# UUU (Universal Update Utility)

| REVISION HISTORY | | | |
|---|---|---|---|
| NUMBER | DATE | DESCRIPTION | NAME |
| 510af57 | Tue Oct 30 11:57:16 2018 -0500 | Updated Start (asciidoc) | nxpfrankli |
| d9128ca | Tue Oct 30 12:07:04 2018 -0500 | Updated Start (asciidoc) | nxpfrankli |
| 05943d3 | Tue Oct 30 12:08:53 2018 -0500 | Updated Example (asciidoc) | nxpfrankli |
| c0243e0 | Tue Oct 30 12:12:59 2018 -0500 | Updated Example (asciidoc) | nxpfrankli |
| aa5643d | Tue Oct 30 13:49:01 2018 -0500 | Updated Start (asciidoc) | nxpfrankli |
| 927034f | Tue Oct 30 13:51:28 2018 -0500 | Updated Example (asciidoc) | nxpfrankli |
| 5dac433 | Tue Oct 30 14:53:27 2018 -0500 | Updated migration from ucl2.xml to uuu script (asciidoc) | nxpfrankli |
| 8db89fc | Tue Oct 30 14:57:35 2018 -0500 | Updated Start (asciidoc) | nxpfrankli |
| 870f770 | Tue Oct 30 15:00:37 2018 -0500 | Updated Start (asciidoc) | nxpfrankli |
| 1317290 | Tue Oct 30 15:03:31 2018 -0500 | Updated UUU default support protocol list (asciidoc) | nxpfrankli |
| 7596a8b | Tue Oct 30 15:06:02 2018 -0500 | Updated Release Notes (asciidoc) | nxpfrankli |
| f5530e1 | Tue Oct 30 16:42:33 2018 -0500 | Updated Home (asciidoc) | nxpfrankli |
| c55d515 | Tue Oct 30 16:53:02 2018 -0500 | Updated Home (asciidoc) | nxpfrankli |
| ff95e5b | Tue Oct 30 17:01:38 2018 -0500 | Updated Home (asciidoc) | nxpfrankli |
| effb56a | Tue Oct 30 17:46:17 2018 -0500 | Updated Example (asciidoc) | nxpfrankli |
| 66b281c | Tue Oct 30 17:47:27 2018 -0500 | Updated Example (asciidoc) | nxpfrankli |
| 226b303 | Wed Oct 31 10:45:16 2018 -0500 | Updated Home (asciidoc) | nxpfrankli |

| REVISION HISTORY | | | |
|---|---|---|---|
| NUMBER | DATE | DESCRIPTION | NAME |
| 160f914 | Wed Oct 31 10:53:40 2018 -0500 | Updated FAQ (asciidoc) | nxpfrankli |
| e6e83a6 | Wed Oct 31 10:56:33 2018 -0500 | Updated FAQ (asciidoc) | nxpfrankli |
| b44c986 | Wed Oct 31 14:09:00 2018 -0500 | format uuu output to fit short width | nxpfrankli |
| 1149fa8 | Wed Oct 31 14:10:03 2018 -0500 | bold command line | nxpfrankli |
| c7d1c57 | Wed Oct 31 14:19:40 2018 -0500 | update link to fix show ? in pdf | nxpfrankli |
| 65e8bf8 | Wed Oct 31 14:21:34 2018 -0500 | Added QXP/QM NAND image can’t download by UUU | nxpfrankli |
| daf3554 | Wed Oct 31 14:22:57 2018 -0500 | Up known issue section | nxpfrankli |
| 70c1114 | Wed Oct 31 16:22:32 2018 -0500 | Updated Release Notes (asciidoc) | nxpfrankli |

# Contents

# 1 Introduction

Welcome to the UUU (Universal Update Utility). This is an evolution of MFGTools (aka MFGTools v3).

UUU is Freescale/NXP I.MX Chip image deploy tools.

With the time, the need for an update utility portable to Linux and Windows increased. UUU have the same usage on both Windows and Linux. It means the same script works on both OS.

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 design as common library and UI. So user can easily integrate into their tools with uuu library. UUU also easy run in any scripts.

PDF of wiki content also is available at release page.

## 1.1 Running environment

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

**Windows 7 user need install winusb driver. A application can do that. https://zadig.akeo.ie/**

## 1.2 Typical Usage

Set board boot pin to serial download mode. Generally iMX ROM will fail back to usb serial download mode if boot failure.

Download uboot

```
uuu bootloader
```

Burn uboot into emmc

```
uuu -b emmc bootloader
```

Burn rootfs image into emmc

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

Decompress rootfs image and burn into emmc (since 1.1.87)

```
uuu -b emmc_all bootloader rootfs.sdcard.bz2/*
```

Burn release image into emmc

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

More usage please refer Example

## 1.3 Typical Script

uuu's script is plain text file

**first line must be

```
uuu_version 1.0.1
```

Version show minimize version of uuu to run this script.

Then flow uuu commands.

UUU command format as

PROTOCOL: CMD

The below is example to boot uboot for imx6 and imx7.

```
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
```

more sample scripts see Sample-script

The below table environment may be used when write uuu script

Table 1: Table 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 |

## 1.4 What Firmware Need

| **What you want** | **Firmware Need** |
|---|---|
| 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) prefer enable fastboot. If ROM HID support write additional image to DDR place, you can write kernel\dtb\initramfs to ddr before jump to uboot. Enable fastboot give more flexibility to change kernel command line.

## 1.5 L4.9.123_2.3.0_8MM GA

It is first official BSP release to support uuu For L4.9.123_2.3.0_8MM GA with i.MX8M Mini, see [?]

## 2 Syntactic

```
uuu (Universal Update Utility) for nxp imx chips -- libuuu_1.1.87-7-gad2ec1f

uuu [-d -m -v -V] <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          Deamon mode, wait for forever.
    -v -V       verbose mode, -V enable libusb error\warning info
    -m          USBPATH Only monitor these pathes.
                    -m 1:2 -m 1:3

uuu -s          Enter shell mode. uuu.inputlog record all input commands
                you can use "uuu uuu.inputlog" next time to run all commands

uuu -h -H       show help, -H means detail helps

uuu [-d -m -v] -b[run] <emmc|emmc_all|qspi|sd|sd_all|spl> arg...
        Run Built-in scripts
        emmc    burn boot loader to eMMC boot partition
                arg0: _flash.bin
        emmc_all        burn whole image to eMMC
                arg0: _flash.bin
                arg1: _rootfs.sdcard
        qspi    burn boot loader to qspi nor flash
                arg0: _flexspi.bin  bootloader
                arg1: _image[Optional]  image burn to flexspi, default is the s  ↩
                                                ame as bootloader
        sd      burn boot loader to sd card
                arg0: _flash.bin
        sd_all  burn whole image to sd card
                arg0: _flash.bin
                arg1: _rootfs.sdcard
        spl     boot spl and uboot
                arg0: _flash.bin

uuu -bshow <emmc|emmc_all|qspi|sd|sd_all|spl>
        Show built-in script

 ------------------------------------------------------------------------
 Command Format PROTOCOL COMMAND ARG
 PROTOCOL
          ALL protocol supported common command
                      done        #last command for whole flow
                      delay <ms>  # delay ms
                      sh\shell <any shell command> #Run shell command, such as  ↩
                                                     wget to file from network
                      <        <any shell command> #use shell command's output  ↩
                                                     as uuu command
                               this command generally used for burn some sequen  ↩
                                                          ce number, such  ↩
```

```
                                production id, mac address
                              for example:
                                    FB:< echo ucmd print

        CFG:  Config protocol of specific usb device vid/pid
              SDPS|SDP|FB\Fastboot|FBK -chip <chip name> -pid <pid> -vid <vid  ↩
                                           > [-bcdversion <ver>]

        SDPS: Stream download after MX8QXPB0
                    boot  -f  <filename> [-offset 0x0000]

        SDP:  iMX6/iMX7 HID download protocol.
                    dcd   -f  <filename>
                    write -f  <filename> [-addr 0x000000] [-ivt 0]
                    jump  -f  <filename> [-ivt 0]
                    boot  -f  <filename> [-nojump]

        FB[-t timeout]:\Fastboot: android fastboot protocol. unit of timeout  ↩
                                      is ms
                    getvar
                    ucmd <any uboot command>
                    acmd <any never returned uboot command, like booti, reboo  ↩
                                                 t>
                    flash [-raw2sparse] <partition> <filename>
                    download -f <filename>

        FBK: community with kernel with fastboot protocol. DO NOT compatible  ↩
                                     with fastboot tools.
                    ucmd <any kernel command> and wait for command finish
                    acmd <any kernel command> don't wait for command finish
                    sync                      wait for acmd 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 ;downl  ↩
                                                                          oad  ↩
                                                 image to path /root/image
                                              copy T:/root/image image ;uploa  ↩
                                                                         d / ↩
                                                 root/image to file image.
                    Example for transfer big file
                            acmd tar -            ; run tar background and ge  ↩
                                                          t data from stdio
                            ucp rootfs.tar.gz T:- ; send to target stdio pipe
                            sync                  ; wait for tar process exit  ↩
                                                            .

For example:
        SDPS: boot -f <filename>
        SDP:  boot -f <filename>
        CFG: SDP: -chip imx6ull -pid 0x1234 -vid 0x5678

SDP: boot -f u-boot-imx7dsabresd_sd.imx -nojump
```

```
 SDP: write -f zImage -addr 0x80800000
 SDP: write -f zImage-imx7d-sdb.dtb -addr 0x83000000
 SDP: write -f fsl-image-mfgtool-initramfs-imx_mfgtools.cpio.gz.u-boot -addr 0x  ↩
                                 83800000
 SDP: jump -f u-boot-dtb.imx -iv
```

# 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
```

**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 2:1 boardA_uuu.auto           monitor port 1:1 and 2:1 for boardsA.
uuu -d -m 1:3: 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}
```

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>
```

## 4 Sample scripts

## 5 Supported protocol

UUU is scripted base multi protocol system.

Built in config:

```
Pctl     Chip            Vid     Pid     BcdVersion
==================================================
SDPS:    MX8QXP          0x1fc9  0x012f  0x0002
SDPS:    MX8QM           0x1fc9  0x0129  0x0002
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
FBK:                     0x066f  0x9afe
FBK:                     0x066f  0x9bff
FB:                      0x0525  0xa4a5
FB:                      0x18d1  0x0d02
```

Table 2: Table UUU Protocol to USB lower level Map

| UUU Protocol | USB Low level transfer |
|---|---|
| SDP | HID |
| SDPU | HID |
| SDPS | HID |
| FB | winusb (windows), raw transfer by libusb |
| FBK | winusb (windows), raw transfer by libusb |

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

### 5.1.1 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]
```

## 5.2 SDPU: uboot implement simplified ROM SDP protocol

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

See below for uboot requirement uboot-config-requirement

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

send image by sdp command.

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

## 5.4 FB: Android fastboot protocol

refer fast boot protocol

See below for uboot requirement

### 5.4.1 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>
```

**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 3: Table 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 |

## 5.5 FBK: Android fastboot protocol, implement at initramfs. See project imx-uuu

### 5.5.1 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 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="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="frf">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 |

# 7 FAQ

- **Win7 can't found driver**

```
Need install winusb driver, you can use https://zadig.akeo.ie/ to install winusb ↩
    driver
```

- **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.
```

- **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
```

- **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:
```

# 8 Build Steps

## 8.1 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

## 8.2 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

# 9 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
```

# 10 kernel config requirement

Some kernel config required

- USB CONFIG FS need be built-in

- One of UDC driver and PHY need be built-in
- Funnction 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)
```

# 11 Release Notes

## 11.1 1.2.TBD (Next)

### 11.1.1 New features

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

### 11.1.2 Bug fixes

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

## 11.2 1.1.81

### 11.2.1 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

### 11.2.2 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

### 11.3 1.1.41

#### 11.3.1 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

#### 11.3.2 Bug fixes

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

## 12 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