UUU (Universal Update Utility)

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME
36bbc5f	Mon Aug 26 11:55:51 2019 -0500	Updated Release Notes (asciidoc)	nxpfrankli
ceabc6d	Tue Sep 3 15:38:31 2019 -0500	Updated Release Notes (asciidoc)	nxpfrankli
77cbd13	Mon Sep 9 14:19:01 2019 -0500	Updated Release Notes (asciidoc)	nxpfrankli
b5b3c21	Fri Sep 20 11:07:03 2019 -0500	Updated Release Notes (asciidoc)	nxpfrankli
791d0b9	Tue Sep 24 10:57:39 2019 -0500	Updated Release Notes (asciidoc)	nxpfrankli
ac59fac	Thu Sep 26 11:09:10 2019 -0500	Updated Release Notes (asciidoc)	nxpfrankli
f5f2193	Thu Oct 10 15:40:48 2019 -0500	Updated Release Notes (asciidoc)	nxpfrankli
cf5dc1d	Mon Oct 14 11:33:56 2019 -0500	Updated Release Notes (asciidoc)	nxpfrankli
3d16892	Tue Oct 22 14:05:39 2019 -0500	Updated Release Notes (asciidoc)	nxpfrankli
f0f6336	Thu Oct 24 13:27:11 2019 -0500	Updated Release Notes (asciidoc)	nxpfrankli
2fee014	Tue Oct 29 10:39:38 2019 -0500	Updated Release Notes (asciidoc)	nxpfrankli
2674a9a	Wed Oct 30 14:51:21 2019 -0500	Updated Release Notes (asciidoc)	nxpfrankli
e32d5a7	Wed Oct 30 14:51:40 2019 -0500	Updated Release Notes (asciidoc)	nxpfrankli
f1fa06f	Thu Oct 31 21:31:04 2019 -0500	Updated FAQ (asciidoc)	nxpfrankli
07fbfdc	Mon Dec 2 10:46:45 2019 -0600	Updated _Sidebar (markdown)	nxpfrankli
daed264	Mon Dec 2 10:47:16 2019 -0600	Updated _Sidebar (markdown)	nxpfrankli
97da919	Mon Dec 2 10:50:08 2019 -0600	Updated _Sidebar (markdown)	nxpfrankli

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME
65f52c2	Wed Feb 12 13:08:50 2020 -0600	Updated Release Notes (asciidoc)	nxpfrankli
d48c02f	Wed Feb 19 10:56:40 2020 -0600	Updated Release Notes (asciidoc)	nxpfrankli
6cf18cf	Wed Mar 25 15:29:13 2020 -0500	add two virual machine screenshot	Frank Li
57dc67e	Wed Mar 25 15:31:29 2020 -0500	Updated How to use UUU on Windows (markdown)	nxpfrankli
79f0584	Wed Mar 25 15:35:18 2020 -0500	Updated FAQ (asciidoc)	nxpfrankli
c56f13f	Wed Mar 25 15:35:49 2020 -0500	Updated FAQ (asciidoc)	nxpfrankli
fab93ef	Wed Mar 25 15:37:31 2020 -0500	Updated FAQ (asciidoc)	nxpfrankli
be40e7f	Mon Apr 6 12:53:47 2020 -0500	Updated Release Notes (asciidoc)	nxpfrankli

Contents

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.

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

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 bootimage into QSPI flash

```
uuu -b qspi qspi_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/*
```

Notes: bootloader means bootable image, which included ROM required header. for imx6/7, it should be uboot.imx generally. for imx8qxp\imx8qm\imx8mm\im8mq, it is flash.bin.

Burn release image into emmc

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

Note: some release combine multi board into one zip package, you need use uuu release.zip/uuu.auto-<boardname>

More usage please refer Example

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

License

uuu is licensed under the BSD license. See LICENSE. The BSD licensed prebuilt Windows binary version of uuu is statically linked with the LGPL libusb library, which remains LGPL.

- bzip2 (BSD license) is from https://github.com/enthought/bzip2-1.0.6
- zlib (zlib license) is from https://github.com/madler/zlib.git
- libusb (LGPL-2.1) is from https://github.com/libusb/libusb.git

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.

Setup auto parameter complete

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 }
```

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
```

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 [?]

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
```

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.

Usage Example

Basic

Download boot loader for imx6\imx7

```
uuu uboot.imx
```

Download boot loader for imx8qxp

```
uuu flash.bin
```

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
```

Burn Android Image to eMMC

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

Burn yocto Image to eMMC

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

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
```

multi boards support

For the same boards

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

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.

Talk with fastboot

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
```

write image to emmc

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

Sample scripts

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]
        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..0xffff]
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 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

SDP: i.MX6/7 ROM download protocol

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]
```

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
```

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

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

send image by sdp command.

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

FB: Android fastboot protocol

refer fast boot protocol

See below for uboot requirement

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

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

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
```

Common command for all protocol

Table 4: 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

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 SDPU: write -f flash.bin -offset 0x57c00 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 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

Win 7 User Guide

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

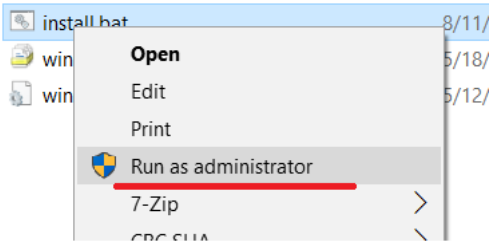
Back Ground

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")

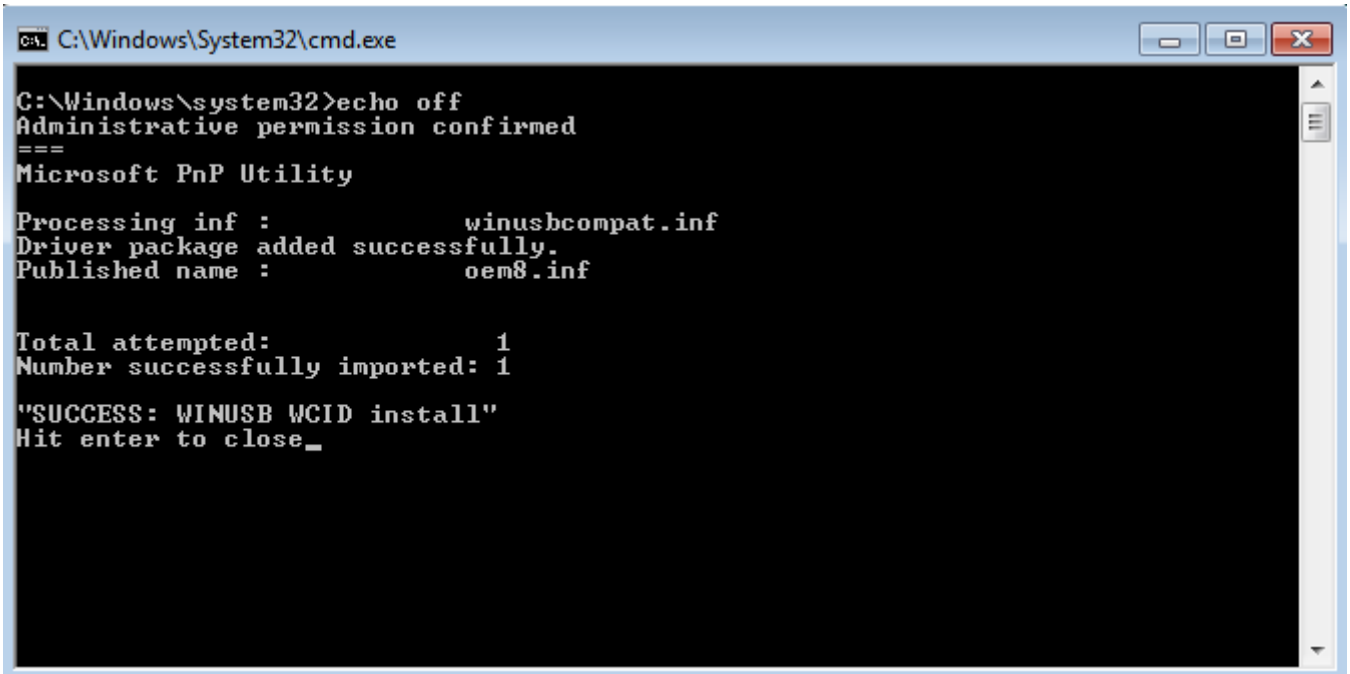
Some windows update also included updated .inf file. You can try run uuu to see what happen. If windows report "can't install driver", that means your system missed such update file.

Install updated winusb inf file

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



- The below screen show install success



Notes:

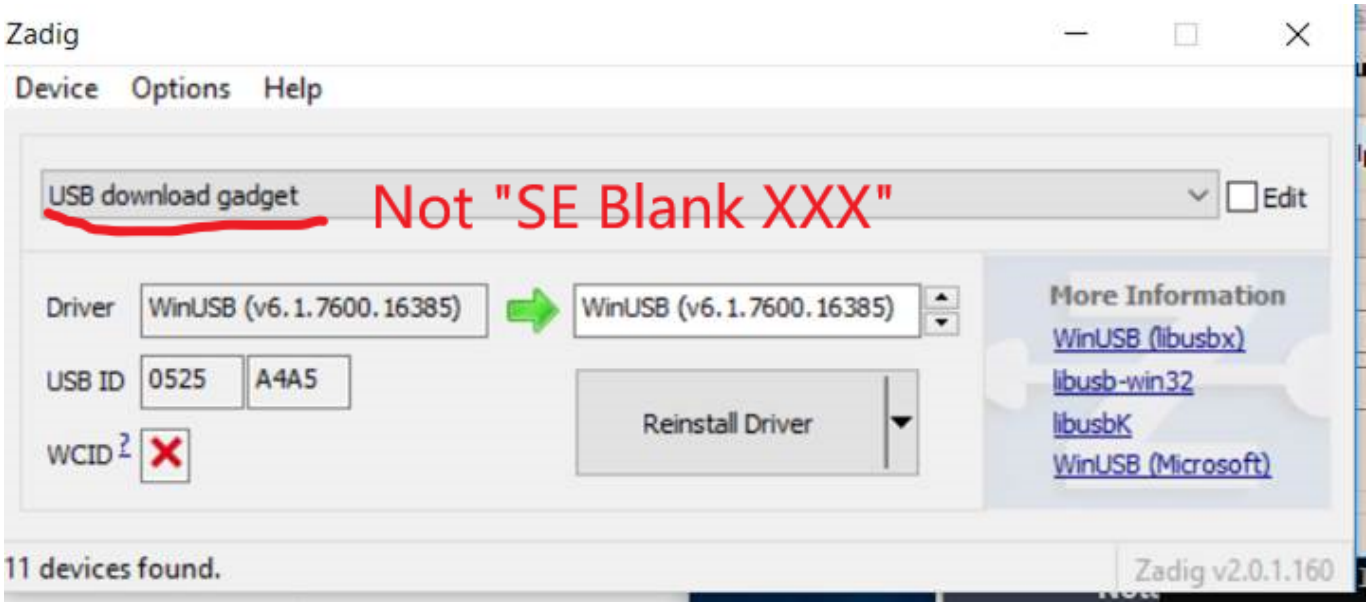
4.9.123 8MM GA and 4.14 beta release missed a patch. Please apply below patch in uboot

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

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.
```

- **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

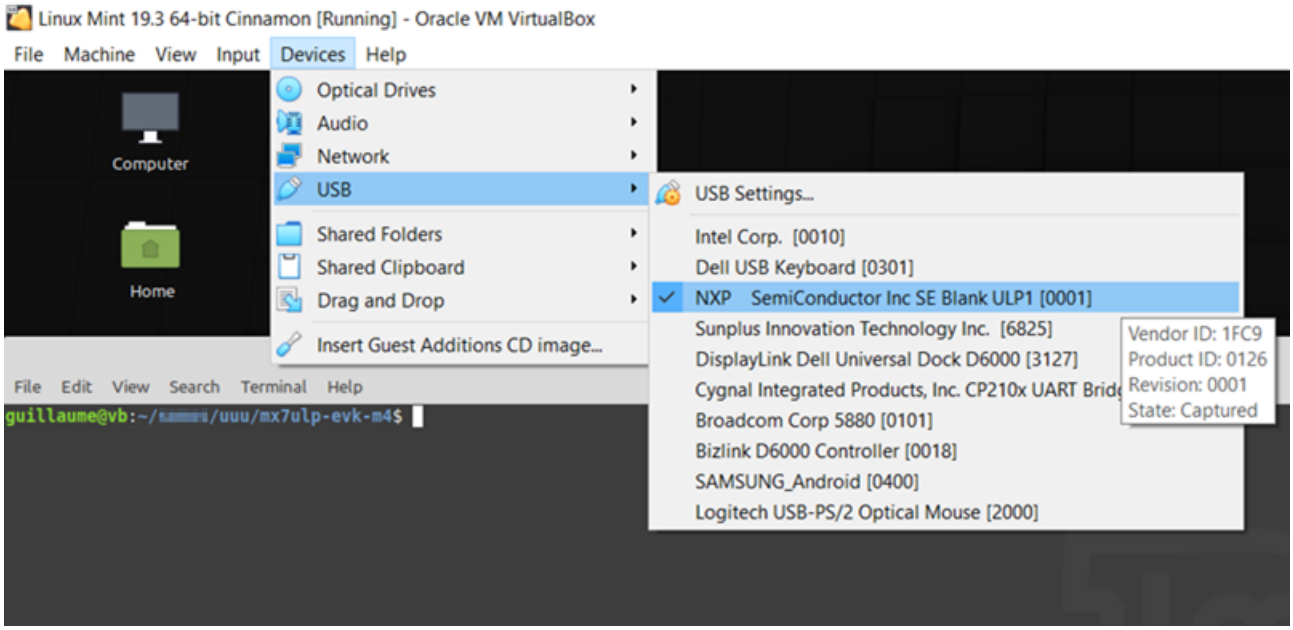
```
diff --git a/drivers/usb/gadget/f_fastboot.c b/drivers/usb/gadget/f_fastboot.c
index ae8fe80..cd46ca4 100644
--- a/drivers/usb/gadget/f_fastboot.c
+++ b/drivers/usb/gadget/f_fastboot.c
@@ -2543,10 +2543,10 @@ static int fastboot_bind(struct usb_configuration *c,  ↩
    struct usb_function *f)
         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**

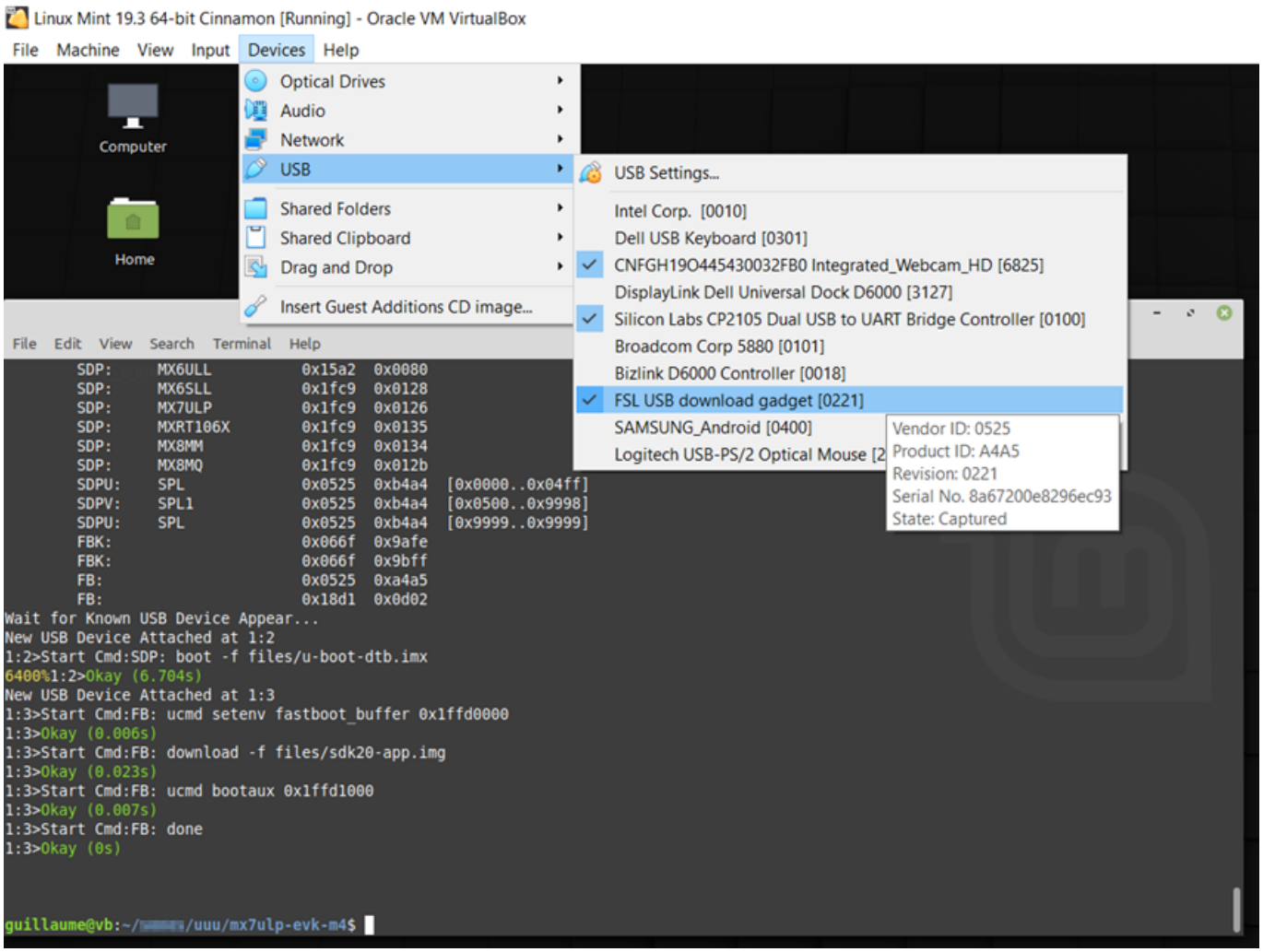
If use bz2 or zip file, it is possible out of memory. Please increase swap partition size in linux. increase virtual memory size in windows.

- **How to use uuu in virtual machine**

1. Map HID device into virtual machine



1. Map fastboot device into virtual machine after uboot boot



1. You can use uuu burn image in virtual machine

Build Steps

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

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

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 5: 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

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)
```

Release Notes

1.3.154

new features

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

Bug fixes

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

1.3.134

Bug fixes

- fix qm\qxp download failure if use spl image

1.3.130

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

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

1.3.102

New features

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

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)

1.3.82

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

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

1.2.135

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

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

1.2.91

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.

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

1.2.0

New features

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

Bug fixes

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

1.1.81

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

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

1.1.41

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

Bug fixes

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

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.