Support Booting from QSPI Flash in Linux i.MX RT1050/RT1060/RT1170

Detailed Requirements and Design

qspi-drad-1_1.doc

Revision:	*1.1*
Date:	*5/19/2023*

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 RT1050/RT1060/RT1170 BSPs.

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 QSPI 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 images to QSPI Flash from SD Card from the U-Boot command line interface.
    - *Rationale*: Explicit customer requirement.
      Implementation: Section: "Design: U-Boot Install Images to QSPI Flash".
      Test: Section: "Test Plan: U-Boot Install Images to QSPI Flash".

5. Support Linux boot from QSPI Flash, in the following configuration:
    - kernel image loaded from QSPI Flash to RAM for execution;
    - `dtb` image loaded from QSPI Flash into RAM for execution;
    - root file system mounted in QSPI Flash as the read-write Flash file system (UBIFS).

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

6. Support QSPI Flash in Linux, using a newly developed device driver following the structure of the following existing device driver: `fsl-quadspi.c` (`{BSP-ROOT}/linux/drivers/mtd/spi-nor/fsl-quadspi.c`). 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".

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 i.MX RT1050/RT1060/RT1170 EVKB 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 `<board_name>-sfboot_defconfig` will be added to U-Boot to support boot from QSPI Flash on the NXP i.MX RT1050/RT1060. The configuration file `mxrt117x-evk_defconfig` will be used in U-Boot to support boot from QSPI Flash on the NXP i.MX RT1170. The standard build procedure will be used to generate the image `u-boot.flexspi` bootable from QSPI Flash:

```
make mxrt1050-evk-sfboot_config
```

or:

```
make mxrt1060-evk-sfboot_config
```

or:

```
make mxrt117x-evk_defconfig
```

then build u-boot:

```
make
```

The `u-boot.flexspi` image must be programmed to the QPSI flash on the NXP i.MX RT1050/1060 EVKB board at offset 0. The `u-boot.flexspi` image must be programmed to the QPSI flash on the NXP i.MX RT1170 EVKB board at offset 0x400. The image consists of the U-Boot itself and two i.MX RT1050-specific headers:

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

The Image Vector Table is generated by `mkimage` using the `board/freescale/<board_name>/imximage.cfg` configuration file.

The The FlexSPI Configuration Block is compiled from the `board/freescale/<board_name>/flexspi_cb.c` file. This file contains a definition of the `flexspi_nor_config_t` structure, as per the corresponding Processor Reference Manual.

`board_name` is as follows:

- mxrt105x-evk - for the i.MX RT1050 board
- mxrt106x-evk - for the i.MX RT1060 board
- mxrt117x-evk - for the i.MX RT1170 board

The default configuration will be set up to support the ISSI QSPI Flash installed on the EVKB 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 `0x50000` - `0x70000` in QSPI flash.

3.4. Design: U-Boot Install Images to QSPI Flash

The QSPI Flash device will be logically divided into 5 sections to store the software components of the system:

- `0x000000 - 0x050000` - U-Boot
- `0x050000 - 0x070000` - U-Boot Environment
- `0x070000 - 0x080000` - DTB Image
- `0x080000 - 0x480000` - Kernel Image
- `0x480000 - 0x800000` - Root File System

The following commands will be defined in the U-Boot environment to update the above components:

- `sf_uboot_update` - Update the U-Boot section
- `sf_dtb_update` - Update the DTB section
- `sf_kernel_update` - Update the Kernel section
- `sf_rootfs_update` - Update the RootFS section

For the NXP i.MX RT1050/RT1060 BSPs all the `sf_*_update` commands will read images from the FAT FS partition on SD-card and install them to the corresponding section in QSPI Flash. The names of the images are defined by the following U-Boot environment variables:

- `uboot` - File name for the U-Boot image, default is `u-boot.flexspi`
- `dtb` - File name for the DTB image, default is `rootfs_ubi.dtb`
- `image` - File name for the Kernel image, default is `rootfs_ubi.uImage`
- `rootfs` - File name for the RootFS image, default is `rootfs.ubi`

For the NXP i.MX RT1170 BSP all the `sf_*_update` commands will download images via tftp and install them to the corresponding section in QSPI Flash. The names of the images are defined by the following U-Boot environment variables:

- `uboot` - File name for the U-Boot image, default is `u-boot.flexspi`
- `dtb` - File name for the DTB image, default is `rootfs_ubi.dtb`
- `image_ubi` - File name for the Kernel image, default is `rootfs_ubi.uImage`
- `rootfs` - File name for the RootFS image, default is `rootfs.ubi`

The `sf_*_update` commands will be available in the `mxrt1050-evk-sfboot` or `mxrt1060-evk-sfboot` or `mxrt117x-evk_defconfig` configuration when booting from QSPI Flash.

If there is no U-Boot installed in QSPI Flash yet on the i.MX RT1050/RT1060 board, the user can use the regular `mxrt1050-evk` or `mxrt1060-evk` configuration to boot from SD-card and make the first installation of U-Boot to the QSPI Flash. The following is the instruction on how to build the U-Boot image to boot from SD-card with write support to QSPI flash:

1. Switch to the regular `mxrt1050-evk` or `mxrt1060-evk` configuration.
2. Run `make menuconfig` to enable the `FSL_FLEXSPI`, `SPI_FLASH`, `SPI_FLASH_ISSI` and `CMD_SF` configuration options.
3. Build the `u-boot-dtb.imx` image and install it to the SD-card.

⚠Note that only the `sf erase` and `sf write` operation supported when booting from SD-card. The `sf read` command is available bu can return wrong data, when booting from SD card. This boot mode is used only for the first installation of U-Boot to QSPI Flash. `sf read` is fully supported when booting from the QSPI Flash.

If there is no U-Boot installed in QSPI Flash yet on the i.MX RT1170 board, the user can follow the instruction described in the MIMXRT1170 EVK Board Hardware User's document.

3.5. Design: Linux Boot from QSPI Flash

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

- Support for 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 an UBIFS file system in QSPI Flash.
- The kernel and the DTB images will be built separately, outside of the `mkimage` 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_ubi` project:

- `RFS_BUILD_DIR` - temporary directory to build the root file system image
- `UBI_IMAGES` - tells the make to build the file system image
- `MKFSUBIFS_FLAGS` - Flash-specific flags for the `mkfs.ubifs` utility
- `UBINIZE_FLAGS` - Flash-specific flags for the `ubinize` utility
- `SEPARATE_DTB` - tells the make not to build the multi-part image but save the DTB separately.

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

An `fsl_flexspi.c` driver will be added to the Linux device drivers.

The driver will provide an API compatible with the standard `SPI-NOR` framework in the kernel similar to the existing `fsl_qspi.c` driver. The QSPI Flash will be available in Linux as a standard `MTD` device.

Support for Linux Flash file systems does not require changes to the kernel.

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 RT1050 BSP:

- https://www.emcraft.com/imxrt1050/rm2595

for the i.MX RT1060 BSP:

- https://www.emcraft.com/imxrt1060/rm2861

for the i.MX RT1170 BSP:

- https://www.emcraft.com/imxrt1170/rm6629

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:

- `u-boot.flexspi` - U-Boot image installable to QSPI Flash.
- `u-boot-dtb.imx` - U-Boot image installable to SD-card with support for `sf_*` commands; this image allows booting from SD Card and installing U-Boot to QSPI Flash. Only for the i.MX RT1050/RT1060 BSPs.
- `rootfs_ubi.dtb` - Linux device tree.
- `rootfs_ubi.uImage` - Linux kernel.
- `rootfs.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/`.

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 i.MX RT1050 EVKB board or i.MX RT1060 EVKB board or i.MX RT1170 EVKB board with serial console connected to PC.

- The following hardware reworks are required on the NXP i.MX RT1050 EVKB board to enable the QSPI Flash, as described in section 2.7 of the NXP IMXRT1050 EVKB Board Hardware User's Guide:
    - Remove resistors: R356, R361 - R366
    - Weld 0Ω resistors: R153 - R158.
- SD-card as the media to transfer the images from the development host to the target board (only for the i.MX RT1050/RT1060 boards).
- The NXP i.MX RT1170 EVK Board is connected to LAN using the J3 100M ENET connector.

4.3.2. Software Set-Up

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 bootable SD-card image with support for the `sf_*` commands for the first U-Boot installation to QSPI Flash (only for the IMXRT1050/IMXRT1060 EVK boards):
    1. Enable the default configuration
        - for the IMXRT1050 EVK board

```
$ make distclean
$ make mxrt105x-evk_config
```

        - for the IMXRT1060 EVK board

```
$ make distclean
$ make mxrt106x-evk_config
```

    2. Run `menuconfig` and enable the `FSL_FLEXSPI`, `SPI_FLASH`, `SPI_FLASH_ISSI` and `CMD_SF` configuration options:

```
$ make menuconfig
Symbol: FSL_FLEXSPI [=y]
Prompt: Freescale Flex SPI controller
  Location:
    -> Device Drivers
(1)   -> SPI Support
Symbol: SPI_FLASH [=y]
Type  : boolean
Prompt: Legacy SPI Flash Interface support
  Location:
    -> Device Drivers
       -> SPI Flash Support
Symbol: SPI_FLASH_ISSI [=y]
Type  : boolean
Prompt: ISSI SPI flash support
  Location:
    -> Device Drivers
      -> SPI Flash Support
         -> Legacy SPI Flash Interface support (SPI_FLASH [=y])

Symbol: CMD_SF [=y]
Type  : boolean
Prompt: sf
  Location:
    -> Command line interface
      -> Device access commands
```

    3. Build U-Boot:

```
$ make
```

4. Install the resultant image to the connected SD-card:

```
$ sudo dd if=u-boot-dtb.imx of=/dev/sdX bs=1k seek=1
$ sync
```

3. Build the U-Boot image bootable from QSPI Flash
    1. Configure and build U-Boot for QSPI Flash
        - for the IMXRT1050 EVK board:

```
$ make distclean
$ make mxrt105x-evk-sfboot_config
$ make
```

        - for the IMXRT1060 EVK board:

```
$ make distclean
$ make mxrt106x-evk-sfboot_config
$ make
```

        - for the IMXRT1170 EVK board:

```
$ make distclean
$ make mxrt117x-evk_defconfig
$ make
```

    2. Copy the resultant image to the FATFS partition on the SD-card for the IMXRT1050/IMXRT1060 EVK boards:

```
$ sudo mount /dev/sdX1 ~/tmp/
$ sudo cp u-boot.flexspi ~/tmp/
$ sudo umount ~/tmp/
```

or copy the resultant image to a TFTP server in the same network as the target board is connected to for the IMXRT1170 EVK board.

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_ubi` project:

```
$ cd rootfs_ubi
$ make
```

3. Copy the resultant images to the FATFS partition on the SD-card for the IMXRT1050/IMXRT1060 EVK boards:

```
$ sudo mount /dev/sdX1 ~/tmp/
$ sudo cp rootfs_ubi.uImage rootfs_ubi.dtb rootfs.ubi ~/tmp/
$ sudo umount ~/tmp/
```

or copy the resultant images (`rootfs_ubi.uImage`, `rootfs_ubi.dtb`, rootfs.ubi) to a TFTP server in the same network as the target board is connected to for the IMXRT1170 EVK board.

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. IMXRT1050/IMXRT1060 EVK boards:
    - Set-up the `SW7` switch on the target board to boot from SD-card (`SW7/1` = `ON`, `SW7/2` = `OFF`, `SW7/3` = `ON`, `SW7/4` = `OFF`)
    - Insert the SD-card prepared in Section: "Software Set-Up" and power on the board.
    - Stop U-Boot at the command monitor.
    - Run the following commands to install U-Boot to QSPI Flash:

```
=> sf probe 0
=> sf erase 0 0x50000
=> fatload mmc 0 ${loadaddr} u-boot.flexspi
=> sf write ${loadaddr} 0 ${filesize}
```

    - Power off the target board.
    - Set-up the `SW7` switch on the target board to boot from QSPI Flash (`SW7/1` = `OFF`, `SW7/2` = `OFF`, `SW7/3` = `ON`, `SW7/4` = `OFF`)
    - Remove the SD-card and power on the board.
    - Validate that U-Boot has successfully booted from QSPI Flash.
3. IMXRT1170 EVK board:
    - Set-up the `SW7` switch on the target board to boot from QSPI Flash (`SW7/1` = `OFF`, `SW7/2` = `OFF`, `SW7/3` = `ON`, `SW7/4` = `OFF`)
    - Power on the target board.
    - Stop U-Boot at the command monitor.
    - Run the following commands to install U-Boot to QSPI Flash:

```
=> run sf_uboot_update
Using ethernet@40424000 device
TFTP from server 192.168.1.79; our IP address is 192.168.1.15
Filename 'imxrt117x/u-boot.flexspi'.
Load address: 0x80007fc0
Loading: ##################
         332 KiB/s
done
Bytes transferred = 258048 (3f000 hex)
SF: 327680 bytes @ 0x0 Erased: OK
device 0 offset 0x0, size 0x3f000
SF: 258048 bytes @ 0x0 Written: OK
=> env default -a -f
## Resetting to default environment
=> setenv ipaddr 192.168.1.15
=> setenv serverip 192.168.1.79
=> saveenv
Saving Environment to SPI Flash...
SF: Detected is25wp128a with page size 256 Bytes, erase size 64 KiB, total 16 MiB
Erasing SPI flash...Writing to SPI flash...done
Valid environment: 1
```

    - Validate that U-Boot has successfully booted from QSPI Flash:

```
=> reset
resetting ...
U-Boot 2017.09-rc1 (May 14 2023 - 20:14:46 +0300)
```

```
CPU: i.MX RT117x at 996MHz
Model: NXP i.MXRT1174 EVK
DRAM: 64 MiB
MMC: FSL_SDHC: 0
SF: Detected is25wp128a with page size 256 Bytes, erase size 64 KiB, total 16 MiB
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 is25wp064a with page size 256 Bytes, erase size 64 KiB, total 8 MiB
=>
```

3. Read the U-Boot partition to RAM:

```
=> sf read ${loadaddr} 0 0x50000
device 0 offset 0x0, size 0x50000
SF: 327680 bytes @ 0x0 Read: OK
=>
```

4. Make sure the FlexSPI Configuration Block is at the beginning of the read data: the first 4 symbols must be "FCFB" for the IMXRT1050/IMXRT1060 EVK boards:

```
=> md ${loadaddr} 1
80007fc0: 42464346                               FCFB
=>
```

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

```
=> md 800083C0 1
800083c0: 42464346 FCFB
```

6. Erase 5 sectors in the middle of QSPI Flash:

```
=> sf erase 0x300000 0x50000
SF: 327680 bytes @ 0x300000 Erased: OK
=>
```

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

```
=> sf write ${loadaddr} 0x300000 0x50000
device 0 offset 0x300000, size 0x50000
SF: 327680 bytes @ 0x300000 Written: OK
=>
```

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

```
=> sf read 0x81000000 0x300000 0x50000
device 0 offset 0x300000, size 0x50000
SF: 327680 bytes @ 0x300000 Read: OK
=>
```

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

```
=> cmp.b ${loadaddr} 0x81000000 0x50000
```

```
Total of 327680 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. Remove SD-card and boot U-Boot from QSPI Flash.
2. Define and save a test variable:

```
=> setenv testvar testval
=> saveenv
Saving Environment to SPI Flash...
SF: Detected is25wp064a with page size 256 Bytes, erase size 64 KiB, total 8 MiB
Erasing SPI flash...Writing to SPI flash...done
Valid environment: 2
=>
```

3. Reset the board:

```
=> reset
```

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

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

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

The following step-wise test procedure will be used:

1. Boot U-Boot from QSPI Flash.
2. Insert the SD-card prepared in Section: "Software Set-Up" to the SD Card holder (for the IMXRT1050/IMXRT1060 EVK boards).
3. Reset the environment:

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

4. Install the software components

   for the IMXRT1050/IMXRT1060 EVK boards:

```
=> sf probe 0
SF: Detected is25wp064a with page size 256 Bytes, erase size 64 KiB, total 8 MiB
=> run sf_uboot_update
reading u-boot.flexspi
256000 bytes read in 78 ms (3.1 MiB/s)
SF: 327680 bytes @ 0x0 Erased: OK
device 0 offset 0x0, size 0x3e800
SF: 256000 bytes @ 0x0 Written: OK
=> run sf_dtb_update
reading rootfs_ubi.dtb
10438 bytes read in 25 ms (407.2 KiB/s)
SF: 65536 bytes @ 0x70000 Erased: OK
device 0 offset 0x70000, size 0x28c6
SF: 10438 bytes @ 0x70000 Written: OK
=> run sf_kernel_update
reading rootfs_ubi.uImage
2957824 bytes read in 693 ms (4.1 MiB/s)
SF: 4194304 bytes @ 0x80000 Erased: OK
```

```
device 0 offset 0x80000, size 0x2d2200
SF: 2957824 bytes @ 0x80000 Written: OK
=> run sf_rootfs_update
reading rootfs.ubi
1310720 bytes read in 320 ms (3.9 MiB/s)
SF: 3670016 bytes @ 0x480000 Erased: OK
device 0 offset 0x480000, size 0x140000
SF: 1310720 bytes @ 0x480000 Written: OK
=>
```

for the IMXRT1170 EVK board:

```
=> sf probe 0
SF: Detected is25wp128a with page size 256 Bytes, erase size 64 KiB, total 16 MiB
=> run sf_dtb_update
Using ethernet@40424000 device
TFTP from server 192.168.1.79; our IP address is 192.168.1.15
Filename 'imxrt117x/rootfs_ubi.dtb'.
Load address: 0x80007fc0
Loading: #
         98.6 KiB/s
done
Bytes transferred = 7896 (1ed8 hex)
SF: 65536 bytes @ 0x70000 Erased: OK
device 0 offset 0x70000, size 0x1ed8
SF: 7896 bytes @ 0x70000 Written: OK
=> run sf_kernel_update
Using ethernet@40424000 device
TFTP from server 192.168.1.79; our IP address is 192.168.1.15
Filename 'imxrt117x/rootfs_ubi.uImage'.
Load address: 0x80007fc0
Loading: #################################################################
         #################################################################
         #################################################################
         ###############################################
         237.3 KiB/s
done
Bytes transferred = 3550944 (362ee0 hex)
SF: 10485760 bytes @ 0x80000 Erased: OK
device 0 offset 0x80000, size 0x362ee0
SF: 3550944 bytes @ 0x80000 Written: OK
=> run sf_rootfs_update
Using ethernet@40424000 device
TFTP from server 192.168.1.79; our IP address is 192.168.1.15
Filename 'imxrt117x/rootfs.ubi'.
Load address: 0x80007fc0
Loading: #################################################################
         #########################
         346.7 KiB/s
done
Bytes transferred = 1310720 (140000 hex)
SF: 3670016 bytes @ 0x480000 Erased: OK
device 0 offset 0x480000, size 0x140000
SF: 1310720 bytes @ 0x480000 Written: OK
```

4.4.5. Test Plan: Linux Boot from QSPI Flash

The following step-wise test procedure will be used:

1. Remove the SD Card from the SD holder. Reset the board and make sure it automatically boots up to `busybox`:

```
=> reset
...
init started: BusyBox v1.24.2 (2018-06-18 18:30:51 MSK)
/ #
```

4.4.6. Test Plan: Linux Device Driver for QSPI Flash and Flash File System

The following step-wise test procedure will be used:

1. Boot from QSPI Flash up `busybox`.
2. Make sure that the UBIFS partition is mounted as the Linux root file system:

```
/ # mount
ubi0:rootfs on / type ubifs (rw,relatime)
devtmpfs on /dev type devtmpfs (rw,relatime,mode=0755)
proc on /proc type proc (rw,relatime)
sysfs on /sys type sysfs (rw,relatime)
/ #
```

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

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

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

```
/ # md5sum busybox
ea6e64963727e1c6bb26946070795644 busybox
/ # md5sum /bin/busybox
ea6e64963727e1c6bb26946070795644 /bin/busybox
/ #
```