Manufacturing Tools for i.MX Applications Processors

AMF-ACC-T1652

Bryan Thomas

SEPT. 2015

freescale™

External Use

MFG Tools

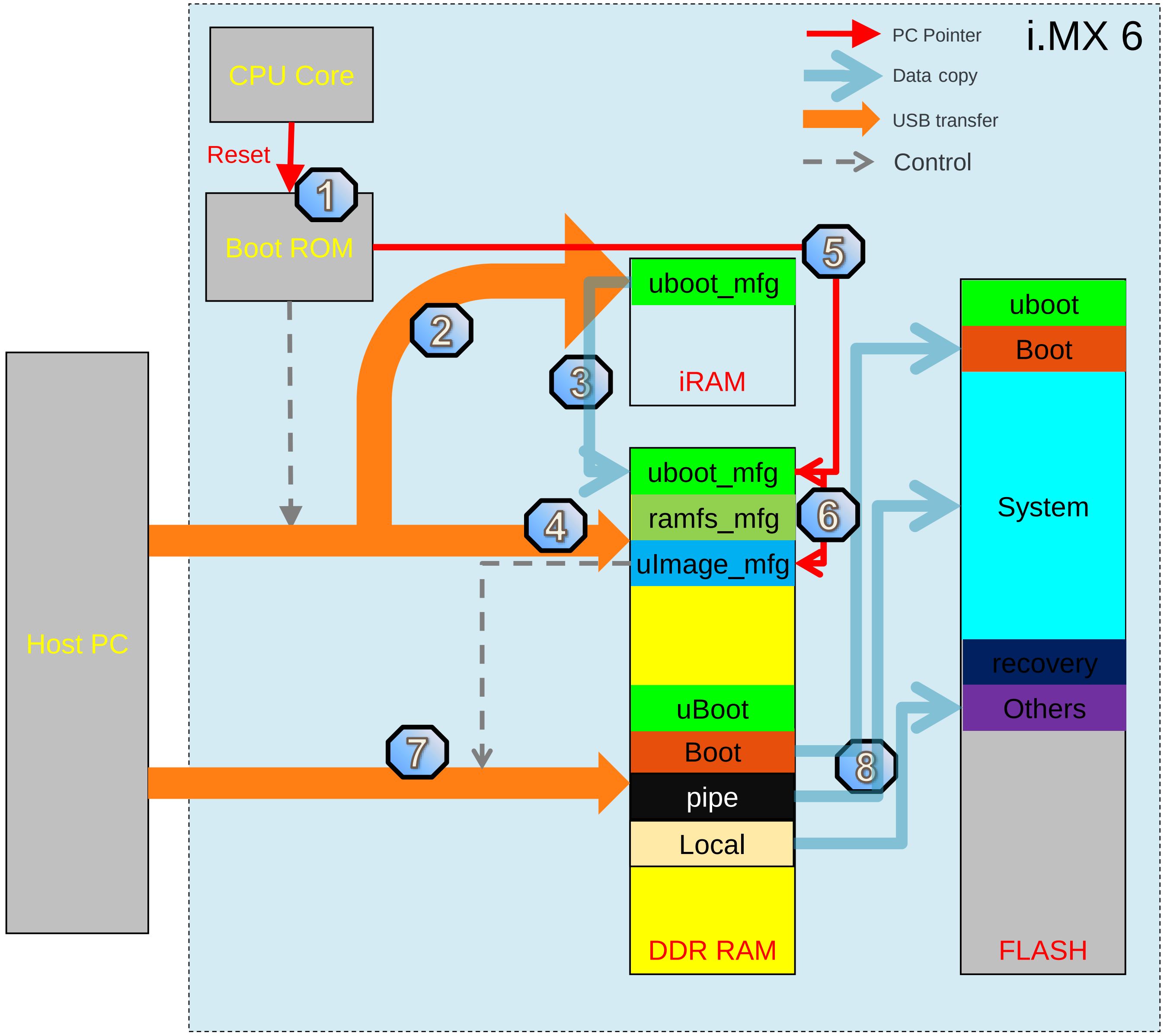
What is in the Mfgtool

- What is the tool used for
  - To burn your own firmware, demo files and other images to storage media (NOR, Nand, SD, etc.).
  - Debug tool in early stage of a project
    - download & execute uboot from RAM…
  - Recover a “brain dead” board
  - Program OTP bits/fuses
  - Etc.?

Work flow

1. Core reset. PC Jump to Boot ROM if configured as download mode or empty boot device.
2. ROM enumerate USB as HID with host, download uboot_mfg from host and store it into iRAM.
3. ROM parse IVT/DCD in uboot_mfg and init the DDR. Copy the body of uboot_mfg to DDR.
4. Boot ROM continue to download uImage_mfg and ramfs_mfg from PC to DDR.
5. PC send "jump" command to ROM. ROM then hand over execution to uboot_mfg in DDR.
6. Uboot_mfg continue to boot into uImage_mfg with ramfs_mfg configured.
7. uImage_mfg re-enumerate USB as UMS devices and download all images from host to DDR .
8. Write images from DDR to flash (dd/pipe) or do some local action with flash such as format.

(*red text is flexible user steps)



Building for mfg. tool – from "*Freescale_Yocto_Project_User's_Guide.pdf*"

- **6.2 Manufacturing Tool, MFGTool**
  - The recipes used to build a manufacturing tool image
    - linux-imx-mfgtool
    - u-boot-mfgtool
  - To build a manufacturing image do the following -
    - $ bitbake fsl-image-mfgtool-initramfs
  - A manufacturing tool kernel is built using the imx_v7_mfg_defconfig while the default kernel is built by using the imx_v7_defconfig. This is handled automatically by the MFGTool recipes listed above.
  - Creates (in tmp/deploy/images/imx6qsabreauto):
    - u-boot-imx6qsabreauto-mfgtool-2014.04-r0.imx
    - zImage_mfgtool
    - zImage-mfgtool--3.14.28-r0-imx6q-sabreauto-20150915175120.dtb (plus a few other dtb files)
    - fsl-image-mfgtool-initramfs-imx6qsabreauto-20150915175120.rootfs.cpio.gz.u-boot

**Freescale Yocto Project User's Guide, Rev. L3.14.28_1.0.0-ga, 04/2015

Main files used with mfg. tool.

- **MFGTool2.exe**
  - Main program.Uses all other files and provides GUI.
- **cfg.ini**
  - let MFGTool2 know where it can get the ucl2.xml
  - Defines which operation list should be used for next programming session
- **UICfg.ini**
  - PortMgrDlg defines the max number of devices you want to operate at the same time
- **ucl2.xml**
  - tell MFGTool2 what kind of SoC it works on
    - different SoCs have different PID/VID, and MFGTool2 needs this information
  - Define operation lists
    - Each board may have different uboot.bin and kernel image
    - One operation list is dedicated to defining a specific storage on a specific board.
- **MfgTool.log**
  - Contains results from last mfg. tool session.

UICfg.ini

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

- PortMgrDlg=1
  - Tells us that one device will be programmed for the next session

Configuration

- **cfg.ini**

  The cfg.ini file is used to configure the target chip profile and target operation list. The format of this file looks like as the following:

[profiles] *chip = Linux*	Indicates the target profile name (i.e. directory) which can be found under "<MFG>/Profiles" MFGTool2 will try to find the ucl2.xml with the path (relative path to mfgtool2.exe) Profiles**${chip}**\OS Firmware. If you take the above chip value as an example, MFGTool2 will try to find the ucl2.xml at "Profiles**Linux**\OS Firmware".
[platform] *board = SabreSD*	Reserved

[LIST]
name = SDCard
	Indicates the target operation list name which can be found in the file located at "<MFG>/Profiles/${chip}/OS Firmware/ucl2.xml". The name specified here points to a list entry in XML.
	For example in "Profiles\Linux\OS Firmware" there should be this based on the above cfg.ini example contents:
	`<LIST name="`**SDCard**`" desc="Choose SD Card as media">`
	…
	`</LIST>`

[variable] *board = sabreauto* *mmc = 0* *sxuboot=17x17arm2* *sxdtb=17x17-arm2* *ldo=*	Variable values used in the ucl2.xml file in command lists... `<CMD state="BootStrap" type="boot" body="BootStrap" file ="firmware/u-boot-imx6q`**%board%**`_sd.imx" ifdev="MX6Q">Loading U-boot</CMD>`

Commands

- **ucl2.xml**

  A collection of all the tasks needed to do burning work. Consists of several parts:

  ➢ Global Configuration is contained between <CFG></CFG>.

```
<UCL>
    <CFG>
      <STATE name="BootStrap" dev="MX6SL" vid="15A2" pid="0063"/>
      <STATE name="BootStrap" dev="MX6D" vid="15A2" pid="0061"/>
      <STATE name="BootStrap" dev="MX6Q" vid="15A2" pid="0054"/>
      <STATE name="BootStrap" dev="MX6SX" vid="15A2" pid="0071"/>
      <STATE name="Updater"   dev="MSC" vid="066F" pid="37FF"/>
    </CFG>
```

"<STATE name="BootStrap" dev="MX6Q" vid="15A2" pid="0054"/>" *- indicates the first phase of the burning process, the phase name is "BootStrap", and a device named "MX6Q" could be connected with the USB pid "0054" and vid "15A2", or MX6SL, MX6D or MX6SX. For i.MX 6 serial, in the phase "BootStrap", the valid strings for dev are: "MX6Q", "MX6D", "MX6SL" and "MX6SX"*

"<STATE name="Updater"   dev="MSC" vid="066F" pid="37FF"/>" *indicates the second phase of the burning process, the phase name is "Updater", and a device named "MSC" should be connected with the USB pid "37FF" and vid "066F".*

- State name indicates the stage the command to be executed: "BootStrap" (communicates with Boot ROM) or "Updater" (communicates with mfg kernel - uuc).

Commands

- **ucl2.xml continued**

  Command LIST is contained between <LIST></LIST>.

List/command name

BootStrap commands – communicate with imx6 BootROM

```
<UCL>
 <LIST name="SDCard" desc="Choose SD as media">
…
        <CMD state="BootStrap" type="boot" body="BootStrap" file ="firmware/u-boot-imx6q%board%_sd.imx" ifdev="MX6Q">Loading U-boot</CMD>
        <CMD state="BootStrap" type="load" file="firmware/zImage" address="0x12000000“
                loadSection="OTH" setSection="OTH" HasFlashHeader="FALSE" ifdev="MX6Q MX6D">Loading Kernel.</CMD>
        <CMD state="BootStrap" type="load" file="firmware/fsl-image-mfgtool-initramfs-imx6qdlsolo.cpio.gz.u-boot" address="0x12C00000“
                loadSection="OTH" setSection="OTH" HasFlashHeader="FALSE" ifdev="MX6Q MX6D">Loading Initramfs.</CMD>
        <CMD state="BootStrap" type="load" file="firmware/zImage-imx6q-%board%%ldo%.dtb" address="0x18000000“
                loadSection="OTH" setSection="OTH" HasFlashHeader="FALSE" ifdev="MX6Q">Loading device tree.</CMD>
        <CMD state="BootStrap" type="jump" > Jumping to OS image. </CMD>

        <CMD state="Updater" type="push" body="send" file="files/u-boot-imx6q%board%_sd.imx" ifdev="MX6Q">Sending u-
                boot.bin</CMD>
        <CMD state="Updater" type="push" body="$ dd if=/dev/zero of=/dev/mmcblk%mmc% bs=1k seek=384 conv=fsync
                count=129">clear u-boot arg</CMD>
        <CMD state="Updater" type="push" body="$ dd if=$FILE of=/dev/mmcblk%mmc% bs=1k seek=1 conv=fsync">write u-boot.bin to
                sd card</CMD>
        <CMD state="Updater" type="push" body="$ mkfs.vfat /dev/mmcblk%mmc%p1">Formatting rootfs partition</CMD>
        <CMD state="Updater" type="push" body="$ mkdir -p /mnt/mmcblk%mmc%p1"/>
…
</LIST>
```

Updater commands – communicate with linux mfg kernel

Commands

- **ucl2.xml continued**

  BootStrap commands

```
<CMD state="BootStrap" type="boot" body="BootStrap" file ="firmware/u-boot-imx6q%board%_sd.imx" ifdev="MX6Q">Loading U-boot</CMD>
```

1. state = "BootStrap" – command communicates with the bootROM

2. type = "boot" - download u-boot-mx6q-sabresd.bin from host and store it into iRAM. ROM parse IVT/DCD in u-boot-mx6q-sabresd.bin and init the DDR. Copy the body of u-boot-mx6q-sabresd.bin to DDR.
3. body = "BootStrap" –
4. file ="firmware/u-boot-imx6q%board%_sd.imx" – file used for this command – located in Profiles/Linux/OS Firmware/, with the %board% coming from cfg.ini, [variable] section.
5. "Loading U-boot" – comment that appears in the mfgtool GUI when this command is executed.

Commands

- **ucl2.xml continued**

  BootStrap commands continued

```
<CMD state="BootStrap" type="load" file="firmware/zImage" address="0x12000000“ loadSection="OTH" setSection="OTH" HasFlashHeader="FALSE"
ifdev="MX6Q MX6D">Loading Kernel.</CMD>
```

1. state = “BootStrap” – command communicates with the bootROM

2. type = “load” – load image to memory
3. file ="firmware/zImage“ – file load use for this command – located in Profiles/Linux/OS Firmware/.
4. loadSection=“OTH” – a parameter used by ROM code, should be set to “OTH”.
5. HasFlashHeader="FALSE“ – set TRUE if the image contains a flash header, or set to FALSE.
6. ifdev="MX6Q MX6D“ – conditional describing which device this statement should be used with (see “dev=“ in <CFG></CFG> in ucl2.xml file – this parameter will be detected by the tool)
7. “Loading Kernel” – comment that appears in the mfgtool GUI when this command is executed.

Commands

- **ucl2.xml continued**

  BootStrap commands “jump”

```
<CMD state="BootStrap" type="jump" > Jumping to OS image. </CMD>
```

1. state = “BootStrap” – command communicates with the bootROM
2. type = “jump” – Notify ROM code to jump to the RAM image to run.
3. “Jumping to OS image” – comment that appears in the mfgtool GUI when this command is executed.

Commands

- **ucl2.xml continued**

  Updater commands – now that we’ve jumped to the mfg uboot/kernel/ramfs, we’re ready to tell it what to do…

```
<CMD state="Updater" type="push" body="send" file="files/u-boot-imx6q%board%_sd.imx" ifdev="MX6Q">Sending u-boot.bin</CMD>
```

1. state = “Updater” – command communicates with the mfg. kernel
2. type = “push” – the command is parsed and executed by the targeted device instead of host, the only thing host has to do is to send the command to the targeted device..
3. body=“send” - Receive the file from the host. Subsequent shell commands can refer to the file received as $FILE.
4. file="files/u-boot-imx6q%board%_sd.imx“ – file to send to the target (imx6), the %board% coming from cfg.ini, [variable] section.
5. ifdev="MX6Q MX6D“ – conditional describing which device this statement should be used with (see “dev=“ in <CFG></CFG> in ucl2.xml file – this parameter will be detected by the tool)
6. “Sending u-boot.bin” – comment that appears in the mfgtool GUI when this command is executed.

Commands

- **ucl2.xml continued**

  Updater commands – now that we've jumped to the mfg uboot/kernel/ramfs, we're ready to tell it what to do…

```
<CMD state="Updater" type="push" body="$ dd if=/dev/zero of=/dev/mmcblk%mmc% bs=1k seek=384 conv=fsync count=129">clear u-boot arg</CMD>
```

1. state = "Updater" – command communicates with the mfg. kernel
2. type = "push" – the command is parsed and executed by the targeted device instead of host, the only thing host has to do is to send the command to the targeted device..
3. body="$ dd if=/dev/zero of=/dev/mmcblk%mmc% bs=1k seek=384 conv=fsync count=129" – The "$" means command to run on target device. the %mmc% coming from cfg.ini, [variable] section.
4. "clear u-boot arg" – comment that appears in the mfgtool GUI when this command is executed.

Commands

- **ucl2.xml continued**

  Updater commands – now that we've jumped to the mfg uboot/kernel/ramfs, we're ready to tell it what to do…

```
<CMD state="Updater" type="push" body="frf">Finishing rootfs write</CMD>
```

1. state = "Updater" – command communicates with the mfg. kernel
2. type = "push" – the command is parsed and executed by the targeted device instead of host, the only thing host has to do is to send the command to the targeted device..
3. body = "frf" – Wait for all data transfer to be finished and processed (i.e. same as flush).
4. "Finishing rootfs write" – comment that appears in the mfgtool GUI when this command is executed.

Commands

- **ucl2.xml continued**

From “*Manufacturing Tool V2 UCL User Guide.docx*”

1.1.2.1 OTP Bits Programming

```
<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 set as needed.

*See “*Manufacturing Tool V2 UCL User Guide.docx*” for more details and examples.

Logfile

- **MfgTool.log**
  - Located in same directory as MfgTool2.exe.
  - Log lines:
    - DLL version: 2.3.4
    - Tuesday, September 15, 2015 17:51:14 Start new logging
    - ModuleID[2] LevelID[10]: DeviceManager::OnMsgDeviceEvent() - DEVICE_ARRIVAL_EVT(\\?\USB#VID_15A2&PID_0054#5&1604d86d&0&1#{a5dcbf10-6530-11d2-901f-00c04fb951ed})
      - VID=15a2, PID=0054 – it's imx6q!
    - ModuleID[2] LevelID[10]: ExecuteCommand--Boot[WndIndex:0], File is E:\mfg_tools\imx-3.14.28_1.0.0_ga-mfg-tools\mfgtools\Profiles\Linux\OS Firmware\firmware\u-boot-imx6qsabreauto_sd.imx
      - Loading the mfg. uboot file
    - ModuleID[2] LevelID[10]: *********MxHidDevice[00E0AC98] Jump to Ramkernel successfully!**********
      - Jumped to the mfg. builds of uboot,kernel,ramfs.
  - Can find error messages when board isn't programming.

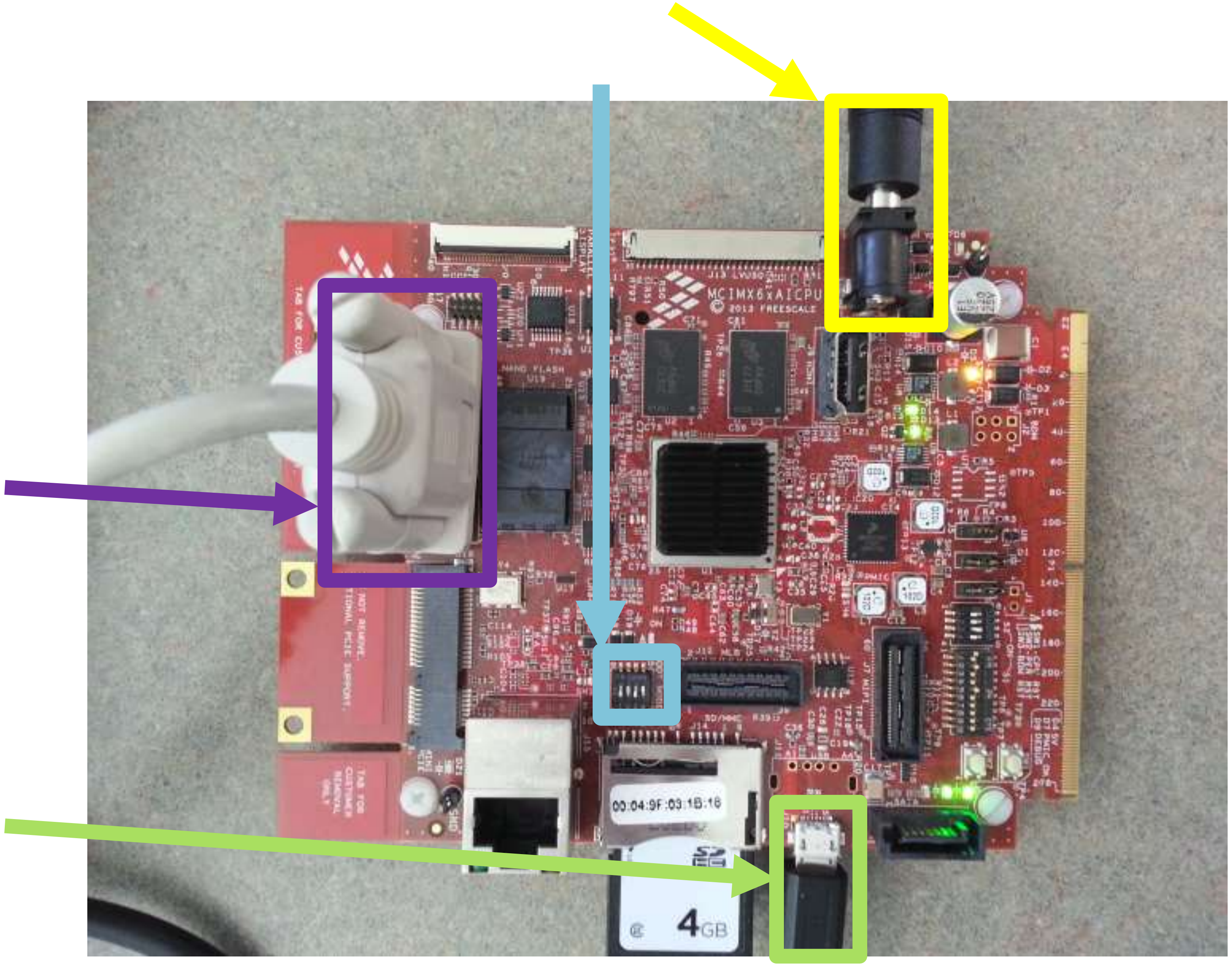
Download Preparation

Setup download environment as following:

1. A target device. Set boot option to “download” mode with DIP switches.
2. Prepare a micro-USB cable.
3. Prepare a PC with proper MFG tool installed (download from www.freescale.com/imx).
4. Prepare images to be downloaded to the target device.

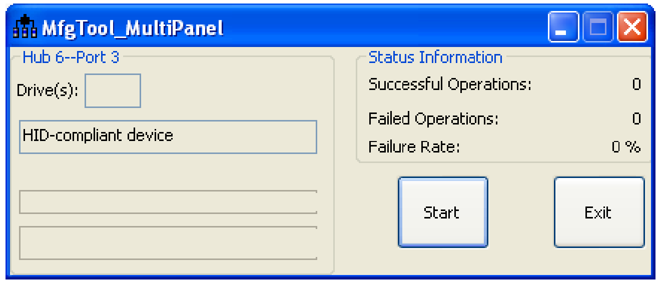
Set HW and connections.

- Set DIP switches on board
- Connect micro USB cable
- Connect other end of USB to PC
- **Connect serial cable (optional – will show linux booting, etc. mainly for debugging purpose if the board fails)**
- Plug power to board



Operation of Mfgtool

- Open “MfgTool.exe”. A device named “HID-compliant device” will be found if there is a valid device plugged in.
- Click “Start” button, it starts burning work.
- Finished successfully.
- Press Stop button before exit.


MfgTool_MultiPanel

Hub 6--Port 3

Drive(s):

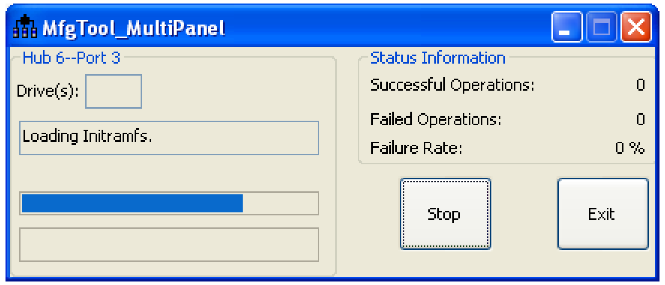
HID-compliant device

Status Information

Successful Operations: 0

Failed Operations: 0

Failure Rate: 0 %


MfgTool_MultiPanel

Hub 6--Port 3

Drive(s):

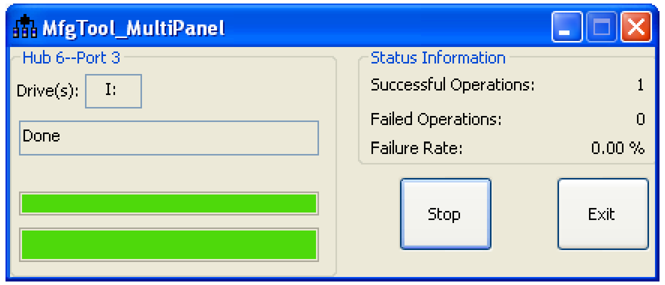
Loading Initramfs.

Status Information

Successful Operations: 0

Failed Operations: 0

Failure Rate: 0 %


MfgTool_MultiPanel

Hub 6--Port 3

Drive(s): I:

Done

Status Information

Successful Operations: 1

Failed Operations: 0

Failure Rate: 0.00 %

www.Freescale.com