

```
$ cat u-boot-dtb.imx.log
  Image Type: Freescale IMX Boot Image
  Image Ver: 2 (i.MX53/6/7 compatible)
  Mode: DCD
  Data Size: 602112 Bytes = 588.00 KiB = 0.57 MiB
  Load Address: 877ff420
  Entry Point: 87800000
  HAB Blocks: 877ff400 00000000 0008ec00
  DCD Blocks: 00910000 0000002c 000001c4
```

Users can copy the information above to create their CSF Authenticate Data command:

```
Block = 0x877ff400 0x00000000 0x0006DC00 "u-boot-dtb.imx", \
        0x00910000 0x0000002c 0x000001c4 "u-boot-dtb.imx"
```

Alternatively users can also extract the DCD length from the DCD table header:

```
$ od -x -j 0x2c -N 4 --endian=big u-boot-dtb.imx
  0000054 d201 c440
  0000060
  DCD Header: 0xd2, DCD Length: 0x01c4, DCD Version: 0x40
```

For the i.MX devices not supporting the skip DCD command (i.MX6Dual/Quad and i.MX6Sololite) the pointer to the DCD table is cleared in the IVT in order to prevent the HAB library from processing the DCD table again during the authentication process. There is no need to re-initialize memory when it already contains valid data.

Since the IVT is modified when downloading to the target the binary must be signed with a cleared DCD pointer. However, the binary must be provided with a valid DCD pointer to allow the UUU tool to locate the DCD table.

The following script can be used to used to handle the DCD pointer:

```
#!/bin/bash
# DCD address must be cleared for signature, as UUU will clear it.
if [ "$1" == "clear_dcd_addr" ]; then
  # store the DCD address
  dd if=$2 of=dcd_addr.bin bs=1 count=4 skip=12
  # generate a NULL address for the DCD
  dd if=/dev/zero of=zero.bin bs=1 count=4
  # replace the DCD address with the NULL address
  dd if=zero.bin of=$2 seek=12 bs=1 conv=notrunc
fi
# DCD address must be set for mfgtool to localize the DCD table.
if [ "$1" == "set_dcd_addr" ]; then
  # restore the DCD address with the original address dd
  if=dcd_addr.bin of=$2 seek=12 bs=1 conv=notrunc
  rm zero.bin
fi
```

The steps below can be used as an example:

```
$ ./mod_4_mfgtool.sh clear_dcd_addr u-boot-dtb.imx
$ ./cst --i u-boot-csf.txt --o u-boot-csf.bin
$ ./mod_4_mfgtool.sh set_dcd_addr u-boot-dtb.imx
```

For the i.MX devices supporting the skip DCD command (i.MX7D, i.MX6UL/ULL, i.MX8MQ and i.MX7ULP), there is no need to do any modification, UUU tool can download the binary directly.

NOTICE: For i.MX7D only, due to an erratum, the UUU download DCD address (0x00911000) is not aligned with the DCD address in u-boot (0x00910000), there are two options:

- use the command to specify the DCD address, uuu boot -f u-boot-signed.imx -dcdaddr 0x00911000
- change the csf DCD address when signing the u-boot: Blocks = 0x00911000 0x0000002c 0x000001c4 "u-boot-dtb.imx

### 6.3 SDPU\SDPV: uboot implement simplified ROM SDP protocol

Uboot implemented i.MX 6/7 ROM SDP protocol. The support command the same as SDP.

SDPV is upgrade version of SDPU, which support -skipspl option for write command

See below for uboot requirement uboot-config-requirement

### 6.4 SDPS: i.MX8QXP and i.MX8QM ROM download protocol

send image by sdp command.

```
boot  -f  <filename> [-offset 0x0000]
```

## 6.5 FB: Android fastboot protocol

refer fast boot protocol

See below for uboot requirement

### 6.5.1 Support command:

```
getvar
ucmd <any uboot command>
acmd <any never returned uboot command, like booti, reboot>

# partition "all" means whole device.
flash [-raw2sparse] <partition> <filename>
download -f <filename>
crc -f <filename> [-format "mmc read $loadaddr"]
                  [-blksz 512][ -crcblock 0x4000000]
                  [-seek 0] [-skip 0] [-nostop]
                    seek          skip block number from storage
                    skip          skips byte from -f
                    nostop        continue check even if found mismatch
```

**Some Uboot command need long time to finish. Default FB timeout is 2s. You can use below method to change timeout value

```
# time out set to 10000ms
FB[-t 10000]: ucmd <any uboot command>
```

Table 5: Table Fastboot environment

| Variable | Description |
|---|---|
| fastboot_dev | fastboot flash device, support mmc and sata |
| fastboot_buffer | fastboot download buffer adddress |
| fastboot_bytes\filesize | fastboot download file size, new uboot provide filesize environment |
| emmc_dev | eMMC device number |
| sd_dev | sd slot device number |

## 6.6 FBK: Android fastboot protocol, implement at initramfs. See project imx-uuc

### 6.6.1 Support command:

```
ucmd <any kernel command> and wait for command finish
acmd <any kernel command> don't wait for command finish
sync                      wait for acmlid processs finish.
ucp <soure> <destinate>   copy file from/to target
                          T:<filename> means target board file.
                          T:- means copy data to target's stdio pipe.
                          copy image T:/root/image ;download image to path /root/ ↩
                             image
                          copy T:/root/image image ;upload /root/image to file  ↩
                             image......
```

Example for transfer big file

```
acmd tar -            ; run tar background and get data from stdio
ucp rootfs.tar.gz T:- ; send to target stdio pipe
sync                  ; wait for tar process exit.
```

Linux environment:

Each command in separate process so environment can not be affect next command. Use below method to workaround this problem.

```
FBK: ucmd source /tmp/mtd.sh; flash_erase /dev/mtd${nandrootfs} 0 0
```

### 6.7 Common command for all protocol

Table 6: Table Common command for all protocol

| Command | Description |
|---|---|
| Done | last command for finish whole flow. |
| Delay | Busy wait for millisecond |
| SH/SHELL | run external command |
| < | stdout as command, such as "< echo ucmd print", which generally used for burn serial number |
| @ | using environment variable, such as "FBK:@ ucmd echo @MSG@" MSG will be replace by OS's environment variable. |
| error | show error message and exit |
| if | basic logic check, such as SDPS: if @chip@ == mx8qm then boot -f flash.bin. Only support == and !=. only support one logic check |

## 7 Migration from mfgtool ucl2.xml

**Most case user can use uboot fastboot protocol to finish image program work.

In case you have to load kernel to burn whole image.

The below simple map ucl2.xml to uuu script.

| ucl2.xml | uuu |
|---|---|
| <CMD state="BootStrap" type="boot" body="BootStrap" file ="firmware/flash.bin" ifdev="MX8QXPB0">Loading boot image</CMD> | SDPS: boot -f flash.bin |
| <CMD state="BootStrap" type="boot" body="BootStrap" file ="firmware/flash.bin" ifdev="MX8MM">Loading U-boot</CMD> | SDP: boot -f flash.bin SDP: boot -f flash.bin<br>SDPU: write -f flash.bin -offset 0x57c00<br>SDPU: jump |
| <CMD state="BootStrap" type="load" file="firmware/Image" address="0x80280000" loadSection="OTH" setSection="OTH" HasFlashHeader="FALSE" ifdev="MX8QXP MX8QM">Loading Kernel.</CMD> | FB: ucmd download -f Image |

| | |
|---|---|
| <CMD state="BootStrap" type="load" file="firmware/initramfs.cpio.gz.uboot" address="0x83800000" loadSection="OTH" setSection="OTH" HasFlashHeader="FALSE" ifdev="MX8QM MX8QXP">Loading Initramfs.</CMD> | FB: ucmd download -f initramfs.cpio.gz.uboot |
| <CMD state="BootStrap" type="load" file="firmware/fsl-imx8qxp.dtb" address="0x83000000" loadSection="OTH" setSection="OTH" HasFlashHeader="FALSE" ifdev="MX8QXP">Loading device tree.</CMD> | FB: ucmd download -f fsl-imx8qxp.dtb |
| <CMD state="BootStrap" type="jump" > Jumping to OS image. </CMD> | |
| <!-- create partition --> <CMD state="Updater" type="push" body="send" file="mksdcard.sh.tar">Sending partition shell</CMD> | FBK: ucp mksdcard.sh.tar t:/tmp |
| <CMD state="Updater" type="push" body="$ tar xf $FILE "> Partitioning...</CMD> | FBK: ucmd tar xf /tmp/mksdcard.sh.tar -d /tmp |
| <CMD state="Updater" type="push" body="$ sh mksdcard.sh /dev/mmcblk%mmc%"> Partitioning...</CMD> | FBK: ucmd mksdcard.sh /dev/mmcblk0mmc |
| <!-- burn uboot --> <CMD state="Updater" type="push" body="send" file="files/imx-boot-imx8qxp-sd.bin" ifdev="MX8QXPB0">Sending u-boot.bin</CMD> | FBK: ucp imx-boot-imx8qxp-sd.bin t:/tmp |
| <CMD state="Updater" type="push" body="$ dd if=/dev/zero of=/dev/mmcblk0 bs=1k seek=4096 conv=fsync count=8">clear u-boot arg</CMD> | FBK: ucmd dd if=/dev/zero of=/dev/mmcblk0 bs=1k seek=4096 conv=fsync count=8 |
| <CMD state="Updater" type="push" body="$ dd if=$FILE of=/dev/mmcblk0 bs=1k seek=33 conv=fsync" ifdev="MX8QM MX8QXP MX8MQ">write u-boot.bin to sd card</CMD> | FBK: ucmd dd if=/tmp/imx-boot-imx8qxp-sd.bin of=/dev/mmcblk0 bs=1k seek=33 conv=fsync |
| <CMD state="Updater" type="push" body="$ while [ ! -e /dev/mmcblk0p1 ]; do sleep 1; echo waiting...; done ">Waiting for the partition ready</CMD> | FBK: ucmd while [ ! -e /dev/mmcblk%mmc%p1 ]; do sleep 1; echo waiting...; done |
| <CMD state="Updater" type="push" body="$ mkfs.vfat /dev/mmcblk0p1">Formatting rootfs partition</CMD> | FBK: ucmd mkfs.vfat /dev/mmcblk0p1 |
| <CMD state="Updater" type="push" body="$ mkdir -p /mnt/mmcblk0p1"/> | FBK: ucmd mkdir -p /mnt/mmcblk0p1 |
| <CMD state="Updater" type="push" body="$ mount -t vfat /dev/mmcblk0p1 /mnt/mmcblk0p1"/> | FBK: ucmd vfat /dev/mmcblk0p1 /mnt/mmcblk0p1 |
| <!-- burn zImage --> <CMD state="Updater" type="push" body="send" file="files/Image">Sending kernel</CMD> | FBK: ucp Image t:/tmp |
| <CMD state="Updater" type="push" body="$ cp $FILE /mnt/mmcblk0p1/Image">write kernel image to sd card</CMD> | FBK: ucmd /tmp/Image /mnt/mmcblk0p1/Image |
| <!-- burn dtb --> <CMD state="Updater" type="push" body="send" file="files/fsl-imx8qxp.dtb" ifdev="MX8QXP MX8QXPB0">Sending Device Tree file</CMD> | FBK: ucp fsl-imx8qxp.dtb /tmp |
| <CMD state="Updater" type="push" body="$ cp $FILE /mnt/mmcblk0p1/fsl-imx8qm.dtb" ifdev="MX8QM">write device tree to sd card</CMD> | FBK: ucmd cp /tmp/fsl-imx8qxp.dtb /mnt/mmcblk0p1/ |
| <CMD state="Updater" type="push" body="$ umount /mnt/mmcblk0p1">Unmounting vfat partition</CMD> | FBK: ucmd umount /mnt/mmcblk0p1 |
| <!-- burn rootfs --> <CMD state="Updater" type="push" body="$ mkfs.ext3 -F -j /dev/mmcblk0p2">Formatting rootfs partition</CMD> | FBK: ucmd mkfs.ext3 -F -j /dev/mmcblk0p2 |

| | |
|---|---|
| <!-- burn rootfs --> <CMD state="Updater" type="push" body="$ mkfs.ext3 -F -j /dev/mmcblk0p2">Formatting rootfs partition</CMD> | FBK: ucmd mkfs.ext3 -F -j /dev/mmcblk0p2 |
| <CMD state="Updater" type="push" body="$ mkdir -p /mnt/mmcblk0p2"/> | FBK: ucmd mkdir -p /mnt/mmcblk0p2 |
| <CMD state="Updater" type="push" body="$ mount -t ext3 /dev/mmcblk0p2 /mnt/mmcblk0p2"/> | FBK: ucmd mount -t ext3 /dev/mmcblk0p2 /mnt/mmcblk0p2 |
| <CMD state="Updater" type="push" body="pipe tar -jxv -C /mnt/mmcblk0p2" file="files/rootfs.tar.bz2">Sending and writting rootfs</CMD> | FBK: acmd tar -jxv -C /mnt/mmcblk0p2<br>FBK: ucp rootfs.tar.bz2 t:- |
| <CMD state="Updater" type="push" body="frs">Finishing rootfs write</CMD> | FBK: sync |
| <CMD state="Updater" type="push" body="$ umount /mnt/mmcblk0p2">Unmounting rootfs partition</CMD> | FBK: ucmd umount /mnt/mmcblk0p2 |
| <CMD state="Updater" type="push" body="$ echo Update Complete!">Done</CMD> | done |

# 8 Win 7 User Guide

**Win7 user may face some additional one time setup work because original win7 missed a updated .inf file**

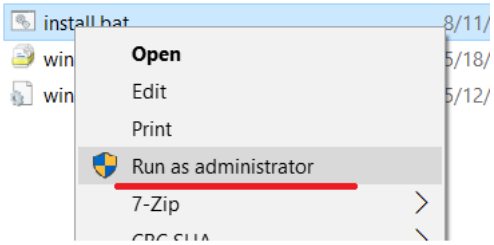
## 8.1 Back Ground

Win7 ships with correct *winusb.sys* file. but is missing an updated *.inf* that associates with "usb\ms_comp_winusb" devices. Normally if the USB device supports Microsoft OS descriptors, then it will allow Windows to automatically install the WinUSB driver. This mechanism is supported "in-box" for Win8 and newer. For Win7 the mechanism is supported through Windows update. Depending on the update policy for the Win7 machine, the appropriate driver may or may not be already available on the machine. If it is not already on the machine, user can use the following manual procedure to install the driver if necessary. (copy from https://www.silabs.com/community/interface/knowledge-base.entry.html/2017/02/06/manually_installwin-A2Jj")

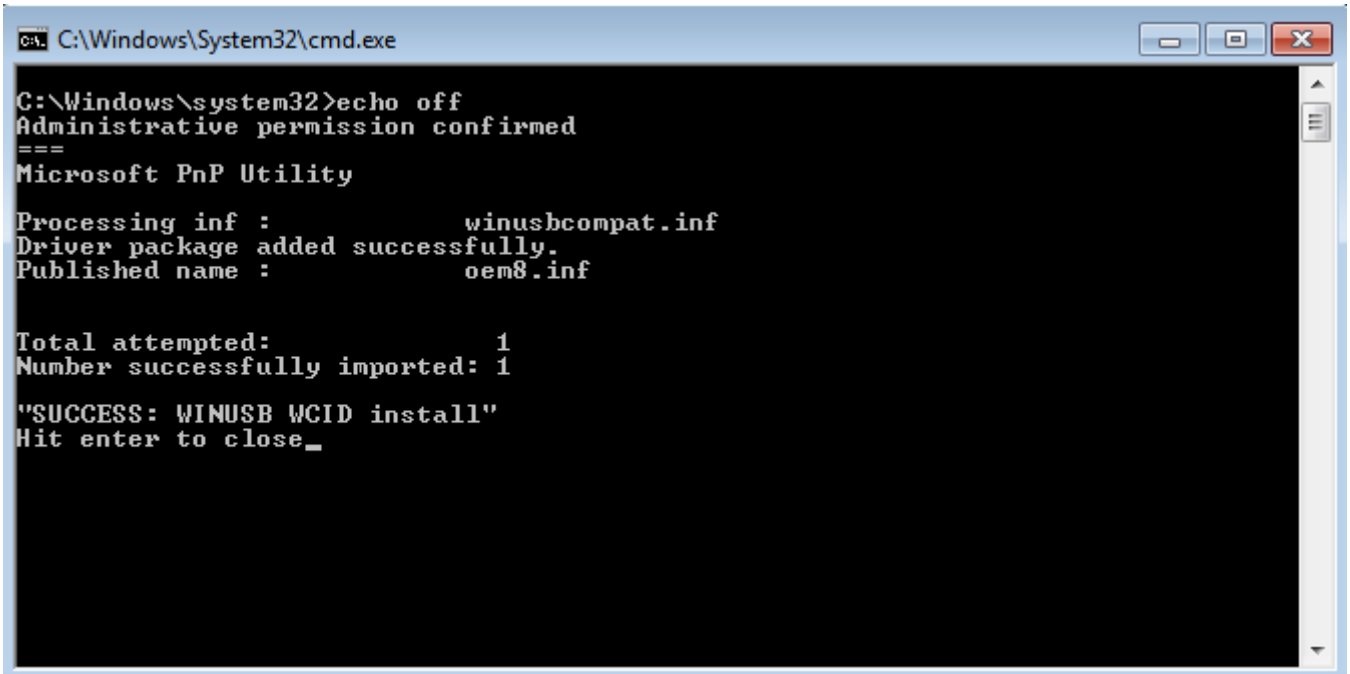
Some windows update also included updated .inf file. You can try run uuu to see what happen. If windows report "can't install driver", that means your system missed such update file.

## 8.2 Install updated winusb inf file

- Download package
- unzip
- run install.bat as administrator permission.




- The below screen show install success




**Notes:**

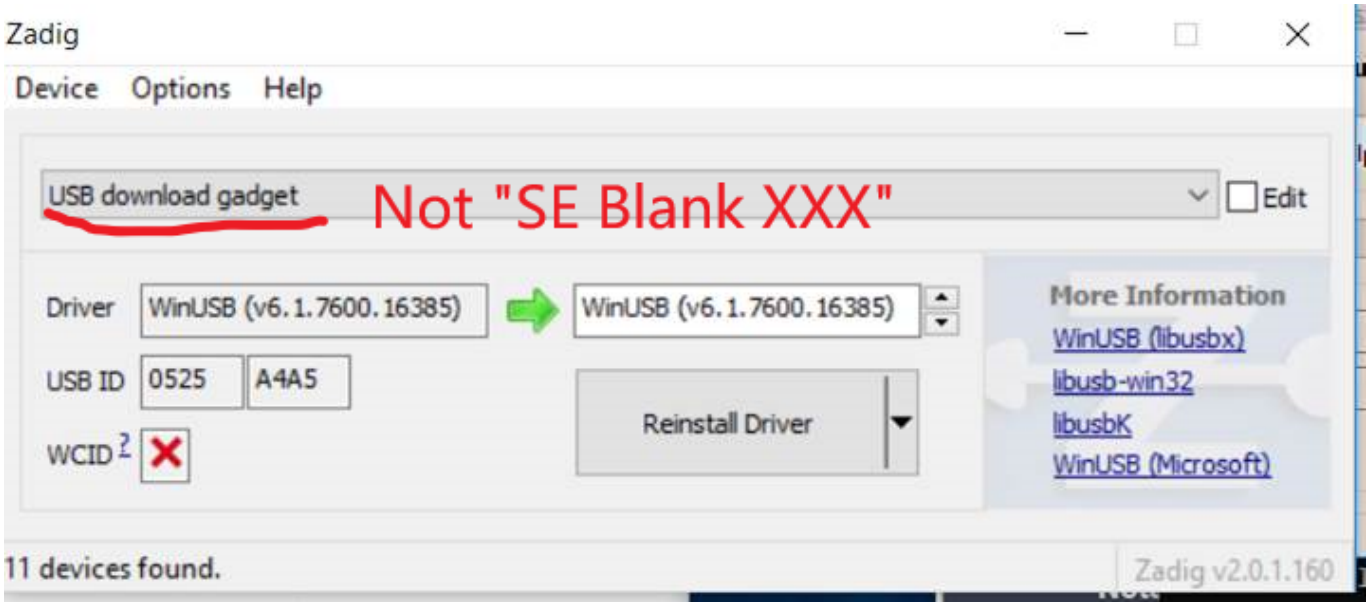
4.9.123 8MM GA and 4.14 beta release missed a patch. Please apply below patch in uboot

### 8.3 Use zadig to install winusb driver

Please make sure you success download uboot and uboot auto launch fastboot command. windows find **Usb download device**, NOT **SE Blank**

If still fail install winusb driver you can try below method. you can try download zadig from https://zadig.akeo.ie/

Choose **USB download device** and click install.




If you already apply patch and still see WCID is red "x", please submit issue.

**Notes:**

"SE Blank xx" is ROM HID device. PLEASE DON'T CHOOSE IT TO WINUSB. libusb can work well with HID device. Only "Usb download device" need check

# 9 FAQ

- **Win7 can't found driver**
  - Need install winusb driver, you can use https://zadig.akeo.ie/ to install winusb driver
  - see page WIN7-User-Guide
- **Linux: Open device failure**

```
sudo uuu xxx
```

- **Some iMX8mm(845) chip failure write at linux system**

```
Need apply ROM patch, contact FAE to get it.
```

- **Failed with higher rate on i.MX8QXP C0 MEK board with Type-C port at linux system**

```
Linux Host PC to force USB driver behavior similar like Windows OS before ↩
    connecting.
echo Y > /sys/module/usbcore/parameters/old_scheme_first
```

- **How to use absolute path in scripts**

```
Default all paths in script is related uuu scripts. if you want to use absolute ↩
    path in scripts

Add ">" in path like


 >/home/xxx
```

- **uuu exit silence or report missed dll**

```
Please upgrade to 1.2.x
```

- **How to avoid sudo in linux**

```
Put below context into /etc/udev/rules.d/99-uuu.rules  (need sudo)
```

```
SUBSYSTEM=="usb", ATTRS{idVendor}=="1fc9", ATTRS{idProduct}=="012f", MODE="0666"
SUBSYSTEM=="usb", ATTRS{idVendor}=="1fc9", ATTRS{idProduct}=="0129", MODE="0666"
SUBSYSTEM=="usb", ATTRS{idVendor}=="15a2", ATTRS{idProduct}=="0076", MODE="0666"
SUBSYSTEM=="usb", ATTRS{idVendor}=="15a2", ATTRS{idProduct}=="0054", MODE="0666"
SUBSYSTEM=="usb", ATTRS{idVendor}=="15a2", ATTRS{idProduct}=="0061", MODE="0666"
SUBSYSTEM=="usb", ATTRS{idVendor}=="15a2", ATTRS{idProduct}=="0063", MODE="0666"
SUBSYSTEM=="usb", ATTRS{idVendor}=="15a2", ATTRS{idProduct}=="0071", MODE="0666"
SUBSYSTEM=="usb", ATTRS{idVendor}=="15a2", ATTRS{idProduct}=="007d", MODE="0666"
SUBSYSTEM=="usb", ATTRS{idVendor}=="15a2", ATTRS{idProduct}=="0080", MODE="0666"
SUBSYSTEM=="usb", ATTRS{idVendor}=="1fc9", ATTRS{idProduct}=="0128", MODE="0666"
SUBSYSTEM=="usb", ATTRS{idVendor}=="1fc9", ATTRS{idProduct}=="0126", MODE="0666"
SUBSYSTEM=="usb", ATTRS{idVendor}=="1fc9", ATTRS{idProduct}=="0135", MODE="0666"
SUBSYSTEM=="usb", ATTRS{idVendor}=="1fc9", ATTRS{idProduct}=="0134", MODE="0666"
SUBSYSTEM=="usb", ATTRS{idVendor}=="1fc9", ATTRS{idProduct}=="012b", MODE="0666"
SUBSYSTEM=="usb", ATTRS{idVendor}=="0525", ATTRS{idProduct}=="b4a4", MODE="0666"
SUBSYSTEM=="usb", ATTRS{idVendor}=="0525", ATTRS{idProduct}=="a4a5", MODE="0666"
SUBSYSTEM=="usb", ATTRS{idVendor}=="066F", ATTRS{idProduct}=="9BFF", MODE="0666"
```

run sudo udevadm control --reload-rules

*uuu -udev* can show above help.

- **How to get file in zip without decompress**

```
Assume there are file name uuu.auto in A.zip file.

A.zip/uuu.auto

for example:

uuu A.zip/uuu.auto
```

- **How to send out uncompress bz2**

```
Added /* after bz2 file

for example

uuu -b emmc_all <bootloader> sdcard.bz2/*
```

- **Boot fail after burn > 4G Image**

```
uboot need below patch


    diff --git a/common/image-sparse.c b/common/image-sparse.c
    index ddf5772..86ff5a0 100644
    --- a/common/image-sparse.c
    +++ b/common/image-sparse.c
    @@ -59,7 +59,7 @@ void write_sparse_image(
            uint32_t bytes_written = 0;
            unsigned int chunk;
            unsigned int offset;
    -       unsigned int chunk_data_sz;
    +       uint64_t chunk_data_sz;
            uint32_t *fill_buf = NULL;
            uint32_t fill_val;
            sparse_header_t *sparse_header;
    @@ -130,7 +130,7 @@ void write_sparse_image(
                                        sizeof(chunk_header_t));
                    }

    -               chunk_data_sz = sparse_header->blk_sz * chunk_header-> ↩
        chunk_sz;
    +               chunk_data_sz = (uint64_t)sparse_header->blk_sz * (uint64_t) ↩
        chunk_header->chunk_sz;
                    blkcnt = chunk_data_sz / info->blksz;
                    switch (chunk_header->chunk_type) {
                    case CHUNK_TYPE_RAW:
```

- **WCID failure load**

Apply below uboot patch

```
diff --git a/drivers/usb/gadget/f_fastboot.c b/drivers/usb/gadget/f_fastboot.c
index ae8fe80..cd46ca4 100644
--- a/drivers/usb/gadget/f_fastboot.c
+++ b/drivers/usb/gadget/f_fastboot.c
@@ -2543,10 +2543,10 @@ static int fastboot_bind(struct usb_configuration *c,  ↩
    struct usb_function *f)
         f->os_desc_table->if_id = id;
         INIT_LIST_HEAD(&fb_os_desc.ext_prop);
         fb_ext_prop.name_len = strlen(fb_ext_prop.name) * 2 + 2;
-        fb_os_desc.ext_prop_len = 14 + fb_ext_prop.name_len;
+        fb_os_desc.ext_prop_len = 10 + fb_ext_prop.name_len;
         fb_os_desc.ext_prop_count = 1;
-        fb_ext_prop.data_len = strlen(fb_ext_prop.data);
-        fb_os_desc.ext_prop_len += fb_ext_prop.data_len;
+        fb_ext_prop.data_len = strlen(fb_ext_prop.data) * 2 + 2;
+        fb_os_desc.ext_prop_len += fb_ext_prop.data_len + 4;
         list_add_tail(&fb_ext_prop.entry, &fb_os_desc.ext_prop);

         id = usb_string_id(c->cdev);
```

- **out of memory at 64bit system**

If use bz2 or zip file, it is possible out of memory. Please increase swap partition size in linux. increase virtual memory size in windows.

- **How to write emmc to second boot partition**

```
FB: ucmd setenv fastboot_dev mmc
FB: ucmd setenv mmcdev $emmc_dev

FB: ucmd echo flashing first parttion

FB: ucmd setenv fastboot_buffer $loadaddr
FB: download -f imx-boot.bin
FB: ucmd setexpr fastboot_blk $fastboot_bytes
FB: ucmd setexpr fastboot_blk $fastboot_blk + 0x1FF
FB: ucmd setexpr fastboot_blk $fastboot_blk / 0x200
FB: ucmd mmc dev $emmc_dev 1
FB: ucmd echo fastboot_buffer: $fastboot_buffer
FB: ucmd echo fastboot_blk: $fastboot_blk
FB: ucmd mmc write $fastboot_buffer 0 $fastboot_blk

FB: ucmd echo flashing second parttion

FB: download -f imx-boot-ALTERNATE.bin
FB: ucmd setexpr fastboot_blk $fastboot_bytes
FB: ucmd setexpr fastboot_blk $fastboot_blk + 0x1FF
FB: ucmd setexpr fastboot_blk $fastboot_blk / 0x200
FB: ucmd mmc dev $emmc_dev 2
FB: ucmd mmc write $fastboot_buffer 0 $fastboot_blk

FB: ucmd mmc partconf $emmc_dev 1 1 0

FB: Done
```

- **How to use uuu in docker**

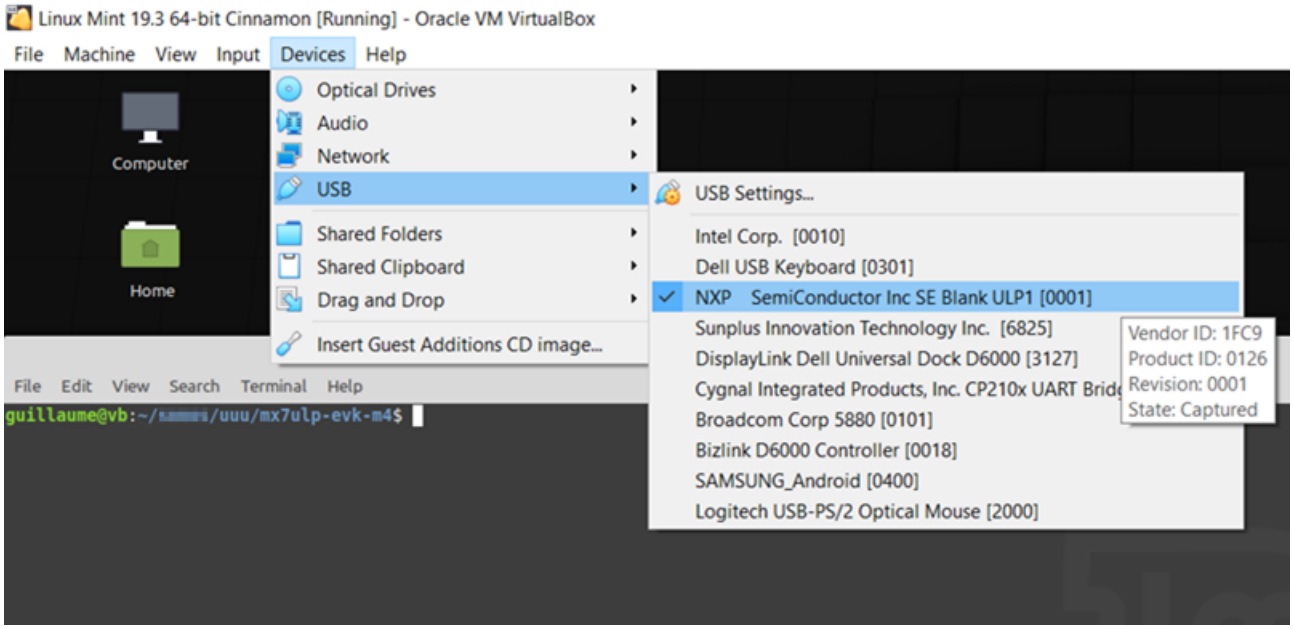
libusb need udev event to get new devices appeared.

Added option

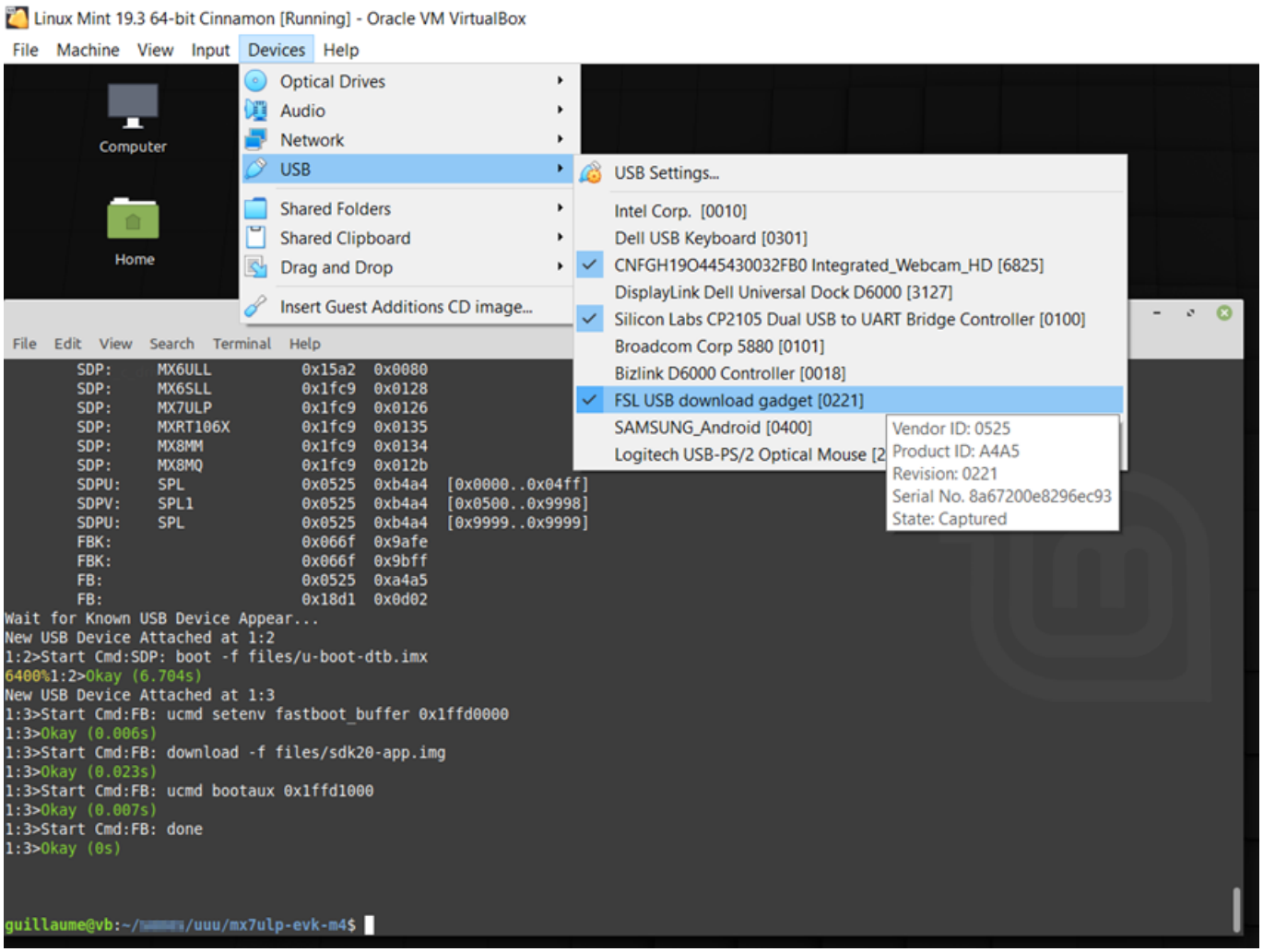
docker run --net=host -v /run/udev/control:/run/udev/control

https://stackoverflow.com/questions/49687378/how-to-get-hosts-udev-events-from-a-docker-container

- **How to use uuu in virtual machine**
  - Map HID device into virtual machine




- Map fastboot device into virtual machine after uboot boot




- You can use uuu burn image in virtual machine

# 10 Build Steps

## 10.1 windows

- download visual studio 2017 community version (free)
- git clone https://github.com/NXPmicro/mfgtools.git
- cd mfgtools
- git submodule init
- git submodule update
- open msvs/uuu.sln by vs2017
- click build

## 10.2 linux

- git clone https://github.com/NXPmicro/mfgtools.git

- cd mfgtools
- sudo apt-get install libusb-1.0.0-dev libzip-dev libbz2-dev
- cmake .
- make

## 11 Uboot config requirement

To talk with uuu, uboot need enable fastboot. fastboot need auto run when detect boot from USB.

```
CONFIG_CMD_FASTBOOT=y
CONFIG_USB_FUNCTION_FASTBOOT=y
CONFIG_USB_GADGET=y
CONFIG_USB_GADGET_DOWNLOAD=y
CONFIG_USB_GADGET_MANUFACTURER="FSL"
CONFIG_USB_GADGET_VENDOR_NUM=0x0525
CONFIG_USB_GADGET_PRODUCT_NUM=0xa4a5
CONFIG_CI_UDC=y                              # UDC need change according system,  ↩
   some system use CONFIG_USB_DWC3, some use CONFIG_USB_CDNS3
CONFIG_FSL_FASTBOOT=y
CONFIG_FASTBOOT=y
CONFIG_FASTBOOT_BUF_ADDR=0x83800000       # Address need change according  ↩
   system, generally it can be the same as ${LOADADDR}
CONFIG_FASTBOOT_BUF_SIZE=0x40000000
CONFIG_FASTBOOT_FLASH=y
CONFIG_FASTBOOT_FLASH_MMC_DEV=1
CONFIG_EFI_PARTITION=y
CONFIG_ANDROID_BOOT_IMAGE=y
```

If use SPL, SDP need be enabled.

```
CONFIG_SPL_USB_HOST_SUPPORT=y
CONFIG_SPL_USB_GADGET_SUPPORT=y
CONFIG_SPL_USB_SDP_SUPPORT=y
CONFIG_SDP_LOADADDR=0x40400000           # Address need change according  ↩
    system, choose free memory
```

uuu related patches.

https://source.codeaurora.org/external/imx/uboot-imx/log/?h=imx_v2017.03_4.9.123_imx8mm_ga

About Fastboot enable:

```
719651a MLK-18257-1 Enable fastboot support in qxp mek board
d5226a3 MLK-18257-2: fix fastboot build warning
219c989 MLK-18257-3 run fastboot if initramfs is in validate
09b1876 MLK-18257-4 use another method check if need run bootcmd_mfg
3b1fa9d MLK-18257-5 enhence fastboot uboot cmd
ca96e0b MLK-18406 fastboot support all partition
```

About uboot SDP enable:

```
192a26d MLK-18707-1: SDP: use CONFIG_SDP_LOADADDR as default load address
9764fb2 MLK-18707-2 iMX8M enable fastboot as default
db9a634 MLK-18862 imx8mm uuu can write emmc by fastboot
```

Additional environment need be define

Table 8: Table uuu environment

| Variable | Usage | Description |
|---|---|---|
| emmc_dev | emmc burn | eMMC device number |
| sd_dev | burn sd | sd slot device number |
| kboot | boot kernel\burn nand | kernel boot command, it is booti for arm64, bootz for arm32 |
| weim_uboot | burn weim nor | uboot burn to position of weim nor |
| weim_base | burn weim nor | weim base address |
| mtdparts | burn nand | NAND flash partition configuration, such as "mtdparts=8000000.nor:1m(boot),-(rootfs)\\;gpmi-nand:64m(nandboot),16m(nandkernel),16m(nand (nandrootfs) " |
| spi_bus | burn spi (not qspi\fspi) | spi nor flash bus number |
| spi_uboot | burn spi (not qspi\fspi) | spi nor flash uboot offset |

# 12 kernel config requirement

Some kernel config required

- USB CONFIG FS need be built-in
- One of UDC driver and PHY need be built-in
- Function FS need be enabled

```
Device Drivers
USB support
       USB gadget support (very last entry)
              USB Gadget Drivers (...)
                     USB functions configurable through configfs
                            Mass storage
                            Function filesystem (functionFS)
```

# 13 Release Notes

## 13.1 1.5.233 (stable)

### 13.1.1 New features

- Update python wrap build to support arm64 build

## 13.2 uuu_1.5.222

### 13.2.1 bug fixes

- script spi: check fastboot_bytes not fastboot_buffer to get the image size
- script spi: only run the fastboot loop when downloading the large image.
- Accessing freed memory in fastboot.c when use zip

## 13.3 uuu_1.5.220

### 13.3.1 Bug fixes

- build for ubuntu 20.04 for old server

## 13.4 uuu_1.5.219

### 13.4.1 Bug fixes

- fix python wrapper work action issue

## 13.5 uuu_1.5.218

### 13.5.1 new feature

- build-in script spi-nor support big files.
- update libusb to v1.0.29.

### 13.5.2 Bug fixes

- libuuu: Fix typo & compilation warning in error message.
- fix: memory corruption in libuuu fastboot.
- fix: change the return type of `load_bmap` to an integer.
- fix: if-command comparison code by use uppercase.

## 13.6 uuu_1.5.201 (stable)

### 13.6.1 new feature

- Support imx943 v2x image container

### 13.6.2 Bug fixes

- Reduce transfer size to 16k because chipidea usb driver limitation
- Fix workflow pdf build error
- Fix workflow arm build fail

## 13.7 uuu_1.5.197

### 13.7.1 new feature

- Add iMX943 support

## 13.8 uuu_1.5.191

### 13.8.1 new feature

- Add python binding

## 13.9 uuu_1.5.182 (stable)

### 13.9.1 his is hot fix for 1.5.181.

- Fix "imx91" device name as "imx93" when -lsusb

## 13.10 uuu_1.5.181

This is hot fix for 1.5.179. - Fix crash when get serial number at SPL/uboot if no serial_no

## 13.11 uuu_1.5.179

## 13.12 New feature

- Provide macos arm prebuild version.

## 13.13 1.5.177

### 13.13.1 New features

- Support i.MX91 ==== Bug fixes
- Fixed a crash with -d option when change back file

## 13.14 1.6.165 (stable)

This is hot fix for 1.5.163. Static link tinyxml to avoid difference version ubuntu tinyxml version mismatch problem.

## 13.15 1.5.163

### 13.15.1 New features

- Support yocto bmap file to skip unused block
- Support i.MX95

### 13.15.2 Bug fixes

- Fixed usb pipe error when download image with FCB

## 13.16 1.5.125 (stable)

This is fixes up for 1.5.120

- force use old libusb to run at ubuntu 18.04

## 13.17 1.5.120

### 13.17.1 New features

- Add built spinand command support
- parser [-t timeout] for all protocol

## 13.18 1.5.109

This is fixes up for 1.5.104

- static link c++ library in prebuild linux image
- provide armlinux version image

## 13.19 1.5.104

### 13.19.1 New fetaures

- Using github working flow action to do release.
- Expand upload command, like uploading the value of a specific variable.

### 13.19.2 Bug fixes

- Fix segfault when boot data is zero in SDP.
- Fix issues when m_skip is not zero in fastboot.
- Fix issues when file size isn't divisible by block size in fastboot.
- Fix read uploaded data for FB.

## 13.20 1.5.21 (stable)

### 13.20.1 New features

- Reduce memory usage when burn single board with compressed Image.
- Http(s) prompt request username and passwords when require autherization.

### 13.20.2 Bug fixes

- fixed android super.img loop download 52byte after some block.
- fix memory leak cause by getaddrinfo.
- fix support of empty environment variables in Windows OS.
- fix crash when environment variable is last word in script

## 13.21 1.4.243

### 13.21.1 New features

- Add zstd support
- Add iMX93 support

### 13.21.2 Bug fixes

- Fixed random crash when download bz2 from http
- Fix crash when download speed slower than decompress speed and need resize buffer

## 13.22 1.4.193

### 13.22.1 New features

- Add NXP FB/FBK/SDPV device PID

### 13.22.2 Bug fixes

- Fixed 314 Corrupt image with larger transfer sizes at sdp(s)
- Fixed missed true at else branch at built-in fat_write script and nand_burn_loader
- Fixed invalid progress percentage in verbose mode
- Fixed QSPI flashing script
- Fix #297 print error when run quit cmd in shell mode
- Fix #180 support check getval return value

## 13.23 1.4.165

### 13.23.1 New features

- Burn wic file(after 2021 Q4 yocto image) directly. uuu -b emmc_all wic
- Add *upload* fastboot command support
- Support uboot that use plugin initialize ddr
- Support simple logical check in uuu script

### 13.23.2 Bug fixes

- fix buildin script bz2 have not replaced with bz2* if path included space
- fixed #284: deadloop when using built-in script with filename include _
- Fix a potential crash if open usb device 400ms after detect attached
- fix uuu return success even ucp failure

## 13.24 1.4.139

### 13.24.1 New features

- Add help option -IgSerNum to set windows registry to ignore serial number above too much entry in register when burn many boards.
- Add snap support
- Add 8ulp support
- Deploy mac image

### 13.24.2 Bug fixes

- Fixed missed last chunk when meet specific android sparse file
- return out of memory when allocate failure
- -bshow don't show uuu version information and recover cursor ESC

## 13.25 1.4.72

### 13.25.1 New features

- http\https add port support at url
- improve progress show when burn android sparse image

### 13.25.2 Bug fixes

- Fix the dead loop when flash specific android super.img. There are bug when split big super.img into small sparse file running time. The bug trigger if split point at whole sparse chuck.

## 13.26 1.4.43

### 13.26.1 New features

- add environment variable support command
- Added support for *FB:reboot* and *FASTBOOT:reboot*
- allow to pass sparse limit from command line
- Update version number support build number > 255
- Mac OS (Catalina) support

### 13.26.2 Bug fixes

- fix #193 fb:< echo ucmd print failure
- fix fat_write partition parser error

## 13.27 1.3.191

### 13.27.1 New features

- Added Boundary Devices fastboot IDs

### 13.27.2 Bug fixes

- Workaround problem: some data pattern cause timeout at dell docker hub.
- update 7D free memory address, errata e11166 https://www.nxp.com/docs/en/errata/IMX7DS_3N09P.pdf
- Fix can't build from source tarball.

## 13.28 1.3.154

### 13.28.1 New features

- ffu image basic support
- add boundary device VID/PID
- Added basic fastboot logical partition support

### 13.28.2 Bug fixes

- fastboot: fix oem command separator
- Fix crash when download bz2 file

## 13.29 1.3.134

### 13.29.1 Bug fixes

- fix qm\qxp download failure if use spl image

## 13.30 1.3.130

### 13.30.1 New features

- Add android fastboot continue command
- Add i.MX8DXL support
- Add i.MX865 support
- build in script sd, support burn difference uboot
- Add option -pp to set pool time
- sdp: bootcmd: add support for --dcdaddr
- change the default nandbcb command from update to init

#### 13.30.2 Bug fixes

- fix one line miss when exit uuu
- fix show 99% when use bz2 file
- fix miss ucmd at nand built-in script
- fix show time wrong, use system time instead of tick as timesample

### 13.31 1.3.102

#### 13.31.1 New features

- Support simple https. (experimental).
- -b option support script file
- support >=4G zip file

#### 13.31.2 Bug fixes

- fixed #136 fix decompress failure if use uncompressed method in zip file.
- fixed something wait forever when download bz2 from http
- fix http request failure at some host (use \r\n at http request)

### 13.32 1.3.82

#### 13.32.1 New features

- add dry-run option to check if all files exist at script
- Start download before scan whole Bz2 file. Progress only show how many data burned instead of percentage before finish scan.
- Add NAND built in script to burn boot loader into nand flash, which need uboot support nandbcb command
- Add --skipfhdr option to skip flexspi header
- Can use SD card uboot for flexspi, which need uboot support qspihdr command
- Add timestamp info for each command. (-v show how many time is consumed for each command).
- Add tar.bz2 support.(experimental).
- Add tar.gz support. (experimental).
- Add http download. (experimental).
- Auto append /* for bz2 file when use built-in script.
- Add -lsusb option to list all connected known devices to help find usb path when do multi boards support

#### 13.32.2 Bug fixes

- Fix SPL download uboot failure if uboot size > 2M
- Upgrade libusb to 1.0.23-rc3 to resolve some windows compatibility problem. such as exit if not usb port under virtual root hub.
- Fix bz2 decompress problem if bz2 created by bzip2 instead of pbzip2.
- Fix missed last chunk data for android sparse image
- fixed #123: implement timeout for wait known usb device appear

### 13.33 1.2.135

#### 13.33.1 New features

- Support i.MX28
- Add Read\write a memory address for i.MX6/i.MX7
- built-in script emmc support burn difference files
- Support i.MX815

#### 13.33.2 Bug fixes

- fix crash when console width between 47 to 54
- fix crash when use ssh <host> uuu
- fix wrong data by decompress sdcard.bz2/* if enable -O2 build option

### 13.34 1.2.91

#### 13.34.1 New features

- Auto parameter complete support
- Add option -udev to help create udev rule to avoid use sudo
- Remove VT color at win7
- Fail back to verbose mode at win7
- Enable uboot shell mode
- Just print help when run uuu and doesn't scan auto.uuu in current directory.
- Added blog command to fetch message from uboot boot log
- Support file path include space. need add "" at file path
- Windows version built-in libusb
- New SDPV support to support -skipsql for uboot, which support auto scan uboot position.

#### 13.34.2 Bug fixes

- Fixed uuu "fb[-t 1000]:" ucmd wait forever problem
- Fixed missed sdpu: done at spl built in script
- Fixed show nothing when wait for usb connection
- Fixed random claim interface failure at windows platform
- Fixed random crash when multi-device download at windows platform
- Fixed memory leak.
- Fixed #79 file all zero when ucp from target to host
- Fixed crash when done is not last cmd

### 13.35 1.2.0

#### 13.35.1 New features

- Support decompress bz2 file, sdcard.bz2/* means decompress it.
- Support enter shell after run script

#### 13.35.2 Bug fixes

- Fix mx6 boot failure when enable security
- Fix windows version dependent on vs redistribute package

### 13.36 1.1.81

#### 13.36.1 New features

- Support shell command
- Support shell command generate dynamic uuu command to burn sequence id, like MAC address
- Reduce cpu usage rate at windows platform by increase each bulk transfer size
- Added q(quit) to exit shell at -s mode

#### 13.36.2 Bug fixes

- fixed some typo
- fixed build script qspi burn failure if file size > 1M
- fixed file locked by uuu to prevent user update it.
- fixed crash at special uboot size

### 13.37 1.1.41

#### 13.37.1 New features

- Support SDP protocol (i.MX6x, i.MX7, i.MX8M, i.MX8MM)
- Support SDPS protocol (i.MX8QXP B0, i.MX8QM B0)
- Support SDPU protocol (uboot\SPL implemented SDP protocol)
- Support FB protocol (fastboot with uboot)
- Support FBK protocol (fastboot in kernel, daemon implement at imx-uuc)
- Support multi-device download
- Support built-in script

#### 13.37.2 Bug fixes

- Fix sometime open usb device failure at windows platform.
- Fix protocol case sensitive problem in script
- Fix open zip file failure

## 14 Known issue

- Some old i.MX8MM board HID can't work in linux system. Need apply ROM patch to fix. Please contact FAE.
- QXP/QM NAND image can't download by UUU because alignment requirement is difference
- Bz2 file only support one that is generated by pbbz2. there are problem if use bzip2 generate bz2.
- i.MX815 only support 3 boards at the same time.
- Some typeC dell dock USB hub cause transfer timeout if meet some magic data pattern, see branch magic_pattern. Try use differece hub or direct connect to PC usb port if this happen.