Support QSPI Flash on STM32H753I-EVAL

Detailed Requirements and Design

rm6774-drad-1_0.doc

RM:	*6774*
Revision:	*1.0*
Date:	*9/21/2023*

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 STM32H753I-EVAL BSP.

2. Requirements

2.1. Detailed Requirements

The following are the requirements for this project:

1. Create separate configurations for U-Boot and Linux Kernel to support booting from QSPI Flash.
   - *Rationale*: Explicit customer requirement.
     Implementation: Section: "Design: Configurations for U-Boot and Linux Kernel".
     Test: Section: "Test Plan: Create separate configurations for U-Boot and Linux Kernel".

2. Support QSPI Flash in Linux (single Flash bank only).
   - *Rationale*: Explicit customer requirement.
     Implementation: Section: "Design: Support QSPI Flash in Linux".
     Test: Section: "Test Plan: Support QSPI Flash in Linux".

3. Support QSPI Flash in U-Boot (single Flash bank only).
   - *Rationale*: Explicit customer requirement.
     Implementation: Section: "Design: Support QSPI Flash in U-Boot".
     Test: Section: "Test Plan: Support QSPI Flash in U-Boot".

4. 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".

5. 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".

6. 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".

7. Support Linux boot from QSPI Flash, in the following configuration:
   - Initial U-boot boots from the on-chip Flash,
   - Kernel and dtb image loaded from QSPI Flash to 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".

8. Partition QSPI Flash as follows:
   - MTD Part 1: `0x0000000` - `0x0010000` - U-Boot Environment

- MTD Part 2: `0x0010000` - `0x0020000` - DTB Image
- MTD Part 3: `0x0020000` - `0x0820000` - Kernel Image
- MTD Part 4: `0x0820000` - `0x1020000` - Root File System
- MTD Part 5: `0x1020000` - `0x4000000` - User Data
- *Rationale*: Explicit customer requirement.
  Implementation: Section: "Design: Partition QSPI Flash ".
  Test: Section: "Test Plan: Partition QSPI Flash".

9. Support fw_setenv/fw_printenv commands in Linux to manage the U-Boot environment.
   - *Rationale*: Explicit customer requirement.
     Implementation: Section: "Design: Support fw_printenv/setenv commands in Linux".
     Test: Section: "Test Plan: Support fw_printenv/setenv commands in Linux".

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 STM32H753I-EVAL board is not required.
   - *Rationale*: Costs reduction measure.

3. Design

3.1. Design: Configurations for U-Boot and Linux Kernel

Dedicated configurations will be created for U-Boot and Linux:

- `stm32h753-eval_sfboot_defconfig` - U-Boot configuration.
- `rootfs_ubi.kernel.STM32H753IEVAL` and `rootfs_ubi.dts.STM32H753IEVAL` - Linux Kernel configuration file and Device Tree.

3.2. Design: Support QSPI Flash in Linux

The existing STM32 QSPI driver `linux/drivers/spi-stm32-qspi.c` will be enabled, and the respective changes will be added to the kernel `.dts` file.

The QSPI Flash functionality will be enabled in the Linux kernel configuration as follows:

- Go to `Device Drivers`
- Enable `Memory Technology Device (MTD) support` (`CONFIG_MTD`)
- Go to `Device Drivers` -> `Memory Technology Device (MTD) support`
- Enable `SPI NOR device support` (`CONFIG_MTD_SPI_NOR`)

This Add-on will provide several modifications for the `linux/drivers/spi-stm32-qspi.c` and `linux/drivers/mtd/spi-nor/micron-st.c` drivers:

- For `linux/drivers/mtd/spi-nor/micron-st.c` - add a description for MT25TL01G Flash and redefine locking/unlocking functions to prevent locking of the Flash regions.
  The default `spi_nor_sr_lock()` sets the `SR_SRWD` bit after configuring `BP` bits.
  In ST and Micron flashes `SR_SRWD` bit is protected when `IO2/W#` is driven low and the default `spi_nor_sr_unlock()` is unable to unlock locked regions.

- For `linux/drivers/spi-stm32-qspi.c` - implement a workaround for the STM32H7 QSPI limitation. For details see "2.9.4 QUADSPI internal timing criticality" in STM32H742x/743xI/G, STM32H750xB and STM32H753xI device errata.

3.3. Design: Support QSPI Flash in U-Boot

The STM32 QSPI driver `u-boot/drivers/stm32_qspi.c` will be enabled, and the respective changes will be added to the`stm32h753i-eval.dts` file.

The QSPI Flash functionality will be enabled in the Linux kernel configuration as follows:

- Go to `Device Drivers -> MTD Support`
- Enable `Enable Driver Model for MTD drivers` (`CONFIG_MTD`)
- Go to `Device Drivers -> MTD Support -> SPI Flash Support`
- Enable `Legacy SPI Flash Interface support` (`CONFIG_SPI_FLASH`) and `STMicro SPI flash support` (`CONFIG_SPI_FLASH_STMICRO`)

The default configuration will be set up to support the MT25TL01G QSPI Flash installed on the STM32H753-EVAL board.

3.4. 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 will be placed at the address range `0x00000` - `0x10000` in QSPI Flash.

3.5. 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.

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

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

- `sf_dtb_update` - Update the DTB section, launches `sf_dtb_update_tftp` by default
- `sf_kernel_update` - Update the Kernel section, launches `sf_kernel_update_tftp` by default
- `sf_rootfs_update` - Update the RootFS section, launches `sf_rootfs_update_tftp` by default
- `sf_dtb_update_tftp` - Update the DTB section via TFTP
- `sf_kernel_update_tftp` - Update the Kernel section via TFTP
- `sf_rootfs_update_tftp` - Update the RootFS section via TFTP
- `sf_dtb_update_mmc` - Update the DTB section form SD Card
- `sf_kernel_update_mmc` - Update the Kernel section form SD Card
- `sf_rootfs_update_mmc` - Update the RootFS section form SD Card

For the STM32H753-EVAL 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:

- `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`

3.7. Design: Linux Boot from QSPI Flash

The separate configuration for STM32H753I-EVAL will be created in `projects/rootfs_ubi` to demonstrate booting Linux from QSPI flash. The following main feature will be enabled in the new project:

- Support for QSPI Flash and UBIFS 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, `rootfs_ubi/Makefile` will be modified the following way:

- `MKFSUBIFS_FLAGS` - Flash-specific flags for the `mkfs.ubifs` utility. Will be modified to match QSPI Flash installed on the STM32H753I-EVAL board.
- `CUSTOM_APPS` - a list of custom applications. Tells `make` to build uboot-envtools.

3.8. Design: Partition QSPI Flash

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

- MTD Part 1: `0x0000000` - `0x0010000` - U-Boot Environment
- MTD Part 2: `0x0010000` - `0x0020000` - DTB Image
- MTD Part 3: `0x0020000` - `0x0820000` - Kernel Image
- MTD Part 4: `0x0820000` - `0x1020000` - Root File System
- MTD Part 5: `0x1020000` - `0x4000000` - User Data

The partition table will be created by adding corresponding partition nodes to the QSPI Flash node in the kernel `.dts` file.

3.9. Design: Support fw_printenv/setenv commands in Linux

The U-Boot `fw-utils` and correspinding configuration file `fw_env.config` will be added to the rootfs_ubi configuration to support `fw_printenv` and `fw_setenv` commands in Linux.

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 STM32H7-EVAL BSP:

- https://www.emcraft.com/stm32h7addon/stm32h7-eval/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:

- `u-boot.bin` - U-Boot image with support for QSPI and `sf_*` commands.
- `rootfs_ubi.dtb` - Linux device tree.
- `rootfs_ubi.uImage` - Linux kernel.

- `rootfs.ubi` - UBIFS image with Linux rootfs.
- `u-boot.patch` - Source code patch for U-Boot.
- `linux.patch` - Source code patch for Linux.
- `projects.patch` - Source code patch for `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 STM32H753I-EVAL board with serial console connected to PC.
- SD-card as the media to transfer the images from the development host to the target board.

4.3.2. Software Set-Up

1. Install the BSP, as per the respective "Installing and activating cross development environment" document in the "uClinux" section on the Emcraft site.
2. Download the files listed in Section: "Downloadable Files" to the top of the Linux STM32H7 installation.
3. From the top of the Linux installation, activate the Linux cross-compile environment by running:

```
$ . ./ACTIVATE.sh
```

U-Boot Build:

1. Apply the U-Boot patch from the top of the fresh Linux installation:

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

2. Build U-Boot:

```
$ make stm32h753-eval_sfboot_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_ubi` project:

```
$ cd ./rootfs ubi
$ make clean
$ make
```

3. Copy the resulting image to a TFTP server in the same network as the target board is connected, or copy the images to the FATFS partition on the SD-card:

```
$ mkdir ~/tmp
$ sudo mount /dev/sdX1 ~/tmp/
$ sudo cp rootfs_ubi.uImage rootfs_ubi.dtb rootfs.ubi ~/tmp/
$ sudo umount ~/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: Create separate configurations for U-Boot and Linux Kernel

The following step-wise test procedure will be used:

1. From the top of the Linux installation, go to the `u-boot` subdirectory:

```
$ cd ./u-boot
```

2. Verify that subdirectory `configs` contains the `stm32h753-eval_sfboot_defconfig` configuration file:

```
$ ls -l ./configs/stm32h753-eval sfboot defconfig
-rw-rw-r-- 1 user user 2190 Sep 12 21:50 ./configs/stm32h753-eval_sfboot_defconfig
```

3. From the top of the Linux installation, go to the `projects` subdirectory:

```
$ cd ./projects
```

4. Verify that subdirectory `rootfs_ubi` contains the `rootfs_ubi.kernel.STM32H753IEVAL` configuration file and the Device Tree:

```
$ ls -l ./rootfs ubi/rootfs ubi.kernel.STM32H753IEVAL
./rootfs ubi/rootfs ubi.dts.STM32H753IEVAL
-rw-rw-r-- 1 user user  3668 Sep 13 02:15 ./rootfs_ubi/rootfs_ubi.dts.STM32H753IEVAL
-rw-rw-r-- 1 user user 97565 Sep 13 02:15 ./rootfs_ubi/rootfs_ubi.kernel.STM32H753IEVAL
```

4.4.2. Test Plan: Support QSPI Flash in U-Boot

The following step-wise test procedure will be used:

1. Install U-Boot as per https://emcraft.com/stm32h7-evk-board/running-linux-on-stm32h753i-eval-in-5-minutes.
2. Power cycle the board. While U-Boot is coming up, press any key on the serial console to enter the U-Boot command line interface:

```
U-Boot 2019.04--g3b2c91d23d-dirty (Sep 12 2023 - 19:07:05 +0400)
Model: STMicroelectronics STM32H753i-EVAL board
DRAM:  32 MiB
Flash: 2 MiB
MMC:   STM32 SDMMC2: 0
Loading Environment from SPI Flash... SF: Detected mt25tl01g with page size 256 Bytes,
erase size 64 KiB, total 64 MiB
*** Warning - bad CRC, using default environment
In:    serial@40011000
Out:   serial@40011000
Err:   serial@40011000
Net:
Error: ethernet@40028000 address not set.
eth-1: ethernet@40028000
Hit any key to stop autoboot:  0
STM32H7-EVAL U-Boot >
```

3. Probe the QSPI Flash. Make sure that the correct Flash info is printed out to the console:

```
STM32H7-EVAL U-Boot > sf probe
SF: Detected mt25tl01g with page size 256 Bytes, erase size 64 KiB, total 64 MiB
STM32H7-EVAL U-Boot >
```

4. Execute commad `mtd list` and verify that is prints out the detected QSPI Flash parameters:

```
STM32H7-EVAL U-Boot > mtd list
List of MTD devices:
* nor2
  - type: NOR flash
  - block size: 0x10000 bytes
```

```
  - min I/O: 0x1 bytes
  - 0x000000000000-0x000004000000 : "nor2"
STM32H7-EVAL U-Boot >
```

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

The following step-wise test procedure will be used:

1. Power cycle the board. While U-Boot is coming up, press any key on the serial console to enter the U-Boot command line interface.
2. Define and save a test variable:

```
STM32H7-EVAL U-Boot > setenv testvar testval
STM32H7-EVAL U-Boot > saveenv
Saving Environment to SPI Flash... SF: Detected mt25tl01g with page size 256 Bytes, erase
size 64 KiB, total 64 MiB
Erasing SPI flash...Writing to SPI flash...done
OK
STM32H7-EVAL U-Boot >
```

3. Reset the board:

```
STM32H7-EVAL U-Boot > reset
resetting ...
```

4. Verify that the test variable exists and has the correct value:

```
STM32H7-EVAL U-Boot > print testvar
testvar=testval
STM32H7-EVAL U-Boot >
```

4.4.4. Test Plan: U-Boot sf_ Commands

The following step-wise test procedure will be used:

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

```
STM32H7-EVAL U-Boot > sf probe 0
SF: Detected mt25tl01g with page size 256 Bytes, erase size 64 KiB, total 64 MiB
STM32H7-EVAL U-Boot >
```

3. Read the U-Boot Environment partition to RAM:

```
STM32H7-EVAL U-Boot > sf read 0xD0C00000 0 0x10000
device 0 offset 0x0, size 0x10000
SF: 65536 bytes @ 0x0 Read: OK
STM32H7-EVAL U-Boot >
```

4. Verify that the `md` command displays contents of the U-Boot Environment partition loaded to RAM:

```
STM32H7-EVAL U-Boot > md 0xD0C00000 0x10000
d0c00000: 731c42ef 69646461 65733d70 766e6574    .B.saddip=setenv
d0c00010: 6f6f6220 67726174 7b242073 746f6f62     bootargs ${boot
d0c00020: 73677261 7069207d 697b243d 64646170    args} ip=${ipadd
d0c00030: 243a7d72 7265737b 69726576 243a7d70    r}:${serverip}:$
d0c00040: 7461677b 79617765 3a7d7069 656e7b24    {gatewayip}:${ne
d0c00050: 73616d74 243a7d6b 736f687b 6d616e74    tmask}:${hostnam
d0c00060: 653a7d65 3a306874 0066666f 68637261    e}:eth0:off.arch
d0c00070: 6d72613d 75616200 74617264 31313d65    =arm.baudrate=11
d0c00080: 30303235 616f6200 733d6472 32336d74    5200.board=stm32
d0c00090: 33353768 6176652d 6f62006c 5f647261    h753-eval.board
d0c000a0: 656d616e 6d74733d 37683233 652d3335    name=stm32h753-e
d0c000b0: 006c6176 746f6f62 73677261 6e6f633d    val.bootargs=con
d0c000c0: 656c6f73 7974743d 304d5453 3531312c    sole=ttySTM0,115
d0c000d0: 20303032 6c726165 69727079 206b746e    200 earlyprintk
d0c000e0: 736e6f63 62656c6f 6b6e616c 7020303d    consoleblank=0 p
d0c000f0: 63696e61 6920333d 726f6e67 6f6c5f65    anic=3 ignore_lo
d0c00100: 76656c67 75206c65 6d2e6962 323d6474    glevel ubi.mtd=2
d0c00110: 6f6f7220 74736674 3d657079 66696275     rootfstype=ubif
d0c00120: 6f722073 753d746f 3a306962 746f6f72    s root=ubi0:root
```

```
d0c00130: 72207366 6e692077 2f3d7469 74696e69    fs rw init=/init
...
STM32H7-EVAL U-Boot >
```

5. Erase a sector in the middle of QSPI Flash:

```
STM32H7-EVAL U-Boot > sf erase 0x300000 0x10000
SF: 65536 bytes @ 0x300000 Erased: OK
STM32H7-EVAL U-Boot >
```

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

```
STM32H7-EVAL U-Boot >  sf write 0xD0C00000 0x300000 0x10000
device 0 offset 0x300000, size 0x10000
SF: 65536 bytes @ 0x300000 Written: OK
STM32H7-EVAL U-Boot >
```

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

```
STM32H7-EVAL U-Boot >  sf read 0xD0C30000 0x300000 0x10000
device 0 offset 0x300000, size 0x10000
SF: 65536 bytes @ 0x300000 Read: OK
STM32H7-EVAL U-Boot >
```

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

```
STM32H7-EVAL U-Boot >  cmp.b 0xD0C00000 0xD0C30000 0x10000
Total of 65536 byte(s) were the same
STM32H7-EVAL U-Boot >
```

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

The following step-wise test procedure will be used:

Installing images via TFTP:

1. Boot U-Boot.
2. Reset the environment:

```
STM32H7-EVAL U-Boot > env default -f -a
## Resetting to default environment
STM32H7-EVAL U-Boot >
```

3. Set the ethaddr, ipaddr and serverip U-Boot environment variables:

```
STM32H7-EVAL U-Boot > setenv -f ethaddr 34:56:78:9a:bc:12
STM32H7-EVAL U-Boot > setenv ipaddr 172.17.0.166
STM32H7-EVAL U-Boot > setenv serverip 172.17.0.1
STM32H7-EVAL U-Boot > saveenv
Saving Environment to SPI Flash... Erasing SPI flash...Writing to SPI flash...done
OK
STM32H7-EVAL U-Boot >
```

4. Install the software components:

```
STM32H7-EVAL U-Boot > run sf_kernel_update
SF: Detected mt25tl01g with page size 256 Bytes, erase size 64 KiB, total 64 MiB
ethernet@40028000 Waiting for PHY auto negotiation to complete. done
Using ethernet@40028000 device
TFTP from server 172.17.0.1; our IP address is 172.17.0.166
Filename 'rootfs_ubi.uImage'.
Load address: 0xd0c00000
Loading: ################################################################
         ################################################################
         ################################################################
         ################################################################
         ################################################################
         ############################################################
         821.3 KiB/s
done
Bytes transferred = 5619712 (55c000 hex)
SF: 8388608 bytes @ 0x20000 Erased: OK
device 0 offset 0x20000, size 0x55c000
```

```
SF: 5619712 bytes @ 0x20000 Written: OK
STM32H7-EVAL U-Boot >
STM32H7-EVAL U-Boot > run sf dtb update
SF: Detected mt25tl01g with page size 256 Bytes, erase size 64 KiB, total 64 MiB
Using ethernet@40028000 device
TFTP from server 172.17.0.1; our IP address is 172.17.0.166
Filename 'rootfs_ubi.dtb'.
Load address: 0xd0c00000
Loading: ##
         994.1 KiB/s
done
Bytes transferred = 17309 (439d hex)
SF: 65536 bytes @ 0x10000 Erased: OK
device 0 offset 0x10000, size 0x439d
SF: 17309 bytes @ 0x10000 Written: OK
STM32H7-EVAL U-Boot >
STM32H7-EVAL U-Boot > run sf rootfs update
SF: Detected mt25tl01g with page size 256 Bytes, erase size 64 KiB, total 64 MiB
Using ethernet@40028000 device
TFTP from server 172.17.0.1; our IP address is 172.17.0.166
Filename 'rootfs.ubi'.
Load address: 0xd0c00000
Loading: ################################################################
         ################################################################
         ############################
         3.4 MiB/s
done
Bytes transferred = 2293760 (230000 hex)
SF: 8388608 bytes @ 0x820000 Erased: OK
device 0 offset 0x820000, size 0x230000
SF: 2293760 bytes @ 0x820000 Written: OK
STM32H7-EVAL U-Boot >
```

Installing images from SD Card:

1. Insert the SD-card prepared in Section: "Software Set-Up" to the SD Card holder.
2. Boot U-Boot.
3. Reset the environment:

```
STM32H7-EVAL U-Boot > env default -f -a
## Resetting to default environment
STM32H7-EVAL U-Boot > saveenv
Saving Environment to SPI Flash... Erasing SPI flash...Writing to SPI flash...done
OK
STM32H7-EVAL U-Boot >
```

4. Install the software components:

```
STM32H7-EVAL U-Boot > run sf kernel update mmc
SF: Detected mt25tl01g with page size 256 Bytes, erase size 64 KiB, total 64 MiB
5619712 bytes read in 369 ms (14.5 MiB/s)
SF: 8388608 bytes @ 0x20000 Erased: OK
device 0 offset 0x20000, size 0x55c000
SF: 5619712 bytes @ 0x20000 Written: OK
STM32H7-EVAL U-Boot >
STM32H7-EVAL U-Boot > run sf_dtb_update_mmc
SF: Detected mt25tl01g with page size 256 Bytes, erase size 64 KiB, total 64 MiB
17309 bytes read in 7 ms (2.4 MiB/s)
SF: 65536 bytes @ 0x10000 Erased: OK
device 0 offset 0x10000, size 0x439d
SF: 17309 bytes @ 0x10000 Written: OK
STM32H7-EVAL U-Boot >
STM32H7-EVAL U-Boot > run sf rootfs update mmc
SF: Detected mt25tl01g with page size 256 Bytes, erase size 64 KiB, total 64 MiB
2293760 bytes read in 154 ms (14.2 MiB/s)
SF: 8388608 bytes @ 0x820000 Erased: OK
device 0 offset 0x820000, size 0x230000
SF: 2293760 bytes @ 0x820000 Written: OK
STM32H7-EVAL U-Boot >
```

4.4.6. 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.
2. Power cycle the board and verify that Linux boots automatically from the QSPI Flash:

```
U-Boot 2019.04--g65b71794ba-dirty (Sep 13 2023 - 23:12:09 +0400)
Model: STMicroelectronics STM32H753i-EVAL board
DRAM:  32 MiB
Flash: 2 MiB
MMC:   STM32 SDMMC2: 0
Loading Environment from SPI Flash... SF: Detected mt25tl01g with page size 256 Bytes,
erase size 64 KiB, total 64 MiB
OK
In:    serial@40011000
Out:   serial@40011000
Err:   serial@40011000
Net:   eth0: ethernet@40028000
Hit any key to stop autoboot:  0
SF: Detected mt25tl01g with page size 256 Bytes, erase size 64 KiB, total 64 MiB
device 0 offset 0x20000, size 0x800000
SF: 8388608 bytes @ 0x10000 Read: OK
## Booting kernel from Legacy Image at d0c00000 ...
   Image Name:   Linux-6.1.28-g3c01796871c7
   Image Type:   ARM Linux Multi-File Image (uncompressed)
   Data Size:    5636913 Bytes = 5.4 MiB
   Load Address: d0008000
   Entry Point:  d0008001
   Contents:
      Image 0: 5619648 Bytes = 5.4 MiB
      Image 1: 17253 Bytes = 16.8 KiB
   Verifying Checksum ... OK
## Loading init Ramdisk from multi component Legacy Image at d0c00000 ...
## Flattened Device Tree from multi component Image at D0C00000
   Booting using the fdt at 0xd115c00c
   Loading Multi-File Image ... OK
   Loading Ramdisk to d1a91000, end d1a95365 ... OK
ERROR: reserving fdt memory region failed (addr=d1c00000 size=300000)
ERROR: reserving fdt memory region failed (addr=d1f00000 size=100000)
   Loading Device Tree to d1a89000, end d1a90364 ... OK
Starting kernel ...
...
/ #
```

4.4.7. Test Plan: Support QSPI Flash in Linux

The following step-wise test procedure will be used:

1. Boot Linux from QSPI Flash.
2. Make sure that the UBIFS 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)
tmpfs on /tmp type tmpfs (rw,noatime,mode=1777)
devpts on /dev/pts type devpts (rw,relatime,gid=5,mode=620,ptmxmode=000)
/ #
```

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
1423018e4ef1542fc0d0d366d628ab24  busybox
/ # md5sum /bin/busybox
1423018e4ef1542fc0d0d366d628ab24  /bin/busybox
```

```
/ #
```

4.4.8. Test Plan: Partition QSPI Flash

The following step-wise test procedure will be used:

1. Power cycle the board.
2. Boot Linux from the QSPI Flash.
3. Verify that the partition table is printed in the Linux boot log:

```
[    0.654068] spi-nor spi0.0: mt25tl01g (65536 Kbytes)
[    0.659526] 5 fixed-partitions partitions found on MTD device spi0.0
[    0.664831] Creating 5 MTD partitions on "spi0.0":
[    0.669334] 0x000000000000-0x000000010000 : "uboot env"
[    0.676819] 0x000000010000-0x000000020000 : "dtb"
[    0.682586] 0x000000020000-0x000000820000 : "kernel"
[    0.688217] 0x000000820000-0x000001020000 : "rootfs"
[    0.694116] 0x000001020000-0x000004000000 : "userdata"
```

4. Verify that the `dev` subdirectory contains `mtd*` device files:

```
/ # ls -l /dev/mtd*
crw-------    1 root     root       90,   0 Jan  1 00:00 /dev/mtd0
crw-------    1 root     root       90,   1 Jan  1 00:00 /dev/mtd0ro
crw-------    1 root     root       90,   2 Jan  1 00:00 /dev/mtd1
crw-------    1 root     root       90,   3 Jan  1 00:00 /dev/mtd1ro
crw-------    1 root     root       90,   4 Jan  1 00:00 /dev/mtd2
crw-------    1 root     root       90,   5 Jan  1 00:00 /dev/mtd2ro
crw-------    1 root     root       90,   6 Jan  1 00:00 /dev/mtd3
crw-------    1 root     root       90,   7 Jan  1 00:00 /dev/mtd3ro
crw-------    1 root     root       90,   8 Jan  1 00:00 /dev/mtd4
crw-------    1 root     root       90,   9 Jan  1 00:00 /dev/mtd4ro
```

   - /dev/mtd0 is the "uboot_env" partition;
   - /dev/mtd1 is the "dtb" partition;
   - /dev/mtd2 is the "kernel" partition;
   - /dev/mtd3 is the "rootfs" partition;
   - /dev/mtd4 is the "userdata" partition.

4.4.9. Test Plan: Support fw_printenv/setenv commands in Linux

1. Power cycle the board.
2. Boot Linux from the QSPI Flash.
3. Run the `fw_printenv` command and verigy that is able to print the U-Boot Environment:

```
/ # fw printenv
addip=setenv bootargs ${bootargs}
ip=${ipaddr}:${serverip}:${gatewayip}:${netmask}:${hostname}:eth0:off
arch=arm
baudrate=115200
board=stm32h753-eval
board name=stm32h753-eval
bootargs=console=ttySTM0,115200 earlyprintk consoleblank=0 panic=3 ignore loglevel
ubi.mtd=3 rootfstype=ubifs root=ubi0:rootfs rw init=/init
...
sf_kernel_update=run sf_kernel_update_tftp
sf kernel update mmc=sf probe 0 && fatload mmc 0 ${loadaddr} ${image ubi} && sf erase
${kernel sf offset} ${kernel sf size} && sf write ${loadaddr} ${kernel sf offset}
${filesize}
sf kernel update tftp=sf probe 0 && tftp ${tftpdir}${image ubi} && sf erase
${kernel_sf_offset} ${kernel_sf_size} && sf write ${loadaddr} ${kernel_sf_offset}
${filesize}
sf rootfs update=run sf rootfs update tftp
sf rootfs update mmc=sf probe 0 && fatload mmc 0 ${loadaddr} ${rootfs} && sf erase
${rootfs_sf_offset} ${rootfs_sf_size} && sf write ${loadaddr} ${rootfs_sf_offset}
${filesize}
```

```
sf_rootfs_update_tftp=sf probe 0 && tftp ${tftpdir}${rootfs} && sf erase
${rootfs_sf_offset} ${rootfs_sf_size} && sf write ${loadaddr} ${rootfs_sf_offset}
${filesize}
sfboot=sf probe 0 && sf read ${loadaddr} ${kernel sf offset} ${kernel sf size} && run addip
&& bootm ${loadaddr}
soc=stm32h7
tftpdir=
uboot_env_sf_offset=0x0
uboot env sf size=0x10000
vendor=st
```

4. Define a test variable:

```
/ # fw_setenv  testvar testval
```

5. Reboot the board and boot to U-Boot:

```
/ # reboot
...
STM32H7-EVAL U-Boot >
```

6. Verify that the test variable exists and has the correct value:

```
STM32H7-EVAL U-Boot > print testvar
testvar=testval
STM32H7-EVAL U-Boot >
```

7. Boot Linux from the QSPI Flash.
8. Verify that the test variable is accessible from Linux and has the same value as in U-Boot:

```
/ # fw printenv testvar
testvar=testval
/ #
```