≝ **Main Site** › Articles

📓 Knowledge article

Linux Boot Process for Sitara TI Processors

Created 1 year, 4 months ago   Active 7 months ago   Last edited 1 year, 4 months ago   Viewed 99 times

1 min read

ti   linux   booting-sitara-processors
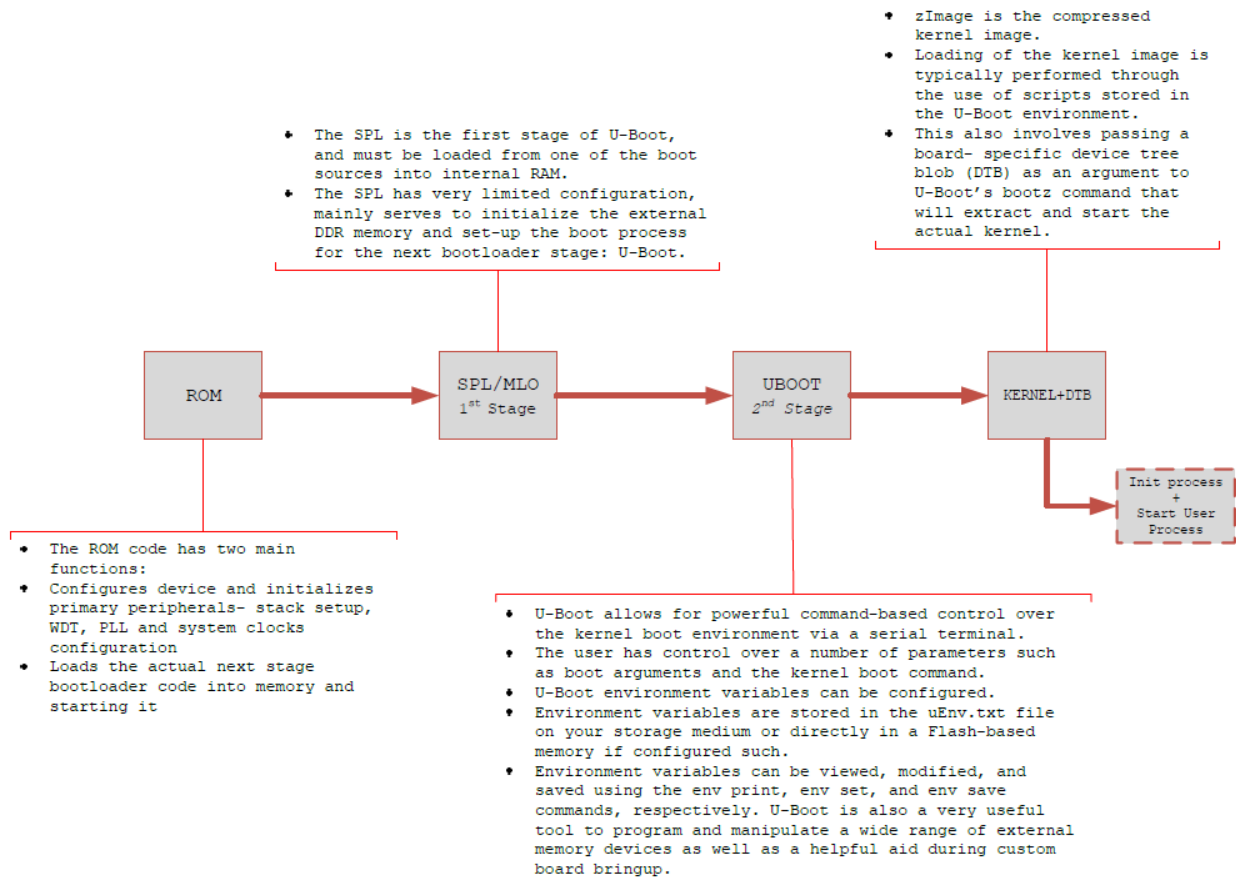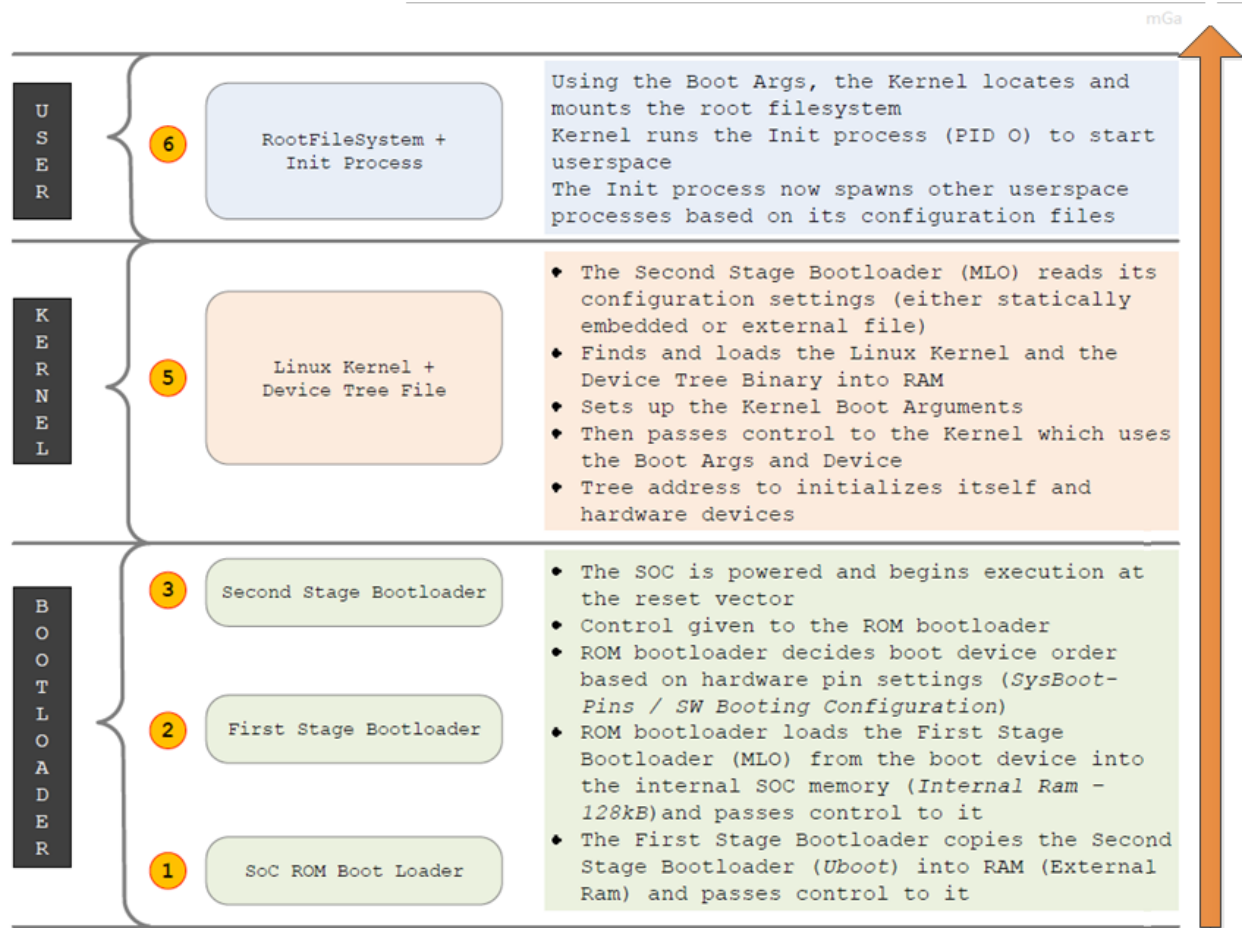
Share article   ···   Edit article

14
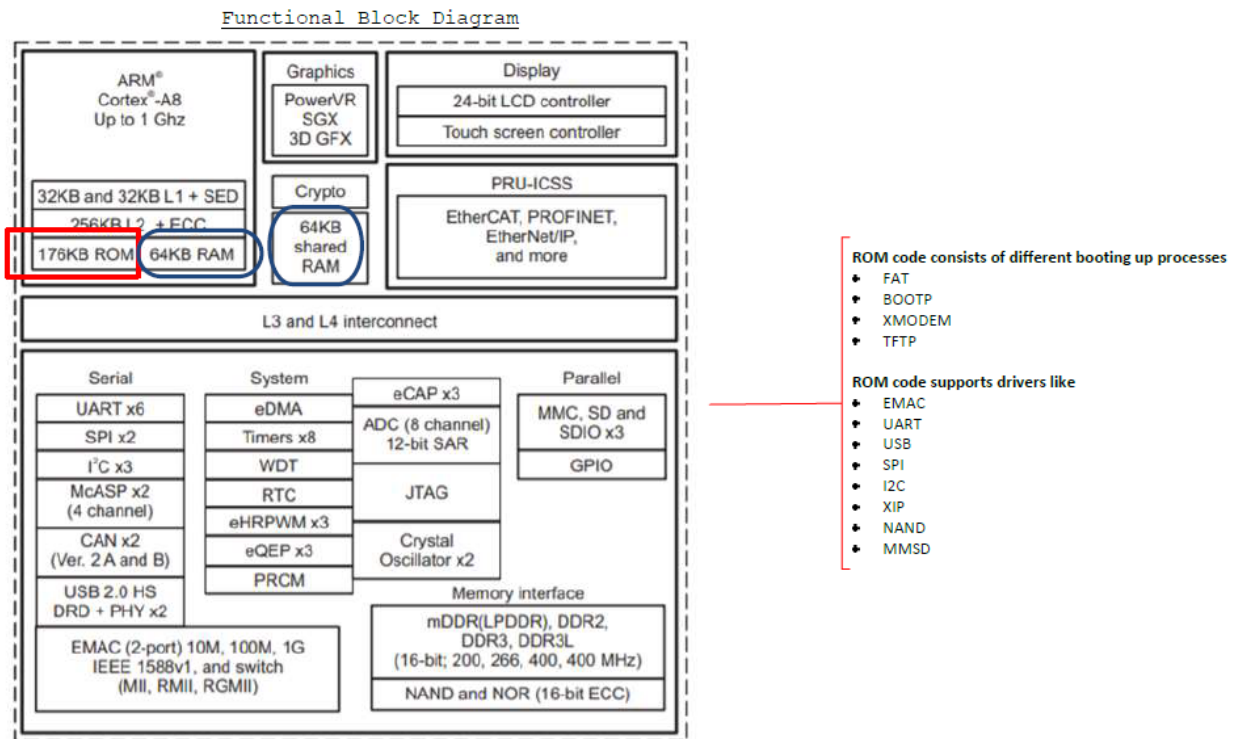
Understanding Linux booting process of TI-SITARA Processors 🔗

Overview of Linux booting 🔗

mGaik

- zImage is the compressed kernel image.
- Loading of the kernel image is typically performed through the use of scripts stored in the U-Boot environment.
- This also involves passing a board-specific device tree blob (DTB) as an argument to U-Boot's bootz command that will extract and start the actual kernel.

- The SPL is the first stage of U-Boot, and must be loaded from one of the boot sources into internal RAM.
- The SPL has very limited configuration, mainly serves to initialize the external DDR memory and set-up the boot process for the next bootloader stage: U-Boot.

```
  ROM  ──▶  SPL/MLO   ──▶  UBOOT    ──▶  KERNEL+DTB
            1st Stage       2nd Stage
                                          │
                                          ▼
                                      ┌─────────────┐
                                      │Init process │
                                      │     +       │
                                      │ Start User  │
                                      │  Process    │
                                      └─────────────┘
```

- The ROM code has two main functions:
- Configures device and initializes primary peripherals- stack setup, WDT, PLL and system clocks configuration
- Loads the actual next stage bootloader code into memory and starting it

- U-Boot allows for powerful command-based control over the kernel boot environment via a serial terminal.
- The user has control over a number of parameters such as boot arguments and the kernel boot command.
- U-Boot environment variables can be configured.
- Environment variables are stored in the uEnv.txt file on your storage medium or directly in a Flash-based memory if configured such.
- Environment variables can be viewed, modified, and saved using the env print, env set, and env save commands, respectively. U-Boot is also a very useful tool to program and manipulate a wide range of external memory devices as well as a helpful aid during custom board bringup.

From two stage booting to loading linux kernel 🔗

U S E R	6	RootFileSystem + Init Process	Using the Boot Args, the Kernel locates and mounts the root filesystem Kernel runs the Init process (PID 0) to start userspace The Init process now spawns other userspace processes based on its configuration files
K E R N E L	5	Linux Kernel + Device Tree File	• The Second Stage Bootloader (MLO) reads its configuration settings (either statically embedded or external file) • Finds and loads the Linux Kernel and the Device Tree Binary into RAM • Sets up the Kernel Boot Arguments • Then passes control to the Kernel which uses the Boot Args and Device • Tree address to initializes itself and hardware devices
B O O T L O A D E R	3 2 1	Second Stage Bootloader First Stage Bootloader SoC ROM Boot Loader	• The SOC is powered and begins execution at the reset vector • Control given to the ROM bootloader • ROM bootloader decides boot device order based on hardware pin settings (*SysBoot-Pins / SW Booting Configuration*) • ROM bootloader loads the First Stage Bootloader (MLO) from the boot device into the internal SOC memory (*Internal Ram – 128kB*) and passes control to it • The First Stage Bootloader copies the Second Stage Bootloader (*Uboot*) into RAM (*External Ram*) and passes control to it

Understanding first stage bootloader - ROMBOOT - ROMCODE 🔗

From AM335x TRM

Functional Block Diagram

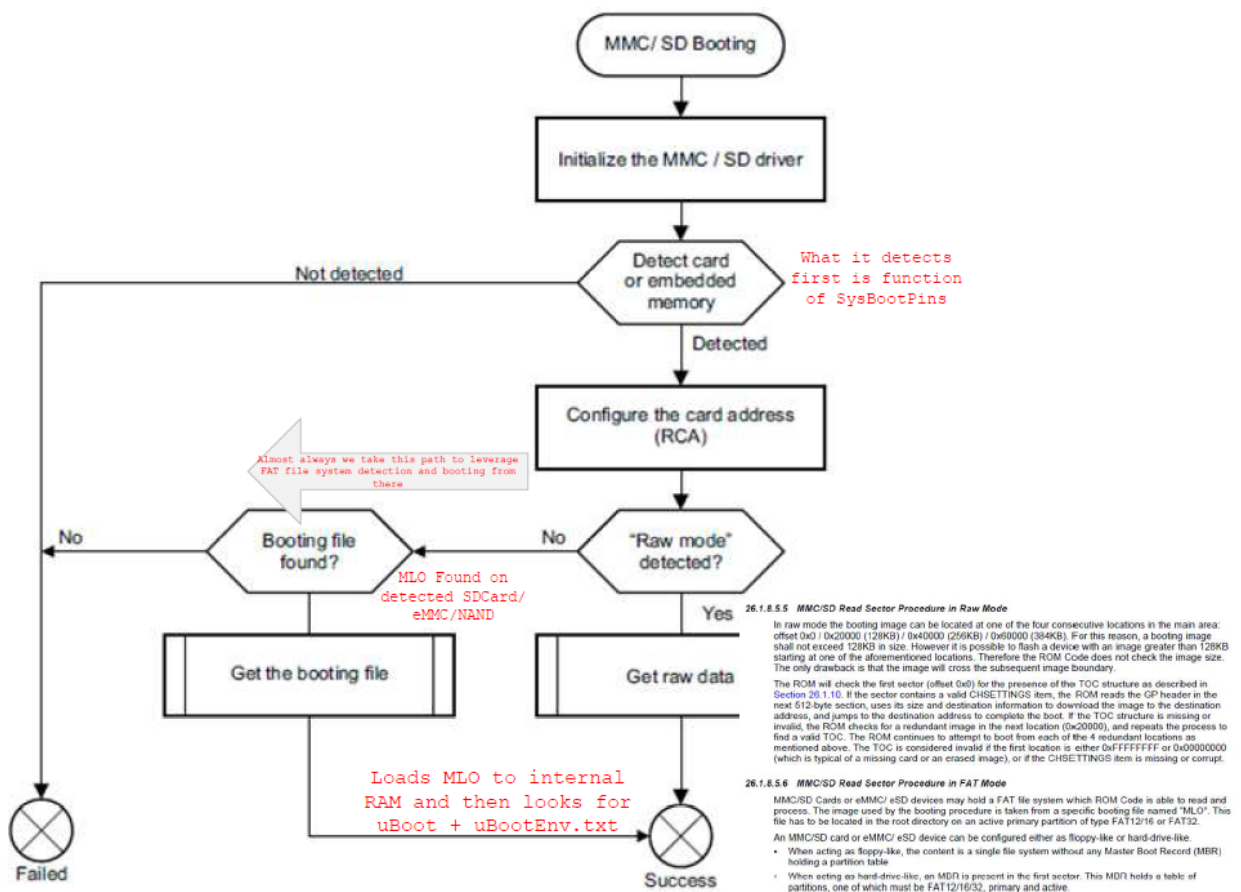**Figure 1-1. AM335x Functional Block Diagram**

Only 176 Kbytes of Internal ROM Space this is where the ROM Code resides

128 Kbytes of Internal Ram (True when we apply generic Boot Mode than Secure Boot Mode)
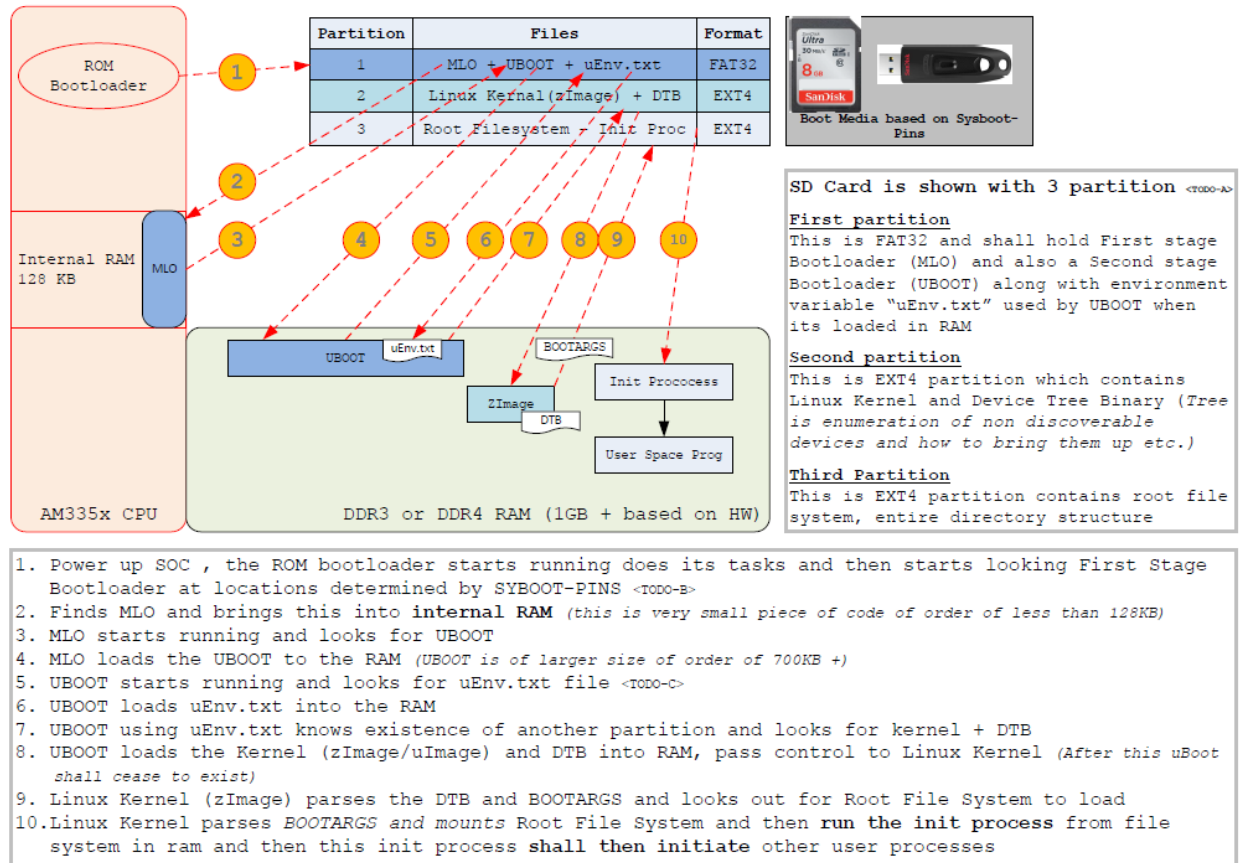
ROM code consists of different booting up processes
- FAT
- BOOTP
- XMODEM
- TFTP

ROM code supports drivers like
- EMAC
- UART
- USB
- SPI
- I2C
- XIP
- NAND
- MMSD

ROM-CODE Startup sequence

mGaik

ROM-CODE Startup sequence - eMMC / SD-card Booting 🔗

Bootup Sequence and What we need to Know & Do 🔗

Partition	Files	Format
1	MLO + UBOOT + uEnv.txt	FAT32
2	Linux Kernal(zImage) + DTB	EXT4
3	Root Filesystem , Init Proc	EXT4

Boot Media based on Sysboot-Pins

SD Card is shown with 3 partition <TODO-A>

First partition
This is FAT32 and shall hold First stage Bootloader (MLO) and also a Second stage Bootloader (UBOOT) along with environment variable "uEnv.txt" used by UBOOT when its loaded in RAM

Second partition
This is EXT4 partition which contains Linux Kernel and Device Tree Binary (*Tree is enumeration of non discoverable devices and how to bring them up etc.*)

Third Partition
This is EXT4 partition contains root file system, entire directory structure

1. Power up SOC , the ROM bootloader starts running does its tasks and then starts looking First Stage Bootloader at locations determined by SYBOOT-PINS <TODO-B>
2. Finds MLO and brings this into **internal RAM** (*this is very small piece of code of order of less than 128KB*)
3. MLO starts running and looks for UBOOT
4. MLO loads the UBOOT to the RAM (*UBOOT is of larger size of order of 700KB +*)
5. UBOOT starts running and looks for uEnv.txt file <TODO-C>
6. UBOOT loads uEnv.txt into the RAM
7. UBOOT using uEnv.txt knows existence of another partition and looks for kernel + DTB
8. UBOOT loads the Kernel (zImage/uImage) and DTB into RAM, pass control to Linux Kernel (*After this uBoot shall cease to exist*)
9. Linux Kernel (zImage) parses the DTB and BOOTARGS and looks out for Root File System to load
10. Linux Kernel parses *BOOTARGS and mounts* Root File System and then **run the init process** from file system in ram and then this init process **shall then initiate** other user processes

References 🔗

https://elinux.org/images/2/26/Enhancing_Real-Time_Capabilities_with_the_PRU.pdf
http://software-dl.ti.com/processor-sdk-linux/esd/docs/latest/linux/How_to_Guides.html

edited Oct 11, 2021 at 2:46

created Oct 9, 2021 at 13:13
mahesh-gaikwad
753 ● 1 ● 11

1 ▲ it is helpful to me understanding boot process – user107 Jan 24, 2022 at 7:03
🚩

1 ▲ There is good video on this that can demonstrate the run time:youtube.com/watch?
🚩 v=DV5S_ZSdK0s – user97 Feb 17, 2022 at 3:48 ✎

Thanks for sharing link, i did refer this link when i was getting my understanding on this subject.
– mahesh-gaikwad Feb 17, 2022 at 3:53