How to populate boards for Android

How to populate boards for Android

Stable: 26.03.2020 - 14:07 / Revision: 26.03.2020 - 13:37

This article describes how to load Android™ distribution images, built for the STM32MPU, to defined Flash device partitions. It is intended for Distribution Package users.

Contents

1 Prerequisites

The recommended host PC setup is specified in the following article: PC prerequisites.

The environment must be installed using the Distribution Package adapted to the selected microprocessor device. See the list of Android™ Distribution Package.

It is assumed that:

- STM32CubeProgrammer tool has been installed (refer to STM32CubeProgrammer)
- STM32MPU distribution for Android™ has been built (refer to How to build STM32MPU distribution for Android)

Additionally, **Fastboot** mode is useful to accelerate the device provisioning (commands sent through USB).

In Linux environments, the Android USB drivers are built-in. The only action required is to set the target device information.

To peform this action, you need administrator rights. In addition, you may need to add *sudo* in front of each executed command

To do this, open a terminal and

- Create (or update if it already exists) the *51-android.rules* file in */etc/udev/rules.d/* with the following information
  - *idVendor* = 0483 (STMicroelectronics vendor)
  - *idProduct* = 0afb (Fastboot on STMicroelectronics device)
  - *Mode* = 0660 (read/write permissions)
  - *Group* = plugdev (Unix group which owns the device node)

Check if you belong to the *plugdev* group by executing the following command:

```
$ groups
```

Example:

```
# Fastboot on STMicroelectronics devices
SUBSYSTEM=="usb", ATTR{idVendor}=="0483", ATTR{idProduct}=="0afb", MODE="0660", GROUP="
plugdev"
```

- Make sure that the access rights to the created file are the correct ones:

```
chmod a+r /etc/udev/rules.d/51-android.rules
```

At this stage, your USB driver is correctly installed and configured.

2 Populate a board

The `out/target/product/<BoardId>` directory contains all images built for the selected board during the setup (`lunch` step).

Build directory structure: images

This is an example: the files seen in a customized build directory might differ slightly from this list

`out/target/product/<BoardId>`:

```
├── boot.img                  Binary for boot partition(s)
├── dt.img                    Linux® kernel device tree image for dt partition(s)
├── fsbl-optee.img            TF-A binary for fsbl partition(s) (case optee boot mode)
├── fsbl-trusted.img          TF-A binary for fsbl partition(s) (case trusted boot
mode)
├── fsbl-programmer.img       TF-A binary used directly by the STM32CubeProgrammer
├── misc.img                  Binary for misc partition
├── splash.img                Binary for splash partition (splashscreen)
├── ssbl-optee-fbsd.img       U-Boot binary for ssbl partition (case optee boot mode)
├── ssbl-trusted-fbsd.img     U-Boot binary for ssbl partition (case trusted boot mode)
├── ssbl-programmer.img       U-Boot binary used directly by the STM32CubeProgrammer
├── system.img                Binary for system partition(s)
├── teed.img                  Binary for teed partition (OP-TEE OS paged data)
├── teeh.img                  Binary for teeh partition (OP-TEE OS header image)
├── teex.img                  Binary for teex partition (OP-TEE OS resident core)
├── userdata.img              Binary for userdata partition
├── vendor.img                Binary for vendor partition(s)
└── [...]
```

The *STM32CubeProgrammer* tool is used to flash the first stages of the images to the board. Then *fastboot* is used to flash the remaining images.

The board shall be started in "Forced USB boot for flashing" mode (also named DFU), selecting it through boot switches as explained in the document *<your board reference - hardware description>*, chapter *Boot related switches* (for example, boot switches on the STM32MP157x-EV1 board).

Connect the board to the host PC through USB.

The board can be designed to support several flash devices, for example: microSD, eMMC and so on. This document uses the microSD card as the Flash device example for this article.

- Go to the distribution root directory, and run flash-device:

```
PC $> flash-device
```

- Select the flashlayout to be used (ex: *FlashLayout_sd_optee.tsv* to flash OP-TEE build for microSD card)

```
1) FlashLayout_emmc_optee.tsv         3) FlashLayout_sd_optee.tsv
2) FlashLayout_sd_trusted.tsv         4) FlashLayout_emmc_trusted.tsv
Which layout do you want to flash ?
```

Once the DFU flashing is completed, the boot switches must be configured so that the correct Flash device (e.g. microSD card) is selected as the boot source.

- •Keep the reset button pressed until the board enters fastboot (refer to LEDs and buttons on STM32 MPU boards). The device provisioning continues until the required images have been loaded.

This operation takes several minutes.

- When the download is finished, press the reset button in the board. The Android software then starts.

3 Populate a microSD card

Images can be flashed directly to a microSD card connected to the board.

Before starting, the microSD card settings need to match the configuration of the board. Refer to the Memory settings page.

3.1 Format the microSD card

The first time the microSD card is used or the partition layout changes, it needs to be formated. The goal is to prepare it with appropriate partition size to receive the images.

```
PC $> format-device <device_path>
```

With:

- *<device_path>*: the device identified in the system to access the microSD card (`/dev/sdX` for microSD card connected through USB dongle or `/dev/mmcblkX` for microSD card connected through reader, `X` is the instance associated to your Flash device).

This command will automatically use the partition configuration from `device/stm/<STM32Series>/layout/android_layout.config`.

For more information see the usage option:

```
Usage: format-device [Options] <device_path>

  This script allows the formatting of the memory device to create required partition
before provisioning.

Options:
  -h/--help: print this message
  -v/--version: get script version
  -s/--size <disk-size>: set requested disk size [4GiB or 8GiB] (default:
PART_MEMORY_SIZE value set in <path_to>/android_layout.config)
  -c/--config <config-file-path>: set used partition configuration file (default:
<path_to>/android_layout.config)

<device_path>: /dev/sdX (sd connected through usb), /dev/mmcblkX (sd connected through
reader)
```

The microSD card needs to be disconnect and reconnect the before going further

3.2 Provision the microSD card

To push the built images to the microSD card, the following instruction has to be executed:

```
PC $> provision-device <device_path>
```

With:

- *<device_path>*: the device identified in the system to access the microSD card (`/dev/sdX` for microSD card connected through USB dongle or `/dev/mmcblkX` for microSD card connected through reader, `X` is the instance associated to the Flash device).

Then all images are flashed on the Flash device.

The microSD card can be plugged to the board and rebooted.

4 Flash a dedicated image

If an Android distribution is already installed and booted on the board, it's possible to update partitions one by one depending on the need. For example, if the kernel is rebuilt, just flash the boot (kernel), the dt (device tree) and the vendor images (modules).

First the device has to be restarted in fastboot mode:

- Keep the reset button pressed until the board enters fastboot (USER PA13 for Evaluation board, USER2 for Discovery kits).

or

- Run the command

```
PC $> adb reboot-bootloader
```

Finally, execute the flashing command:

```
PC $> provision-device -i reboot
```

- **-i** option means that a confirmation will be needed for flashing each partition at a time (select the partitions that need to be updated)
- **reboot** option means that the device will reboot automatically at the end

5 Update an Android distribution

If the content of the partitions dedicated to Android™ needs to be updated (`vendor`,`system` or `userdata`) after a rebuild, use ADB tool.

The read-only partitions need to be changed to read-write:

```
PC $> adb root
PC $> adb remount
```

It is then possible to update the Android™ partitions.

```
PC $> adb sync
```

Finally, the device needs to be rebooted to take these changes into account

```
PC $> adb reboot
```

eval,disco (Generic term used, to complete configuration modules paths depending on used board)

Trusted Firmware for Arm Cortex-A

Das U-Boot -- the Universal Boot Loader (see U-Boot_overview)

Open Portable Trusted Execution Environment

Operating System

Device Firmware Upgrade

former spelling for e•MMC ('e' in italic)

stm32mp1