# Distro Boot with Boot.scr

## Table of contents



## Introduction

the industry standard "distro boot" boot flow for U-boot has been implemented as default by Xilinx. (see xilinx-wiki.atlassian."Using Distro Boot With Xilinx U-Boot")

The distro boot functionality is implemented as an extension of the existing U-Boot bootcmd functionality in U-Boot. It extends its functionality by redirecting the call to bootcmd to a call to the distro_bootcmd variable.

```
bootcmd=run distro_bootcmd
```

## Distro_bootcmd and Boot_targets

The **distro_bootcmd** variable typically contains a sequence of commands that scans a pre-defined list of potential **boot targets** in search of boot collateral as shown below in an snippet of the printenv.

```
scriptaddr=0x3000000
script_offset_f=fc0000
script_size_f=0x40000

boot_targets=qspi jtag mmc0 mmc1 qspi nand nor usb0 usb1 pxe dhcp
distro_bootcmd=for target in ${boot_targets}; do run bootcmd_${target};
done

bootcmd_qspi=sf probe 0 0 0 && sf read ${scriptaddr} ${script_offset_f}
${script_size_f} && echo QSPI: Trying to boot script at ${scriptaddr} &&
source ${scriptaddr}; echo QSPI: SCRIPT FAILED: continuing...;
bootcmd_mmc0=devnum=0; run mmc_boot
mmc_boot=if mmc dev ${devnum}; then devtype=mmc; run
scan_dev_for_boot_part; fi
```

The **boot_targets** list is defined in the platform definition in the U-Boot source tree.

Depending on the specifics of the target platform, the distro boot infrastructure can either use the **boot_targets** list to explicitly find specific boot collateral (eg. hard-coded loading once a valid boot target is located) or it can be used to locate some kind of decoupled configuration information.

> ⚠ Not all devices listed in the **boot_target** list are available. Which devices can be used depends on
>
> - the target platform used
> - the correct configuration of the hardware in Vivado
> - the correct configuration of the u-boot and the device tree in Petalinux
>
> Configuration examples can be found in the corresponding **reference designs**.

## Boot Method Precedence

The order of precedence for searching for boot components is as follows

1. extlinux.conf file (located in /extlinux/ or /boot/extlinux/)

2. boot.scr file + image.ub file
3. boot.scr file + individual files (Image, system.dtb, etc.)

In absence of a valid extlinux.conf file, U-Boot will scan the boot_targets list looking for a file named **boot.scr.uimg** or **boot.scr** (in that order) and run any commands located in the script file. If your environment requires a different sequence of commands or behaviour, you can edit the **boot.scr** file to suit your needs.

# Using the boot.src method

## Install u-boot-tools on Linux OS

Before using the mkimage function, the u-boot-tools needs to be installed.

```
sudo apt-get update
sudo apt install u-boot-tools
```

## Generate boot.script file from boot.scr file

For the modification of the **boot.scr** file convert the existing **boot.scr** file into an **boot.script** file:

```
dd if=boot.scr of=boot.script bs=72 skip=1
```

## Modifying boot.script file

Modify the **boot.script** file based on your needs. For example, when booting **image.ub** from **SD card**, **USB** or **QSPI flash**, the following **boot.script** file should work after conversion into an **boot.scr** file. The **boot.scr** file could be located on all working **boot_targets**.

**example boot.script file**

```
# This is a boot script for U-Boot
# Generate boot.scr:
# mkimage -c none -A arm -T script -d boot.script boot.scr
#
# Generate boot.script (for modifications in boot.scr):
# dd if=boot.scr of=boot.script bs=72 skip=1
#
#################
imageub_addr=0x10000000
#
imageub_flash_addr=0x200000
imageub_flash_size=0xD90000

for boot_target in ${boot_targets};
do

        # Boot target is mmc0 or usb0: image.ub on mmc0 or usb0
        if test "${boot_target}" = "${devtype}0"; then
                if test -e ${devtype} 0:${distro_bootpart} /image.ub; then
                        fatload ${devtype} 0:${distro_bootpart}
${imageub_addr} image.ub;
                        bootm ${imageub_addr};
                fi
        fi
        # Boot target is mmc1 or usb1: image.ub on mmc1 or usb1
        if test "${boot_target}" = "${devtype}1"; then
                if test -e ${devtype} 1:${distro_bootpart} /image.ub; then
                        fatload ${devtype} 1:${distro_bootpart}
${imageub_addr} image.ub;
                        bootm ${imageub_addr};
                fi
        fi
        # Boot target is qspi: image.ub on qspi; image.ub is included in
BOOT.bin
        if test "${boot_target}" = "qspi" || test "${boot_target}" =
"qspi0"; then
                sf probe 0 0 0;
                sf read ${imageub_addr} ${imageub_flash_addr}
${imageub_flash_size};
                bootm ${imageub_addr};
        fi
done
```

**Note**

The alias name **${devnum},** used in Xilinx default files, does not work in all variants tested, therefore the **${devtype}** boot_targets were split up

## Generate boot.scr file from boot.script file

Please remember that the **boot.script** file needs to be converted back into the **boot.scr** file after the edits are complete

```
mkimage -c none -A arm -T script -d boot.script boot.scr
```

# Distro Boot for Zynq-7000 series

## SD-Boot mode

The following table provides an overview of the possible boot combinations for the devices SD card, USB and QSPI in SD Boot mode. The table is not complete, but gives an impression of the possibilities of the Distro Boot.

| Boot Variants | SD-Card | USB-Stick | QSPI-Flash | Comments |
|---|---|---|---|---|
| **Variant 1** | BOOT.bin[1]<br>image.ub<br>boot.scr<br>Init.sh | -- | -- | recommended workflow |
| **Variant 2** | BOOT.bin[1]<br>Init.sh | image.ub<br>boot.scr | -- | used **boot.scr** and **image.ub** from **USB** |
| **Variant 3** | BOOT.bin[1]<br>image.ub<br>Init.sh | -- | BOOT.bin[2] | used **boot.scr** from **QSPI-Flash** and **image.ub** from **SD card** |
| **Variant 4** | BOOT.bin[1]<br>boot.scr<br>Init.sh | -- | BOOT.bin[2] | used **boot.scr** from **SD card** and **image.ub** from **QSPI-Flash** |
| **Variant 5** | BOOT.bin[1]<br>image.ub<br>Init.sh | -- | BOOT.bin[3] | used **boot.scr** from **QSPI-Flash** and **image.ub** from **SD card** |

**Table of possible boot variants in SD-Boot mode**

| File name | BOOT.bin[1] | BOOT.bin[2] | BOOT.bin[3] | QSPI Flash Offset Address | RAM Load Address |
|---|---|---|---|---|---|
| **fsbl.elf** | x | x | x | -- | NA |
| **test_board.bit** | x | x | x | -- | NA |
| **u-boot.elf** | x | x | x | -- | NA |
| **image.ub** | | x | | **0x200000*** | **0x10000000*** |
| **boot.scr** | | x | x | **0xfc0000*** | **0x3000000*** |

**Possible content of BOOT.bin files for qspi flash or sd/usb devices**

**Note**

*When booting from a QSPI device, care must be taken to maintain a very specific scheme for where files are located both in the nonvolatile flash device as well as where they are loaded into DDR prior to hand-off to the robust operating system.

## Variant 1 (Boot.bin, image.ub and boot.scr on SD card):

1. Copy image.ub, boot.scr and Boot.bin on SD-Card.
2. Set Boot Mode to SD-Boot.
3. Insert SD-Card in SD-Slot and start the board.

## Variant 2 (Boot.bin on SD card; image.ub and boot.scr on USB):

1. Copy image.ub on SD-Card.
2. Copy boot.scr and Boot.bin on USB Stick
3. Set Boot Mode to SD-Boot.
4. Insert SD-Card in SD-Slot.
5. Insert USB-Stick in USB-Slot and start the board.

# QSPI-Boot mode

The following table provides an overview of the possible boot combinations for the devices SD card, USB and QSPI in QSPI Boot mode. The table is not complete, but gives an impression of the possibilities of the Distro Boot.

| Boot Variants | SD-Card | USB-Stick | QSPI-Flash | Comments |
|---|---|---|---|---|
| **Variant 1** | -- | -- | BOOT.bin[2]<br>zynq_fsbl_flash | Boot only from **QSPI flash** |
| **Variant 2** | image.ub<br>boot.scr | -- | BOOT.bin[1]<br>zynq_fsbl_flash | used **boot.scr** and **image.ub** from **SD card** |
| **Variant 3** | -- | image.ub<br>boot.scr | BOOT.bin[1]<br>zynq_fsbl_flash | used **boot.scr** and **image.ub** from **USB** |
| **Variant 4** | BOOT.bin[1]<br>image.ub | -- | BOOT.bin[3]<br>zynq_fsbl_flash | used **boot.scr** from **QSPI-Flash** and **image.ub** from **SD card** |

**Table of possible boot variants in QSPI-Boot mode**

| File name | BOOT.bin[1] | BOOT.bin[2] | BOOT.bin[3] | QSPI Flash Offset Address | RAM Load Address |
|---|---|---|---|---|---|
| **fsbl.elf** | x | x | x | -- | NA |
| **test_board.bit** | x | x | x | -- | NA |
| **u-boot.elf** | x | x | x | -- | NA |
| **image.ub** | | x | | **0x200000*** | **0x10000000*** |
| **boot.scr** | | x | x | **0xfc0000*** | **0x3000000*** |

**Possible content of BOOT.bin files for qspi flash or sd/usb devices**

**Note**

*When booting from a QSPI device, care must be taken to maintain a very specific scheme for where files are located both in the nonvolatile flash device as well as where they are loaded into DDR prior to hand-off to the robust operating system.

### Variant 1 (Boot.bin on QSPI Flash; image.ub and boot.scr on SD card):

1. Connect JTAG and power on carrier with module (boot mode set to **SD boot** and **no SD card** is inserted)
2. Open Vivado Project and program **QSPI flash**
3. Copy **boot.scr** and **image.ub** on **SD card**
4. Set boot mode to **QSPI-Boot mode**, insert **SD card** and start the board.

### Variant 2 (Boot.bin, image.ub and boot.scr on QSPI Flash):

1. Connect JTAG and power on carrier with module (boot mode set to **SD boot** and **no SD card** is inserted)
2. Open Vivado Project and program **QSPI flash**
3. Set boot mode to **QSPI-Boot mode** and start the board.

### Variant 3 (Boot.bin on QSPI Flash; image.ub and boot.scr on USB):

1. Connect JTAG and power on carrier with module (boot mode set to **SD boot** and **no SD card** is inserted)
2. Open Vivado Project and program **QSPI flash**
3. Copy **boot.scr** and **image.ub** on **SD card**.
4. Set boot mode to **QSPI-Boot mode**, insert **USB** and start the board.

## Code Snippets for Boot.scr file

**image.ub on sd card**

```
        # Boot target is mmc0: image.ub on mmc0
        if test "${boot_target}" = "mmc0"; then
                mmc dev 0
                if test -e mmc 0:${distro_bootpart} /image.ub; then
                        echo [TE_BOOT-20] boot_target is ${boot_target}.
Found image.ub on mmc0:${distro_bootpart} ;
                        fatload mmc 0:${distro_bootpart} ${imageub_addr}
image.ub;
                        bootm ${imageub_addr};
                fi
        fi
        # Boot target is mmc1: image.ub on mmc1
        if test "${boot_target}" = "mmc1"; then
                mmc dev 1
                if test -e mmc 1:${distro_bootpart} /image.ub; then
                        echo [TE_BOOT-21] boot_target is ${boot_target}.
Found image.ub on mmc1:${distro_bootpart};
                        fatload mmc 1:${distro_bootpart} ${imageub_addr}
image.ub;
                        bootm ${imageub_addr};
                fi
        fi
```

**Note**

The alias name **${devnum},** used in Xilinx default files, does not work in all variants tested, therefore the **${devtype}** boot_targets were split up

**image.ub on sd card**

```
        # Boot target is mmc0 or usb0: image.ub on mmc0 or usb0
        if test "${boot_target}" = "${devtype}0"; then
                if test -e ${devtype} 0:${distro_bootpart} /image.ub; then
                        echo [TE_BOOT-20] boot_target is ${boot_target}.
Found image.ub on ${devtype}0:${distro_bootpart} ;
                        fatload ${devtype} 0:${distro_bootpart}
${imageub_addr} image.ub;
                        bootm ${imageub_addr};
                fi
        fi
        # Boot target is mmc1 or usb1: image.ub on mmc1 or usb1
        if test "${boot_target}" = "${devtype}1"; then
                if test -e ${devtype} 1:${distro_bootpart} /image.ub; then
                        echo [TE_BOOT-21] boot_target is ${boot_target}.
Found image.ub on ${devtype}1:${distro_bootpart};
                        fatload ${devtype} 1:${distro_bootpart}
${imageub_addr} image.ub;
                        bootm ${imageub_addr};
                fi
        fi
```

> **Note**
>
> The alias name **${devnum},** used in Xilinx default files, does not work in all variants tested, therefore the **${devtype}** boot_targets were split up

**image.ub on usb0**

```
        # image.ub on usb0
        if test "${boot_target}" = "usb0"; then
                if test -e usb 0:${distro_bootpart} /image.ub; then
                        echo [TE_BOOT-20] boot_target is ${boot_target}.
Found image.ub on usb0:${distro_bootpart} ;
                        fatload usb 0:${distro_bootpart} ${imageub_addr}
image.ub;
                        bootm ${imageub_addr};
                fi
        fi
        # image.ub on usb1
        if test "${boot_target}" = "usb1"; then
                if test -e usb 1:${distro_bootpart} /image.ub; then
                        echo [TE_BOOT-21] boot_target is ${boot_target}.
Found image.ub on usb1:${distro_bootpart};
                        fatload usb 1:${distro_bootpart} ${imageub_addr}
image.ub;
                        bootm ${imageub_addr};
                fi
        fi
```

**image.ub on usb0**

```
        # Boot target is qspi: image.ub on qspi; image.ub is included in
BOOT.bin
        if test "${boot_target}" = "qspi"; then
                echo [TE_BOOT-30] Try to use image.ub from ${boot_target},
load image.ub from qspi-flash address ${imageub_flash_addr};
                echo [TE_BOOT-30] max. image.ub size is
${imageub_flash_size} ;
                sf probe 0 0 0;
                sf read ${imageub_addr} ${imageub_flash_addr}
${imageub_flash_size};
                bootm ${imageub_addr};
        fi
```

# References


https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/749142017/Using+Distro+Boot+With+Xilinx+U-Boot#UsingDistroBootWithXilinxU-Boot-BootTargets

https://wiki.ubuntu.com/ARM/EditBootscr