Manufacturing Tool V2 (MFGTool2) Quick Start Guide

Contents

Overview

This document describes a simple step-by-step guide on how to use Manufacturing (MFG) tool V2 and some Q/A which may be referred to frequently.

OK, let’s begin.

- **Step 1**

Before running the application, there are two files which must exist in the same folder with the application: cfg.ini and UICfg.ini. Additionally, the corresponding configuration must be set correctly. An incorrect configuration will cause the application to work abnormally.

- The UICfg.ini file is used to configure the number of ports which indicates how many boards can be supported simultaneously.

  The format of the UICfg.ini file is as indicated below:

  ```
  [UICfg]
  PortMgrDlg=1
  ```

  For example, if only one board at a time will be supported, "PortMgrDlg=1" should be set. Currently, Up to four boards can be supported, so PortMgrDlg can be set to 1-4.

- The cfg.ini file is used to configure the target chip profile and target operation list.

  The format of this file looks like like the following:

  ```
  [profiles]
  chip = MX6DL Linux Update

  [platform]
  board = ARM2

  [LIST]
  name = Android-SD
  ```

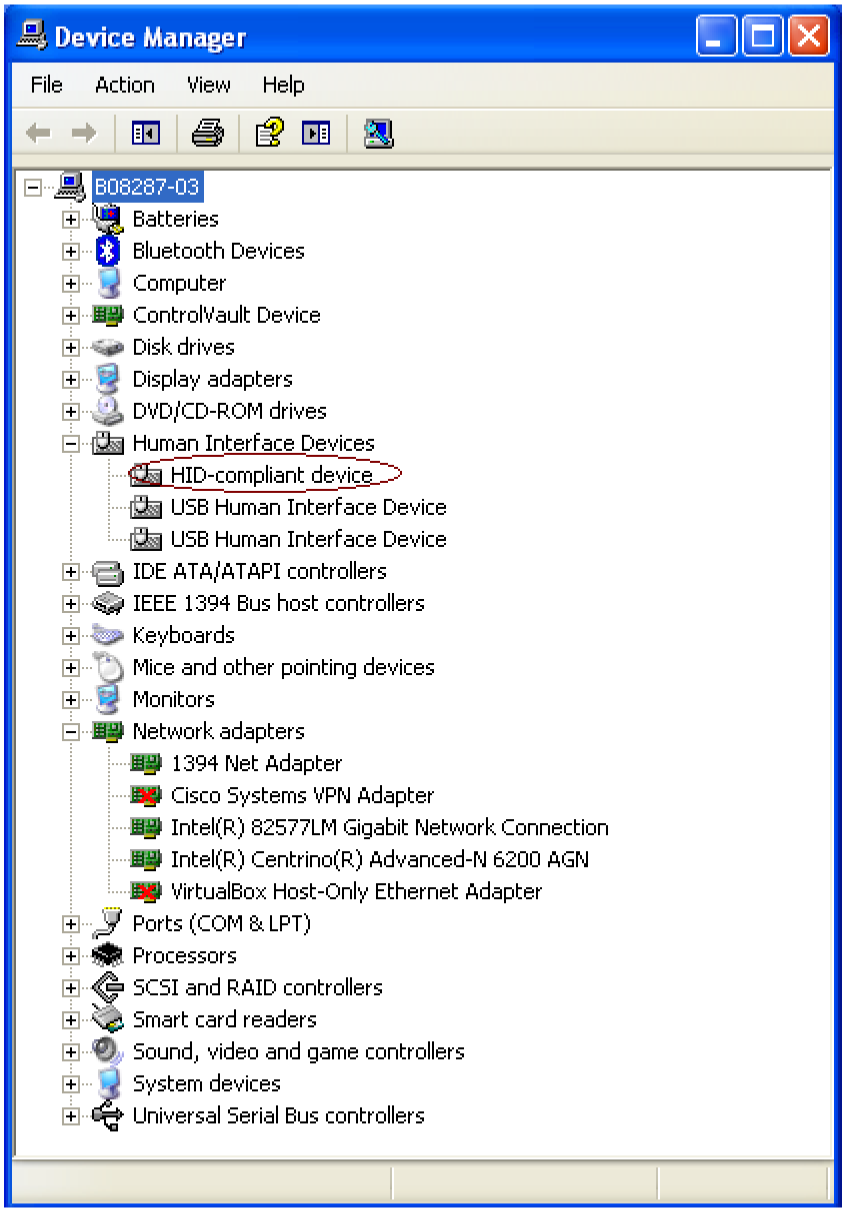
  “profiles/chip” indicates the target profile name, and “list/name” indicates the target operation list name which can be found in the file located at "profiles/CHIP_PROFILE/OS Firmware/ucl2.xml". Currently, the “platform/board” is reserved and not used, please ignore it.

- **Step 2**

Set the correct boot mode and connect the OTG port to the PC on which the MFG Tool application will be run.

To set the correct boot mode, refer to the Linux BSP User Guide document.

After connecting to PC with the correct boot mode setting, a HID-compliant device will be shown in the Device Manager as shown below:



```
B08287-03
    Batteries
    Bluetooth Devices
    Computer
    ControlVault Device
    Disk drives
    Display adapters
    DVD/CD-ROM drives
    Human Interface Devices
        HID-compliant device
        USB Human Interface Device
        USB Human Interface Device
    IDE ATA/ATAPI controllers
    IEEE 1394 Bus host controllers
    Keyboards
    Mice and other pointing devices
    Monitors
    Network adapters
        1394 Net Adapter
        Cisco Systems VPN Adapter
        Intel(R) 82577LM Gigabit Network Connection
        Intel(R) Centrino(R) Advanced-N 6200 AGN
        VirtualBox Host-Only Ethernet Adapter
    Ports (COM & LPT)
    Processors
    SCSI and RAID controllers
    Smart card readers
    Sound, video and game controllers
    System devices
    Universal Serial Bus controllers
```

- **Step 3**

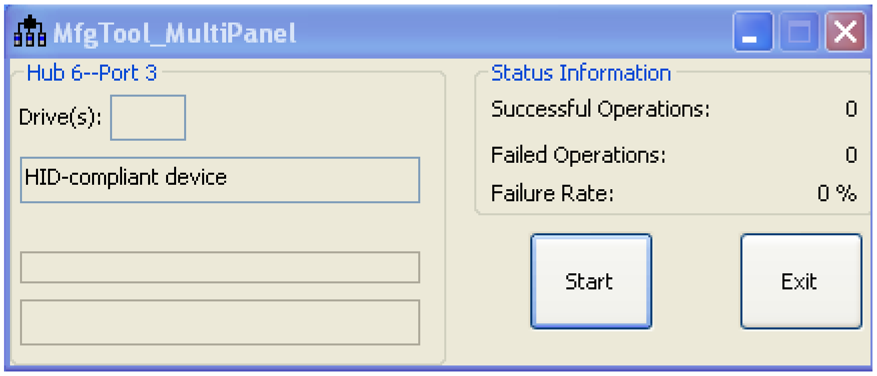
Double click the application to run.

There is a limitation that, when using the MFGTool V2 for the first time to burn an image to a device (such as: i.MX 6Quad ARM2, i.MX 6DualLite Sabre-SD, and so on), the device must be connected to PC before MFGTool V2 starts running.

Only two buttons can be clicked, Start/Stop, and Exit.

Start/Stop is used to start/stop the burning process. If you re-start the burning process after you stop it, the process will try to continue from the point where you stopped before, but it is not guaranteed that it can continue successfully. It is NOT recommended to do this.

Exit is used to exit this application. Please note that you can exit the application only after you stop the burning process.

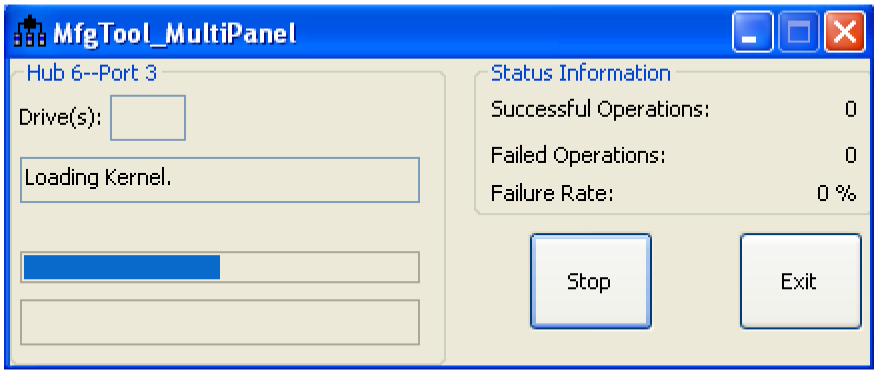


HID-compliant device

Successful Operations: 0
Failed Operations: 0
Failure Rate: 0 %

- **Step 4**

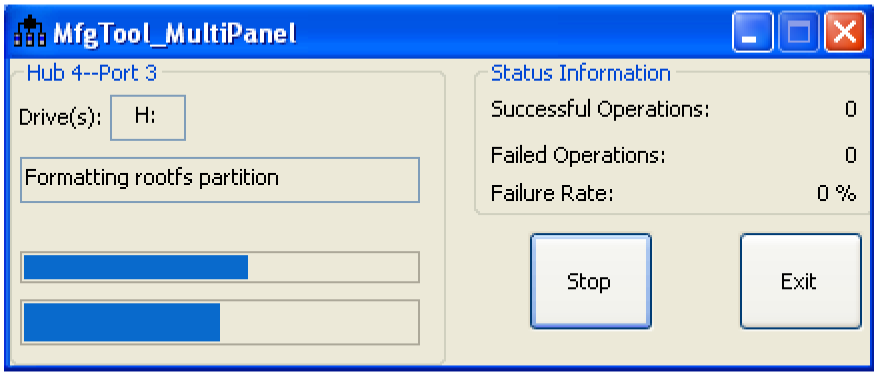
Click “Start” button. If you have a terminal tool to monitor the debug serial port of your board, it is suggested to open it. You can get more information from it.



Loading Kernel.

Successful Operations: 0
Failed Operations: 0
Failure Rate: 0 %

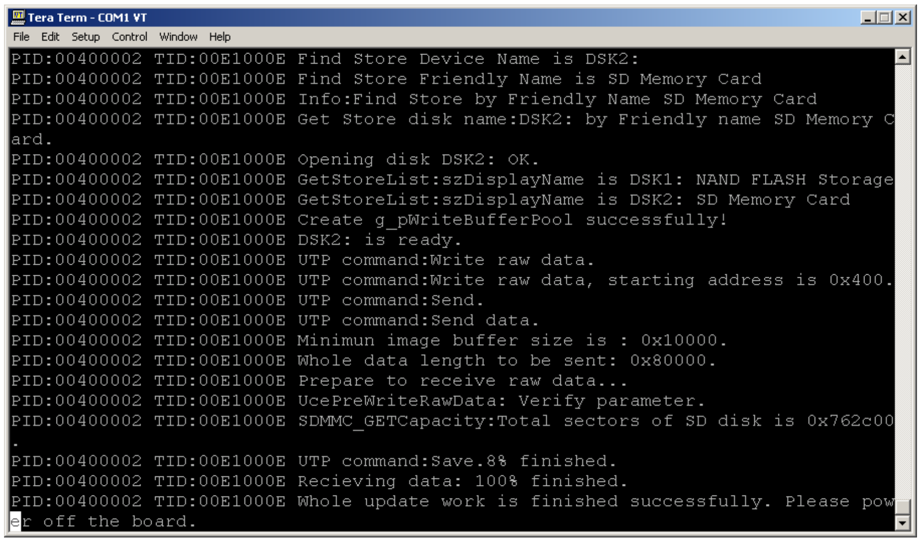
Process continuing:



Formatting rootfs partition

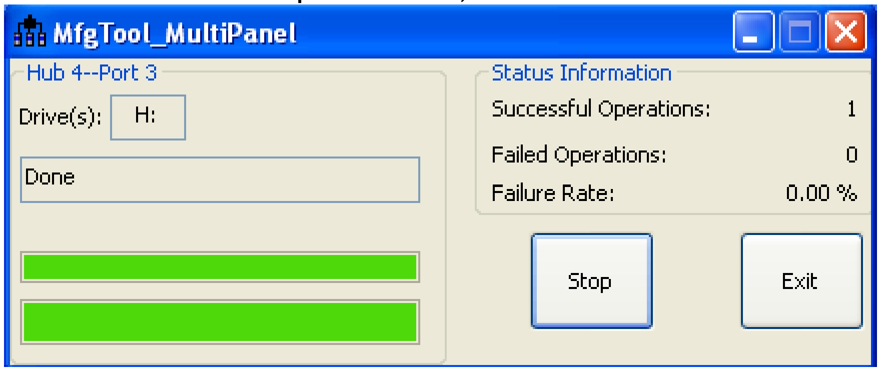
Successful Operations: 0
Failed Operations: 0
Failure Rate: 0 %

You can find information from the terminal.



```
PID:00400002 TID:00E1000E Find Store Device Name is DSK2:
PID:00400002 TID:00E1000E Find Store Friendly Name is SD Memory Card
PID:00400002 TID:00E1000E Info:Find Store by Friendly Name SD Memory Card
PID:00400002 TID:00E1000E Get Store disk name:DSK2: by Friendly name SD Memory C
ard.
PID:00400002 TID:00E1000E Opening disk DSK2: OK.
PID:00400002 TID:00E1000E GetStoreList:szDisplayName is DSK1: NAND FLASH Storage
PID:00400002 TID:00E1000E GetStoreList:szDisplayName is DSK2: SD Memory Card
PID:00400002 TID:00E1000E Create g_pWriteBufferPool successfully!
PID:00400002 TID:00E1000E DSK2: is ready.
PID:00400002 TID:00E1000E UTP command:Write raw data.
PID:00400002 TID:00E1000E UTP command:Write raw data, starting address is 0x400.
PID:00400002 TID:00E1000E UTP command:Send.
PID:00400002 TID:00E1000E UTP command:Send data.
PID:00400002 TID:00E1000E Minimun image buffer size is : 0x10000.
PID:00400002 TID:00E1000E Whole data length to be sent: 0x80000.
PID:00400002 TID:00E1000E Prepare to receive raw data...
PID:00400002 TID:00E1000E UcePreWriteRawData: Verify parameter.
PID:00400002 TID:00E1000E SDMMC_GETCapacity:Total sectors of SD disk is 0x762c00
.
PID:00400002 TID:00E1000E UTP command:Save.8% finished.
PID:00400002 TID:00E1000E Recieving data: 100% finished.
PID:00400002 TID:00E1000E Whole update work is finished successfully. Please pow
er off the board.
```

It is done. Click “Stop” to finish, and Click “Exit” to terminate the application.



Done

Successful Operations: 1
Failed Operations: 0
Failure Rate: 0.00 %

- **Note:** The manufacturing tool may sometimes report an error message when it is downloading the file system to the SD card. This can be caused by insufficient space in the SD card due to a small partition size. To fix this, unzip the file

"Profiles\ CHIP_PROFILE \OS Firmware\mksdcard.sh.tar" and then modify the script to increase the size of the partition and create more partitions according to your file system requirements. After the modification is done, tar the script again.

For MFGTool V1.x User

MFG tool V2.0 uses ucl2.xml which is different than V1 in which ucl.xml is used. All description about ucl.xml in V1 documents can be applied to ucl2.xml, with small changes as shown below:

- The <STATE>...</STATE> item is simplified,
  Old format in ucl.xml
  <STATE name="Recovery" dev="MX6Q"/>
  <DEV name="MX6DL/Solo" vid="15A2" pid="0054"/>

  New format in ucl2.xml
  <STATE name="BootStrap" dev="MX6Q" vid="15A2" pid="0054"/>

  For state "BootStrap", the valid strings for dev are: "MX6Q", "MX6D", "MX6SL". Others are invalid.
  For state "Updater", the valid string for dev is: "MSC". This name is constant for all the SoC.
  The valid strings for state name are: "BootStrap", "Updater". Others are invalid.

- Add a "state" attribute into each <CMD.../>. This attribute specifies the state that this command will run in, for example:
  Old format in ucl.xml:

  <CMD type="boot" body="Recovery" file ="u-boot-mx6sl.bin" >Loading U-boot</CMD>

  New format in ucl2.xml:

  <CMD state="BootStrap" type="boot" body="Recovery" file ="u-boot-mx6sl.bin" >Loading U-boot</CMD>

  It indicates that this command will be running at BootStrap state.

- "state" concept
  In general, the whole burning process includes two phases:
  I. BootStrap
     In this phase, the MFG tool will burn the specific U-Boot image and kernel image to the target board, so that MFG tool firmware can run successfully on the target board.
  II. Updater
     Host side application builds connection with firmware and transfers the U-Boot image and kernel image which will be used

normally with the firmware. Eventually, firmware burns these images or roofs to the storage to finish the whole process.

- The “find” command is not used anymore. It needs to be removed from the ucl script.
- The last command, the command text must be "Done" (case-insensitive) For example:
  <CMD state="Updater" type="push" body="$ echo Update Complete!">Done</CMD>

How to Choose the Dedicated Storage Device on the Dedicated Board

One operation list is dedicated for one kind of storage on the dedicated board. You can get this information from the operation list name in ucl2.xml.

For example, “<LIST name="Sabre-SD" desc="Choose SD as media">” indicates this operation will burn image to SD on Freescale Sabre-SD reference board.

“<LIST name="ubuntu-SabreSD-eMMC" desc="Choose eMMC as media">” indicates this operation will burn image to eMMC on Freescale Sabre-SD reference board.

“<LIST name="ARM2-SD" desc="Choose SD as media">” indicates this operation will burn image to SD on Freescale ARM2 reference board.

How to Program a Fuse

MfgTool V2 supports writing the specified value into the fuse.
The OTP fuse can be written through the following commands:

<CMD state="Updater" type="push" body="$ ls /sys/fsl_otp ">Showing HW_OCOTP fuse bank</CMD>

<CMD state="Updater" type="push" body="$ echo 0x11223344 > /sys/fsl_otp/HW_OCOTP_MAC0">write 0x11223344 to HW_OCOTP_MAC0 fuse bank</CMD>

<CMD state="Updater" type="push" body="$ cat /sys/fsl_otp/HW_OCOTP_MAC0">Read value from HW_OCOTP_MAC0 fuse bank</CMD>

The fuse bank name (ex: HW_OCOTP_MAC0) should be changed according as needed.

How to Burn an Image

In the file ucl2.xml located at Profiles\ CHIP_PROFILE\OS Firmware, you can find instructions like this:

<CMD state="Updater" type="push" body="send" file="files/u-boot-mx6q-arm2.bin">Sending u-boot.bin</CMD>

<CMD state="Updater" type="push" body="$ dd if=$FILE of=/dev/mmcblk0 bs=1k seek=1 skip=1 conv=fsync">write u-boot.bin to sd card</CMD>

<CMD state="Updater" type="push" body="send" file="files/uImage">Sending kernel uImage</CMD>

<CMD state="Updater" type="push" body="$ dd if=$FILE of=/dev/mmcblk0 bs=1M seek=1 conv=fsync">write kernel image to sd card</CMD>

The value of key word "file" here means the relative path (based on ucl2.xml path) of the image to be burned.

Key word "send" indicates this file will be sent to target.

After file is received by target device, we can use "dd" command to burn the file to the related storage. "dd" is a standard Linux command. For detailed information refer to the i.MX 6Dual/6Quad Linux Reference Manual.

How to Burn Your Own Image with Manufacturing Tool

The processes above are limited to the reference design boards provided by Freescale. All the U-Boot images and kernel images are used to support Freescale reference design boards. If you want to utilize the tool to burn your own image on your own board, all you need to do is the following:

- Generate a special U-Boot image and kernel image for MFG tool
- Generate a normal U-Boot image and kernel image, maybe customizing rootfs which can be used by the end user.
- Refer to the ucl2.xml to create your own operation list. Usually, only changing the original image (U-Boot and kernel) is enough.

The detailed information about how to generate Manufacturing firmware, refer to the Document "Manufacturing Tool v2 Linux or Android Firmware Development Guide."

Command line feature

MfgTool2 can support command line feature, the commands that can be accepted are "-c, -l, -p and -noui".

The format of command line looks like:

MfgTool2.exe [-noui] [-c] ["*chip profile folder name*"] [-l] ["*list name*"] [-p] [*number*] [-s] ["variable=value"]

Parameters description:

- -noui: this command has no any parameter, if this command is used, the application will use the console interface, otherwise, the GUI interface will be used.

Note: this command must be the second parameter (the first parameter is application name), if it is used.

- -c: indicate the target profile name. The parameter of this command is a string with a pair of double quotes.
- -l: indicate the target operation list name which can be found in the file located at "profiles/CHIP_PROFILE/OS Firmware/ucl2.xml". The parameter of this command is a string with a pair of double quotes.
- -p: indicate the number of ports which indicates how many boards can be supported simultaneously. The parameter of this command must be a number between 1 and 4.
- -s: Set ucl variable value. Support multiply if you need set more one variable.

- All above parameters are not mandatory, if no parameter in the command line, the application will try to find the parameter from the corresponding file, e.g. if '-c' parameter is not provided in the command line, the application will try to get it from 'cfg.ini' file just like the v2.0.x, if the application can't get the parameter from both the command line and cfg.ini file, the application will fail to run.

- If both command line and cfg.ini/UIcfg.ini assign the same parameter, the application will take the command line parameter with priority.

- Press CTRL+C or the Close button to close the APP.

Some examples on how to use command line feature:

1. MfgTool2.exe –noui –c "MX6Q Linux Update" –l "Sabre-SD" –p 4
   The application will use the console interface to burn image to four boards simultaneously. The target profile is 'MX6Q Linux Updater' and the operation list is 'Sabre-SD' which is located at "profiles/ MX6Q Linux Updater/OS Firmware/ucl2.xml".

2. MfgTool2.exe –c "MX6Q Linux Update" –l "Android-SabreSD-eMMC" –p 2
   The application will use the GUI interface to burn image to two boards simultaneously. The target profile is 'MX6Q Linux Updater' and the operation list is 'Android-SabreSD-eMMC' which is located at "profiles/ MX6Q Linux Updater/OS Firmware/ucl2.xml".

3. MfgTool2.exe -noui –l "Android-SabreSD-eMMC" –p 1
   The application will use the console interface to burn image to one board. The target profile is gotten from "profiles/chip" in file cfg.ini,

the operation list is ‘Android-SabreSD-eMMC’ which is located at “profiles/ CHIP_PROFILE /OS Firmware/ucl2.xml”.

4. MfgTool2.exe –noui –c “MX6Q Linux Update” –l “Sabre-SD”
   The application will use the console interface to burn image to one board. The target profile is ‘MX6Q Linux Updater’ and the operation list is ‘Sabre-SD’ which is located at “profiles/ MX6Q Linux Updater/OS Firmware/ucl2.xml”. The maximum number of boards supported simultaneously is gotten from “UICfg/PortMgrDlg” in file UICfg.ini.

5. mfgtool2.exe -c "linux" -l "SDCard" -s "board=sabresd" -s "mmc=0" -s "sxuboot=sabresd" -s "sxdtb=sdb"
   The application will use gui interface to burn image to one board. The target profile is ‘linux’ and operation list is ‘SDCard’ which is located at profiles/Linux/OS Firmware/ucl2.xml. Set value ‘sabresd’ to ucl variable ‘board’. Set value ‘0’ to ucl variable ‘mmc’. Set value ‘sxuboot’ to ucl variable ‘sabresd’. Set value ‘sxdtb’ to ucl variable ‘sdb’.

6. MfgTool2.exe
   It is just same with v2.0.x’s behavior.

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support