A Document on

1. Device Tree & Atags

2. Board Booting Sequence

3. SD card Partition Procedure

- By Rathod Raja V.

Device Tree & Atags

- ➔ **The kernel contains the entire description of the hardware.**
- ➔ **The bootloader loads a single binary, the kernel image, and executes it. uImage or zImage**

ATAGS interface:

Minimal information is passed from firmware
to the kernel with a tagged list of predefined parameters.

r0 : 0
r1 : Machine type number
r2 : Physical address of tagged list in system RAM

- ➔ **The bootloader prepares some additional information, called ATAGS, which address is passed to the kernel through register r2 Contains information such as memory size and location, kernel command line, etc.**

- ➔ **The bootloader tells the kernel on which board it is being booted through a machine type integer, passed in register r1.**

Device Tree File [.dts]:

- ➔ **The kernel no longer contains the description of the hardware, it is located in a separate binary: the device tree blob [.dtb]**
- ➔ **The bootloader loads two binaries: the kernel image and the DTB.**
- ➔ **Kernel image remains uImage or zImage**
- ➔ **Device Tree (DT), is a data structure and language for describing hardware. More specifically, it is a description of hardware that is readable by an operating system so that the operating system doesn't need to hard code details of the machine.**
- ➔ **DTB located in arch/arm/boot/dts, one per board**
- ➔ **The bootloader passes the DTB address through r2. It is supposed to adjust the DTB with memory information, kernel command line, and potentially other info.**

Board Booting Sequence

1] Power on

2] ROM Code [Similler to BIOS Code]

Location: ROM

Execution: ROM

Functionality:

a) Basic Hardware initialization

b) Load second stage bootloader

3] X-Loader [MLO] (Also called secondary program loader)

Location: SD card/ emmc [SD card First Partition]

Execution: SRAM [Internal]

Functionality:

a) Initialization of DRAM (Externel memory)

b) Load u-boot bootloader (u-boot.img)

4] U-boot loader [Universal boot loader]

Location: SD card/ emmc [SD card First Partition]

Execution: DRAM

Functionality:

a) Board Support Package Initialization [BSP] (Mux configuartion)

b) Load kernel Image (Uimage)

5] Kernel (Linux Kernel Image) [Uimage]

Location: SD card/ emmc [SD card Second Partition]

Execution: DRAM

Functionality:

a) Initialization of Device driver,wi-fi, bluetooth,MMU , interrupt, kernel service etc.

b) Mount RFS (Root File System)

U-Image contain the board specific information

6) Root File System [RFS]

SD card Partition Procedure

[Using cmd]

1] Begin partitioning the microSD Card by typing (sudo fdisk /dev/sdX)

2] Initialize a new partition table by selecting (o), then verify the partition table is empty by selecting (p).

3] Create a boot partition by selecting (n) for 'new', then (p) for 'primary', and (1) to specify the first partition. Press (enter) to accept the default first sector and specify (4095) for the last sector.

4] Change the partition type to FAT16 by selecting (t) for 'type' and (e) for 'W95 FAT16 (LBA)'.

5] Set the partition bootable by selecting (a) then (1).

6] Next, create the data partition for the root filesystem by selecting (n) for 'new', then (p) for 'primary', and (2) to specify the second partition. Accept the default values for the first and last sectors by pressing (enter twice).

7] Press (p) to 'print' the partition table. It should look similar to the one below.

```
Disk /dev/sdb: 7948 MB, 7948206080 bytes
255 heads, 63 sectors/track, 966 cylinders, total 15523840 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0xafb3f87b

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1   *        2048        4095        1024    e  W95 FAT16 (LBA)
/dev/sdb2            4096    15523839     7759872   83  Linux
```

8.Finally, commit the changes by selecting (w) to ‘write’ the partition table and exit fdisk.

9.Format the Partitions

Format partition 1 as FAT by typing (sudo mkfs.vfat /dev/sdX1)

Format partition 2 as ext4 by typing (sudo mkfs.ext4 /dev/sdX2)