# How to Use an Older Uboot version with 3.1x.xx Kernel Version.

### About this document

This document describe the setup detail to learn how to install a new kernel version which are device tree dependable with an older Uboot version.

### Software versions

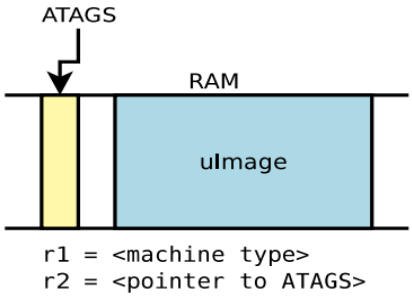
The contents of this document are valid for the latest versions of i.MX6 BSP and older Uboot version than 2014.4v by the time of writing, listed below:

- Uboot 2009
- L3.14.38_6qp_ga

## 1. Introduction

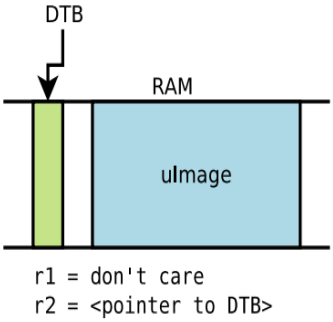
Before The device tree the normal kernel booting process was:

1. Kernel containing the description of the hardware
2. Single Binary file (uImage)
3. Bootloader prepares additional information ATAGS (KERNEL command line).




4.

5. Since kernel version 3.1x and up, device tree is a must for linux embedded devices, where kernel does not contain any hardware description. The hardware description is contained in a different binary file, the device tree blob. Bootloader load two binaries:




## 2. Bootloader DTB Compatible

Not all the bootloaders support the DTB loading process (old Uboot versions), for these cases Kernel configuration changes to ease transition:

CONFIG_ARM_APPENDED_DTB

- The kernel looks for the dtb after the kernel image.
- Developer must add the dtb to your kernel image.

CONFIG_ARM_ATAG_DTB_COMPAT

- Kernel reads ATAGS from Bootloader and pathces device tree with them.

```
cat arch/arm/boot/zImage arch/arm/boot/dts/myboard.dtb > my-zImage
mkimage ... -d my-zImage my-uImage
```

## 3. Setup Examples

In this example we are going to show how to use Uboot 2009 with latest BSP i.MX6Q supported kernel 3.14.38

1. Setup the toolchain:

```
$ cd ~/
```

```
$ wget -c
https://releases.linaro.org/14.09/components/toolchain/binaries/gcc-
linaro-arm-linux-gnueabihf-4.9-2014.09_linux.tar.xz
$ tar xf gcc-linaro-arm-linux-gnueabihf-4.9-2014.09_linux.tar.xz
$ export ARCH=arm
$ export CROSS_COMPILE=gcc-linaro-arm-linux-gnueabihf-4.9-
2014.09_linux/bin/arm-linux-gnueabihf-
$ unset LDFLAGS
```

For these instructions, we are assuming: DISK=/dev/sdg, cat /proc/partitions is very useful for determining the device id.

```
$ export DISK=/dev/sdg
```

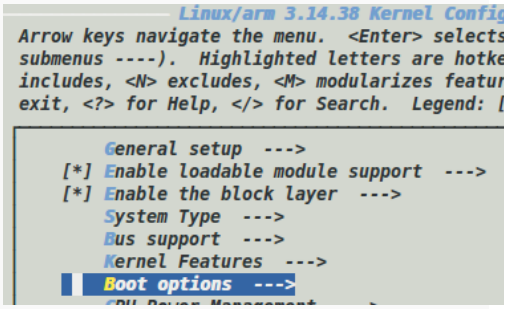
2. Download Uboot 2009 and compile it, and load it to SD:

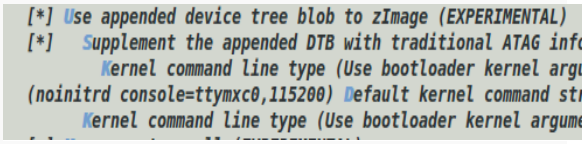
```
$ git clone http://git.freescale.com/git/cgit.cgi/imx/uboot-imx.git -b
imx_v2009.08_3.0.35_4.1.0
$ export ARCH=arm
$ export CROSS_COMPILE= $TARGET_PREFIX
$ make <board>_config
$ make
Erase uSD/SD card:
$ sudo dd if=/dev/zero of=${DISK} bs=1M count=10
Install the bootloader:
$ sudo dd if=uboot.bin of=${DISK} && sync
```

3. Configure Linux Kernel (In this example, L3.14.38 )

```
$ cd ~/
$ wget -c http://git.freescale.com/git/cgit.cgi/imx/linux-2.6-
imx.git/snapshot/linux-2.6-imx-rel_imx_3.14.38_6qp_ga.tar.gz
```

```
$ tar xf linux-2.6-imx-rel_imx_3.14.38_6qp_ga.tar.gz
$ cd linux-2.6-imx-rel_imx_3.14.38_6qp_ga

$ make ARCH=arm CROSS_COMPILE=../gcc-linaro-arm-linux-gnueabihf-4.9-
2014.09_linux/bin/arm-linux-gnueabihf- imx_v7_defconfig

$ make ARCH=arm CROSS_COMPILE=../gcc-linaro-arm-linux-gnueabihf-4.9-
2014.09_linux/bin/arm-linux-gnueabihf- menuconfig

For CONFIG_ARM_APPENDED_DTB, Enable:
-> Boot options
  ->  Use appended device tree blob to zImage

for CONFIG_ARM_ATAG_DTB_COMPAT
-> Boot options
     -> Use appended device tree blob to zImage (EXPERIMENTAL)
       -> Supplement the appended DTB with traditional ATAG information
```




4. Create uImage:

```
$ make ARCH=arm CROSS_COMPILE=../gcc-linaro-arm-linux-gnueabihf-4.9-
2014.09_linux/bin/arm-linux-gnueabihf- -j4 zImage

$ make ARCH=arm CROSS_COMPILE=../gcc-linaro-arm-linux-gnueabihf-4.9-
2014.09_linux/bin/arm-linux-gnueabihf- dtbs
```

```
$ cat arch/arm/boot/zImage arch/arm/boot/dts/imx6q-sabresd.dtb > zImage
$ mkimage -A arm -O linux -T kernel -C none -a 0x10800000 -e 0x10800000
-n "Linux Kernel" -d zImage uImage
```

5. Create your SD system partitions:

```
sudo fdisk ${DISK}
```

- d ///delete all partitions currently on sd
- n // create new partition
- p // Primary partition
- 1 // partition number 1
- 2048 //default
- +1G //
- N // created 2d parition
- p
- 2
- default
- default
- t  //change partition t
- 1 // firts
- B // to be fat32
- W // write partiotions

6. Create partition types:

```
$ sudo mkfs.vfat -n KERNEL ${DISK}1
$ sudo mkfs.ext3  ${DISK}2
```

7. Mount the sd partitions on your host

```
$ sudo mount ${DISK}1 /"mount directory"
```

8. Copy the uImage to a FAT partition of the SD

```
$ sudo cp uImage /"mount directory"
```

9. Umount the SD form the host

```
$ sudo mount /"mount directory"
```

10. Insert the eSD and boot up the board. Stop in Uboot menu.

11. Check your mmc device (for this case mmcdev=2)

12. Load the uImage to memory and boot the Linux kernel

```
$ fatload mmc 2:1 0x10008000 uImage
$ bootm 0x10008000
You should get the next output:
```