Support Booting from QSPI Flash in Linux 6.1 BSP for the i.MX RT1170 targets

Detailed Requirements and Design

rm6664-drad-1_1.doc

TABLE OF CONTENTS

1. Overview

The following is a high-level overview of the problem being resolved by this project:

This project develops support for booting from QSPI Flash in the Linux i.MX RT1170 BSP.

2. Requirements

2.1. Detailed Requirements

The following are the requirements for this project:

1. Support booting of U-Boot from QSPI Flash, with no reliance on presence of SD Card or any other storage devices.
    - *Rationale*: Explicit customer requirement.
      Implementation: Section: "Design: U-Boot Boot from QSPI Flash ".
      Test: Section: "Test Plan: U-Boot Boot from QSPI Flash".

2. Support the U-Boot standard SPI Flash commands (the `sf` commands family) for QSPI Flash.
    - *Rationale*: Explicit customer requirement.
      Implementation: Section: "Design: U-Boot sf Commands ".
      Test: Section: "Test Plan: U-Boot sf Commands".

3. Store the U-Boot environment in QSPI Flash.
    - *Rationale*: Explicit customer requirement.
      Implementation: Section: "Design: U-Boot Environment in QSPI Flash".
      Test: Section: "Test Plan: U-Boot Environment in QSPI Flash".

4. Support installation of the U-Boot images to the QSPI Flash using the IMXRT1170 Serial Downloader feature.
    - *Rationale*: Explicit customer requirement.
      Implementation: Section: "Design: Install U-Boot Images to the QSPI Flash using Serial Downloader".
      Test: Section: "Test Plan: Install U-boot Images to QSPI Flash".

5. Support installation of the Linux images to QSPI Flash from the networking from the U-Boot command line interface.
    - *Rationale*: Explicit customer requirement.
      Implementation: Section: "Design: Install Linux Images to the QSPI Flash using U-Boot command line interface".
      Test: Section: "Test Plan: Install Linux Images (JFFS2 variant) to QSPI Flash" Section: "Test Plan: Install Linux Images (UBIFS variant) to QSPI Flash".

6. Support Linux boot from QSPI Flash, in the following configuration:
    - multiimage including kernel and DTB loaded from QSPI Flash to RAM for execution;
    - root file system mounted in QSPI Flash as the read-write Flash file system (JFFS2 or UBIFS optionally).

      Rationale: Explicit customer requirement.
      Implementation: Section: "Design: Linux Boot from QSPI Flash ".

Test: Section: "Test Plan: Linux Boot from QSPI Flash (JFFS2 variant)" Section: "Test Plan: Linux Boot from QSPI Flash (UBIFS variant)".

7. Support QSPI Flash in Linux. This must include support for Linux Flash file system.
    - *Rationale*: Explicit customer requirement.
      Implementation: Section: "Design: Linux Device Driver for QSPI Flash and Flash File System ".
      Test: Section: "Test Plan: Linux Device Driver for QSPI Flash and Flash File System (JFFS2 variant)" Section: "Test Plan: Linux Device Driver for QSPI Flash and Flash File System (UBIFS variant)".

2.2. Detailed Non-Requirements

The following are the non-requirements for this project that may otherwise not be obvious:

1. Support for any Flash devices other than the QSPI Flash device present on the NXP IMXRT1170-EVK boards is not required.
    - *Rationale*: Costs reduction measure.
2. Support for Linux boot scenarios other than the one listed in Section: "Detailed Requirements" is not required.
    - *Rationale*: Costs reduction measure.

3. Design

3.1. Design: U-Boot Boot from QSPI Flash

A dedicated configuration file `imxrt1170-evk-qspi_defconfig` will be added to U-Boot to support booting from the QSPI Flash on the NXP IMXRT1170-EVK board. The standard build procedure will be used to generate the bootable SPL and TPL images. The `Makefile` in the U-Boot source tree will be updated to generate the special header and prepend it to the general `SPL` image in order to make it bootable from the QSPI Flash if the corresponding option is enabled in the config file, the resulting file will be saved with the ".flexspi" extension in the filename. So in case of the `imxrt1170-evk-qspi` configuration the resultunt binaries from the `make` command for U-Boot will be `SPL.flexspi` and `u-boot.img` images:

```
make mxrt1170-evk-qspi_defconfig
make
ls SPL.flexspi u-boot.img
```

The `SPL.flexspi` image must be programmed to the QPSI flash on the NXP IMXRT1170-EVK board at offset 0. The image consists of the U-Boot SPL and two i.MXRT1170-specific headers:

- Image Vector Table (IVT);
- The FlexSPI Configuration Block.

The Image Vector Table is generated by `mkimage` using the `board/freescale/imxrt1170-evk/imximage.cfg` configuration file.

The The FlexSPI Configuration Block is compiled from the `board/freescale/imxrt1170-evk/flexspi_cb.c` file. This file contains the FlexSPI Configuration Block parameters, as per the corresponding Processor Reference Manual.

The `u-boot.img` image is the the U-Boot TPL itself. It must be programmed to the QPSI flash on the NXP IMXRT1170-EVK board at offset 0x10000.

The default configuration will be set up to support the ISSI QSPI Flash installed on the EVK board.

3.2. Design: U-Boot sf Commands

The standard U-Boot `sf` commands will be enabled in the U-Boot configuration to support the SPI Flash read, erase and write operations.

3.3. Design: U-Boot Environment in QSPI Flash

Whenever the corresponding configuration described above is selected in U-Boot, the U-Boot environment will be stored in the QSPI Flash.

The environment, along with the redundant environment copy, will be placed at the address range `0x60000` - `0x80000` in the QSPI Flash.

3.4. Design: Install U-Boot Images to the QSPI Flash using Serial Downloader

The QSPI Flash device will be logically divided into 4 sections to store the software components of the system:

- `0x000000 - 0x060000` - U-Boot (both SPL + TPL images)
- `0x060000 - 0x080000` - U-Boot Environment
- `0x080000 - 0x880000` - Kernel Image (multi image including the kernel and the device tree blob)
- `0x880000 - 0x1000000` - Root File System

The IMXRT1170 Serial Downloader feature can be used to write the SPL and TPL images to U-Boot section of the QSPI Flash on the IMXRT1170-EVK board. The `SPL.flexspi` image must be installed at offset 0 of the QSPI Flash, the `u-boot.img` image must be installed at offset 0x10000 of the QSPI Flash.

The NXPs `MCUXpresso Secure Provisioning Tool` (actual version for Nov 2023 is v7) can be used to install the SPL and TPL image to the QSPI Flash. The tool is available for download at NXP's web site: https://www.nxp.com/design/design-center/software/development-software/mcuxpresso-software-and-tools-/mcuxpresso-secure-provisioning-tool:MCUXPRESSO-SECURE-PROVISIONING This tool provides graphical user interface to program images to the on board mass storage for various NXP's EVK boards.

For user convenience, the dedicated installation script `imxtr1170_install_uboot.sh` will be provided for installing U-Boot using the command line interface. The script will invoke only the `blhost` utility from the `Secure Provisioning Tool`, so running the whole tool set is not necessary. The `flashloader` binary required by `blhost` will be provide along with the installation script. The script and the `flashloader` must be installed to the `tools/bin` directory in top of the the `linux-cm-imxrt117x-3.0.3` installation.

Script usage is as foolows:

```
$ ./imxtr1170_install_uboot.sh [-f <path the flashloader.bin>] [-p <path to the Secure
Provisioning Tool>] [-u <path to U-Boot>] [-v] [-h]
```

Where:

- `f` option - override the default flashloader binary file;

- `p` option - specify alternative path to the installation of the MCUXpress Secure Provisioning Tool;
- `u` option - specify alternative path to the directory containing the U-Boot binaries;
- `v` option - enable verbose;
- `h` option - help;

3.5. Design: Install Linux Images to the QSPI Flash using U-Boot command line interface

The QSPI Flash device will be logically divided into 4 sections to store the software components of the system:

- `0x000000 - 0x060000` - U-Boot (both SPL + TPL images)
- `0x060000 - 0x080000` - U-Boot Environment
- `0x080000 - 0x880000` - Kernel Image (multi image including the kernel and the device tree blob)
- `0x880000 - 0x1000000` - Root File System

The following commands will be defined in the U-Boot environment to update the `kernel` and `rootfs` components:

- `sf_kernel_update` - Update the Kernel section
- `sf_rootfs_update` - Update the RootFS section

For the NXP i.MXRT1170 BSP the `sf_*_update` commands will download images from the TFTP server via networking and install them to the corresponding section in the QSPI Flash. The names of the images to download will be constructed automatically using the following U-Boot environment variables:

- `project` - is the project name, default is `rootfs_flash`
- `fstype` - is the type of the flash file system image : `jffs2` or `ubi`, default is `jffs2`

The name of the kernel multiimage in the `sf_kernel_update` command is `${project}.uImage`, i.e. `rootfs_flash.uImage` in the default settings. The name of the rootfs image in the `sf_rootfs_update` command is `${project}.${fstype}`, i.e. `rootfs_flash.jffs2` in the default settings.

3.6. Design: Linux Boot from QSPI Flash

The separate project `projects/rootfs_flash` will be created to demonstrate booting Linux from QSPI flash. The following main feature will be enabled in the new project:

- Support for the QSPI Flash will be enabled in the kernel configuration.
- `initramfs` will be disabled in the kernel configuration. Instead the root filesystem will be mounted on JFSS2 or alternatively on an UBIFS file system in the QSPI Flash.
- The kernel and the DTB images will be built in a single multi-part image.

To implement these features the following options will be added to the common build rules and will be used in `Makefile` for the `rootfs_flash` project:

- `RFS_BUILD_DIR` - temporary directory to build the root file system image
- `FLASHFS_TYPE` - to select certain file system image to build: `jffs2` or `ubi`
- `MKFSUBIFS_FLAGS` - Flash-specific flags for the `mkfs.ubifs` utility (if `FLASHFS_TYPE` is `ubi`)
- `UBINIZE_FLAGS` - Flash-specific flags for the `ubinize` utility (if `FLASHFS_TYPE` is `ubi`)

- `SEPARATE_DTB` - tells the make either build the multi-part image or save the DTB separately.

3.7. Design: Linux Device Driver for QSPI Flash and Flash File System

The existing `nxp-fspi` driver from the common NXP's codebase will be used to provide support for the FlexSPI controller. The driver will be updated to support the i.MXTRT1170 SoC.

The `MTD`, `SPI_MEM`, `MTD_SPI_NOR` kernel features will be enabled in the `rootfs_flash` project's configurations, so that the QSPI Flash will be available in Linux as a standard `MTD` device.

The `UBI`, `UBIFS` and `JFFS2` related options will be enabled to support for Linux Flash file systems.

4. Test Plan

4.1. Secure Download Area

The downloadable materials developed by this project are available from a secure Web page on the Emcraft Systems web site. Specifically, proceed to the following URL to download the software materials.

for the i.MX RT1170 BSP:

- https://www.emcraft.com/imxrtaddon/imxrt1170-3.0.3/qspi/

The page is protected as follows:

- Login: *CONTACT EMCRAFT FOR DETAILS*
- Password: *CONTACT EMCRAFT FOR DETAILS*

4.2. Downloadable Files

The following files are available from the secure download area for this release:

- `SPL.flexspi` - Bootable U-Boot SPL image to be installed to the QSPI Flash.
- `u-boot.img` - U-Boot TPL image to be installed to the QSPI Flash.
- `rootfs_flash.uImage` - Multiimage with Linux kernel and DTB.
- `rootfs_flash.jffs2` - JFFS2 image with Linux rootfs.
- `rootfs_flash.ubi` - UBIFS image with Linux rootfs.
- `u-boot.patch` - Source code patch to U-Boot.
- `linux.patch` - Source code patch to Linux.
- `projects.patch` - Source code patch to `projects/`.
- `imxrt1170_install_uboot.sh` - Script to install the U-Boot images to the QSPI Flash using SDP.
- `unsigned_MIMXRT1176_flashloader.bin` - The prebuilt `flashloader` binary from NXP's `MCUXpresso Secure Provisioning Tool v6`.

4.3. Test Set-Up

4.3.1. Hardware Set-Up

The following hardware set-up is required for execution of the test plan in this project:

- A development host Linux PC.

- The NXP IMXRT1170-EVK board.
- The NXP IMXRT1170-EVK Board is connected to LAN using the J3 100M ENET connector.
- The NXP IMXRT1170-EVK Board is connected to the development host via the J11 micro-USB connector to provide Serial Console.
- The NXP IMXRT1170-EVK Board is connected to the development host via the J20 micro-USB connector for Serial Downloader.

4.3.2. Software Set-Up

MCUXpresso Secure Provisioning Tool

1. Download and install the `MCUXpresso Secure Provisioning Tool v7` from https://www.nxp.com/design/design-center/software/development-software/mcuxpresso-software-and-tools-/mcuxpresso-secure-provisioning-tool:MCUXPRESSO-SECURE-PROVISIONING

Flashloader:

1. Copy the `imxrt1170_install_uboot.sh` script along with the `unsigned_MIMXRT1176_flashloader.bin` binary to the `tools/bin` directory on the top of the `linux-cm-imxrt117x-3.0.3` installation:

```
$ cd linux-cm-imxrt117x-3.0.3
$ cp ~/Downloads/imxrt1170_install_uboot.sh tools/bin
$ cp ~/Downloads/unsigned_MIMXRT1176_flashloader.bin tools/bin
```

U-Boot Build:

1. Apply the U-Boot patch from the top of the fresh `linux-cortexm` installation:

```
$ cd u-boot
$ patch -p1 < ../u-boot.patch
```

2. Build the U-Boot images bootable from QSPI Flash
    1. Configure and build U-Boot for QSPI Flash
        - for the IMXRT1170 EVK board:

```
$ make mrproper
$ make imxrt1170-evk-qspi defconfig
$ make
```

Linux Build:

1. Apply the Linux and `projects/` patches:

```
$ cd linux
$ patch -p1 < ../linux.patch
$ cd ../projects
$ patch -p1 < ../projects.patch
```

2. Build the `rootfs_flash` project in the default `jffs2` configuration:

```
$ cd rootfs flash
$ make
```

3. Redefine `FLASHFS_TYPE` as `ubi` to build the `rootfs_flash` project in the `ubi` configuration:

```
$ cd rootfs_flash
```

```
$ make FLASHFS_TYPE=ubi
```

4. Copy the resultant images to a TFTP server:

```
$ sudo cp rootfs_flash.uImage rootfs_flash.jffs2 rootfs_flash.ubi ~/tmp/
```

Prebuilt Binaries: For convenience, the prebuilt binaries resulting from the above build procedure are available in the area documented in Section: "Downloadable Files"

4.4. Detailed Test Plan

4.4.1. Test Plan: U-Boot Boot from QSPI Flash

The following step-wise test procedure will be used:

1. Power off the target board.
2. Set-up the SW1 switch on the target board for the Serial Downloader mode (SW1/3 = OFF, SW1/4 = ON)
3. Power on the target board.
4. Run the provided imxtr1170_install_uboot.sh in terminal with activated cross-development environment:

```
./imxrt1170_install_uboot.sh -v
```

5. Power off the target board.
6. Set-up the SW1 and SW2 switches on the target board to boot from QSPI Flash (SW1/1 = OFF, SW1/2 = OFF, SW1/3 = ON, SW1/4 = OFF, SW2 all OFF)
7. Power on the target board.
8. Validate that U-Boot has successfully booted from QSPI Flash:

```
U-Boot SPL 2023.04-00036-g038e51ec38b-dirty (Nov 09 2023 - 15:50:58 +0300)
Trying to boot from NOR
U-Boot 2023.04-00036-g038e51ec38b-dirty (Nov 09 2023 - 15:50:58 +0300)
Model: NXP imxrt1170-evk board
DRAM:  960 KiB (effective 64.9 MiB)
Core:  79 devices, 17 uclasses, devicetree: separate
MMC:   FSL_SDHC: 0
Loading Environment from SPIFlash... SF: Detected is25wp128 with page size 256 Bytes, erase
size 4 KiB, total 16 MiB
OK
In:    serial@4007c000
Out:   serial@4007c000
Err:   serial@4007c000
Net:   eth0: ethernet@40424000
Hit any key to stop autoboot:  0
=>
```

4.4.2. Test Plan: U-Boot sf Commands

The following step-wise test procedure will be used:

1. Boot U-Boot from QSPI Flash.
2. Probe the QSPI Flash. Make sure that the correct Flash info is printed out to the console:

```
=> sf probe 0
SF: Detected is25wp128 with page size 256 Bytes, erase size 4 KiB, total 16 MiB
=>
```

3. Read the U-Boot partition to RAM:

```
=> sf read 0x80000000 0 0x60000
device 0 offset 0x0, size 0x60000
SF: 327680 bytes @ 0x0 Read: OK
=>
```

4. Make sure the FlexSPI Configuration Block is at the offset 0x400 of the read data: the 4 symbols at offset 0x400 must be "FCFB" for the IMXRT1170-EVK boards:

```
=> md 0x80000400 1
80000400: 42464346                              FCFB
=>
```

5. Erase an area in the middle of QSPI Flash:

```
=> sf erase 0x300000 0x60000
SF: 393216 bytes @ 0x300000 Erased: OK
=>
```

6. Write the U-Boot image to the erased area:

```
=> sf write 0x80000000 0x300000 0x60000
device 0 offset 0x300000, size 0x60000
SF: 393216 bytes @ 0x300000 Written: OK
=>
```

7. Read it back to a separate area in RAM:

```
=> sf read 0x81000000 0x300000 0x60000
device 0 offset 0x300000, size 0x60000
SF: 393216 bytes @ 0x300000 Read: OK
=>
```

8. Make sure that the data in 2 areas are identical:

```
=> cmp.b 0x80000000 0x81000000 0x60000
Total of 393216 byte(s) were the same
=>
```

4.4.3. Test Plan: U-Boot Environment in QSPI Flash

The following step-wise test procedure will be used:

1. Define and save a test variable:

```
=> setenv testvar testval
=> savee
Saving Environment to SPIFlash... Erasing SPI flash...Writing to SPI flash...done
Valid environment: 2
OK
=>
```

2. Reset the board:

```
=> reset
```

3. Make sure the test variable exists and has the correct value:

```
=> print testvar
testvar=testval
=>
```

4.4.4. Test Plan: Install U-boot Images to QSPI Flash

The following step-wise test procedure will be used:

1. The procedure for initial installation of the U-Boot imaged described in Section: "Test Plan: U-Boot Boot from QSPI Flash" can be used to re-install U-boot on the working board, another components, including the U-Boot environment, linux kernel and rootfs will not be damaged by this:

```
$ imxrt1170 install uboot.sh
+ [ x != x ]
+ MCUX PROVI PATH=/opt/nxp/MCUX Provi v7
+ [ x != x ]
+ UBOOT_PATH=/home/user/imxrt/1170/linux-cm-imxrt117x-3.0.3/u-boot
+ QSPI_SPL_BASEADDR=0x30000000
+ QSPI TPL BASEADDR=0x30010000
+ UBOOT TOTAL SIZE=0x60000
+ SPL IMG=SPL.flexspi
+ TPL_IMG=u-boot.img
+ FLASHLOADER_BIN=/home/user/imxrt/1170/linux-cm-imxrt117x-
3.0.3/tools/bin/unsigned MIMXRT1176 flashloader.bin
+ BLHOST=/opt/nxp/MCUX Provi v7/bin/tools/spsdk/blhost
+ rom connect=-u 0x1FC9,0x013D
+ blhost connect=-u 0x15A2,0x0073
+ /opt/nxp/MCUX_Provi_v7/bin/tools/spsdk/blhost -u 0x1FC9,0x013D -- load-image
/home/user/imxrt/1170/linux-cm-imxrt117x-3.0.3/tools/bin/unsigned_MIMXRT1176_flashloader.bin
Loading image  [####################################]  100%
Response status = 0 (0x0) Success.
+ sleep 3
+ /opt/nxp/MCUX Provi v7/bin/tools/spsdk/blhost -u 0x15A2,0x0073 -- fill-memory 0x2000 4
0xCF900001 word
Response status = 0 (0x0) Success.
+ /opt/nxp/MCUX Provi v7/bin/tools/spsdk/blhost -u 0x15A2,0x0073 -- configure-memory 9 0x2000
Response status = 0 (0x0) Success.
+ /opt/nxp/MCUX Provi v7/bin/tools/spsdk/blhost -u 0x15A2,0x0073 -- fill-memory 0x2000 4
0xC0000007 word
Response status = 0 (0x0) Success.
+ /opt/nxp/MCUX_Provi_v7/bin/tools/spsdk/blhost -u 0x15A2,0x0073 -- configure-memory 9 0x2000
Response status = 0 (0x0) Success.
+ /opt/nxp/MCUX Provi v7/bin/tools/spsdk/blhost -u 0x15A2,0x0073 -t 50000 -- flash-erase-
region 0x30000000 0x60000
Response status = 0 (0x0) Success.
+ /opt/nxp/MCUX_Provi_v7/bin/tools/spsdk/blhost -u 0x15A2,0x0073 -- write-memory 0x30000000
/home/user/imxrt/1170/linux-cm-imxrt117x-3.0.3/u-boot/SPL.flexspi
Writing memory  [####################################]  100%
Response status = 0 (0x0) Success.
Response word 1 = 55064 (0xd718)
+ /opt/nxp/MCUX Provi v7/bin/tools/spsdk/blhost -u 0x15A2,0x0073 -- write-memory 0x30010000
/home/user/imxrt/1170/linux-cm-imxrt117x-3.0.3/u-boot/u-boot.img
Writing memory  [####################################]  100%
Response status = 0 (0x0) Success.
Response word 1 = 300580 (0x49624)
```

4.4.5. Test Plan: Install Linux Images (JFFS2 variant) to QSPI Flash

The following step-wise test procedure will be used:

1. Boot U-Boot from QSPI Flash.
2. Reset the environment:

```
=> env default -f -a
=> saveenv
=>
```

3. Set IP addresses as per actual networking settings:

```
=> setenv ipaddr 192.168.1.117
=> setenv serverip 192.168.1.73
=> setenv gatewayip 192.168.1.1
=> saveenv
=>
```

4. Make sure the `JFFS2` file system type is selected:

```
=> printenv fstype
fstype=jffs2
=>
```

5. Install the software components:

```
=> sf probe 0
SF: Detected is25wp128 with page size 256 Bytes, erase size 4 KiB, total 16 MiB
=> run sf_kernel_update
Using ethernet@40424000 device
TFTP from server 192.168.1.73; our IP address is 192.168.1.117
Filename 'imxrt1170/rootfs flash.uImage'.
Load address: 0x80007fc0
Loading: ##################################################################
         ##################################################################
         ##################################################################
         ##################################################################
         ##################################################################
         ##################################################################
         ##################################################################
         ##################################################################
         ##################################################################
         ##################################################################
         ##################################################################
         ##################################################################
         ##################################################################
         ##################################################################
         ##################################################################
         ##############################
         2.9 MiB/s
done
Bytes transferred = 5467251 (536c73 hex)
SF: 8388608 bytes @ 0x80000 Erased: OK
device 0 offset 0x80000, size 0x536c73
SF: 5467251 bytes @ 0x80000 Written: OK
=> run sf rootfs update
Using ethernet@40424000 device
TFTP from server 192.168.1.73; our IP address is 192.168.1.117
Filename 'imxrt1170/rootfs_flash.jffs2'.
Load address: 0x80007fc0
Loading: ##################################################################
         ##################################################################
         2.9 MiB/s
done
Bytes transferred = 655552 (a00c0 hex)
SF: 7864320 bytes @ 0x880000 Erased: OK
device 0 offset 0x880000, size 0xa00c0
SF: 655552 bytes @ 0x880000 Written: OK
=>
```

4.4.6. Test Plan: Linux Boot from QSPI Flash (JFFS2 variant)

The following step-wise test procedure will be used:

⚠This item requires the JFFS2 image installed to the QSPI Flash as per Section: "Test Plan: Install Linux Images (JFFS2 variant) to QSPI Flash" and the `fstype` variable is set as `jffs2` in the U-Boot environment.

- Reset the board and make sure it automatically boots up to `busybox`:

```
=> reset
...
This architecture does not have kernel memory protection.
Run /sbin/init as init process
/ #
```

4.4.7. Test Plan: Linux Device Driver for QSPI Flash and Flash File System (JFFS2 variant)

The following step-wise test procedure will be used:

⚠This item requires the JFFS2 image installed to the QSPI Flash as per Section: "Test Plan: Install Linux Images (JFFS2 variant) to QSPI Flash" and the `fstype` variable is set as `jffs2` in the U-Boot environment.

- Boot from the QSPI Flash up the `busybox` shell.
- Make sure that the rootfs partition is mounted as the Linux root file system:

```
/ # mount
mtd3 on / type jffs2 (rw,relatime)
devtmpfs on /dev type devtmpfs (rw,relatime)
proc on /proc type proc (rw,relatime)
sysfs on /sys type sysfs (rw,relatime)
devpts on /dev/pts type devpts (rw,relatime,gid=5,mode=620,ptmxmode=000)
/ #
```

- Make copy of the `busybox` binary in the Flash-based root file system and reboot:

```
/ # cp /bin/busybox /
/ # reboot
```

- After reboot, make sure that the original file and the copy are identical:

```
/ # md5sum /busybox /bin/busybox
9bd57b5c430a1bb69e798b8eaa591675  /busybox
9bd57b5c430a1bb69e798b8eaa591675  /bin/busybox
/ #
```

4.4.8. Test Plan: Install Linux Images (UBIFS variant) to QSPI Flash

The following step-wise test procedure will be used:

1. Boot U-Boot from QSPI Flash.
2. Reset the environment:

```
=> env default -f -a
=> saveenv
=>
```

3. Set IP addresses as per actual networking settings:

```
=> setenv ipaddr 192.168.1.117
=> setenv serverip 192.168.1.73
=> setenv gatewayip 192.168.1.1
=> saveenv
=>
```

4. Set the `UBI` file system type and save the environment:

```
=> setenv fstype ubi
=> savee
Saving Environment to SPIFlash... Erasing SPI flash...Writing to SPI flash...done
Valid environment: 2
OK
=>
```

5. Install the software components:

```
=> sf probe 0
SF: Detected is25wp128 with page size 256 Bytes, erase size 4 KiB, total 16 MiB
=> run sf kernel update
Using ethernet@40424000 device
TFTP from server 192.168.1.73; our IP address is 192.168.1.117
```

```
Filename 'imxrt1170/rootfs_flash.uImage'.
Load address: 0x80007fc0
Loading: ##################################################################
         ##################################################################
         ##################################################################
         ##################################################################
         ##################################################################
         ##################################################################
         ##################################################################
         ##################################################################
         ##################################################################
         ##################################################################
         ##################################################################
         ##################################################################
         ##################################################################
         ##################################################################
         ##############################
         2.9 MiB/s
done
Bytes transferred = 5467251 (536c73 hex)
SF: 8388608 bytes @ 0x80000 Erased: OK
device 0 offset 0x80000, size 0x536c73
SF: 5467251 bytes @ 0x80000 Written: OK
=> run sf rootfs update
Using ethernet@40424000 device
TFTP from server 192.168.1.73; our IP address is 192.168.1.117
Filename 'imxrt1170/rootfs_flash.ubi'.
Load address: 0x80007fc0
Loading: ##################################################################
         ##################################################################
         ##################################################################
         ##################################################################
         ##################################################################
         2.9 MiB/s
done
Bytes transferred = 1638400 (190000 hex)
SF: 7864320 bytes @ 0x880000 Erased: OK
device 0 offset 0x880000, size 0x190000
SF: 1638400 bytes @ 0x880000 Written: OK
=>
```

4.4.9. Test Plan: Linux Boot from QSPI Flash (UBIFS variant)

The following step-wise test procedure will be used:

⚠This item requires the UBIFS image installed to the QSPI Flash as per Section: "Test Plan: Install Linux Images (UBIFS variant) to QSPI Flash" and the `fstype` variable is set as `ubi` in the U-Boot environment.

- Reset the board and make sure it automatically boots up to `busybox`:

```
=> reset
...
This architecture does not have kernel memory protection.
Run /sbin/init as init process
/ #
```

4.4.10. Test Plan: Linux Device Driver for QSPI Flash and Flash File System (UBIFS variant)

The following step-wise test procedure will be used:

⚠This item requres the UBIFS image installed to the QSPI Flash as per Section: "Test Plan: Install Linux Images (UBIFS variant) to QSPI Flash" and the `fstype` variable is set as `ubi` in the U-Boot environment.

- Boot from the QSPI Flash up the `busybox` shell.

- Make sure that the rootfs partition is mounted as the Linux root file system:

```
/ # mount
ubi0:rootfs on / type ubifs (rw,relatime,assert=read-only,ubi=0,vol=0)
devtmpfs on /dev type devtmpfs (rw,relatime)
proc on /proc type proc (rw,relatime)
sysfs on /sys type sysfs (rw,relatime)
devpts on /dev/pts type devpts (rw,relatime,gid=5,mode=620,ptmxmode=000)
/ #
```

- Make copy of the `busybox` binary in the Flash-based root file system and reboot:

```
/ # cp /bin/busybox /
/ # reboot
```

- After reboot, make sure that the original file and the copy are identical:

```
/ # md5sum /busybox /bin/busybox
9bd57b5c430a1bb69e798b8eaa591675  /busybox
9bd57b5c430a1bb69e798b8eaa591675  /bin/busybox
/ #
```