Provide support for UBIFS Flash file system in FreeRTOS i.MX RT105x BSP

Detailed Requirements and Design

rm3401-drad-1_2.doc

RM:	*3401*
Revision:	*1.2*
Date:	*4/26/2019*

TABLE OF CONTENTS

1. Overview

This project adds support for UBIFS Flash file system to FreeRTOS i.MX RT105x BSP.

2. Requirements

2.1. Detailed Requirements

The following are the requirements for this project:

1. Support NAND Flash partitions in FreeRTOS.
    - *Rationale*: Explicit customer requirement.
      Implementation: Section: "Design: Support NAND Flash partitions in FreeRTOS".
      Test: Section: "Test Plan: Support NAND Flash partitions in FreeRTOS".

2. Support UBIFS file system in NAND Flash in FreeRTOS. UBIFS file system will reside in a NAND Flash partition created in U-Boot.
    - *Rationale*: Explicit customer requirement.
      Implementation: Section: "Design: Support UBIFS file system in NAND Flash in FreeRTOS".
      Test: Section: "Test Plan: Support UBIFS file system in NAND Flash in FreeRTOS".

3. UBIFS data integrity and automatic recovery after power-cut, including power failures when writing to UBIFS.
    - *Rationale*: Explicit customer requirement.
      Implementation: Section: "Design: UBIFS data integrity and automatic recovery after power-cut".
      Test: Section: "Test Plan: UBIFS data integrity and automatic recovery after power-cut".

4. Perform stress test of UBIFS functionality.
    - *Rationale*: Explicit customer requirement.
      Implementation: Section: "Design: Perform stress test of UBIFS functionality".
      Test: Section: "Test Plan: Perform stress test of UBIFS functionality".

2.2. Detailed Non-Requirements

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

1. This project will support NAND Flash partitions defined by U-Boot. There will be no explicit API added to allow re-partition NAND Flash from FreeRTOS.
    - *Rationale*: Customer creates partitions in NAND Flash.
2. This project will support operations with UBIFS from a single FreeRTOS thread. Access to UBIFS from multiple threads will not be allowed.
    - *Rationale*: Costs reduction measure.

3. Design

3.1. Design: Support NAND Flash partitions in FreeRTOS

The U-Boot code in `drivers/mtd/mtdcore.c` and `drivers/mtd/mtdpart.c` will be reused to implement support for Flash partitions in NAND in FreeRTOS.

The source code will be located in the `third_party/u-boot/drivers/mtd` subdirectory of the `NandFlash_Test` application.

3.2. Design: Support UBIFS file system in NAND Flash in FreeRTOS

The Linux code for the UBI layer and UBIFS subsystem will be reused to implement UBIFS support in FreeRTOS.

The code will be located in `third_party/u-boot/drivers/mtd/ubi` and `third_party/u-boot/fs/ubifs` subdirectories of the `NandFlash_Test` application.

The code will implement the following APIs:

Function	Description	Comments
`int uboot_ubifs_mount(char *vol_name)`	Mount specified UBIFS partition	
`int ubifs_read(const char *filename, void *buf, loff_t offset, loff_t size, loff_t *actread)`	Read `size` bytes from offset `offset` of file `filename`	`filename` must be an absolute path; `offset` must be a multiple of 4096; if `size` is zero, all content of the file is read; amount of the actually read data is stored in `actread`
`int ubifs_write(const char *filename, void *buf, loff_t offset, loff_t size, loff_t *actwritten)`	Write `size` bytes to offset `offset` of file `filename`	`filename` must be an absolute path; `offset` must be a multiple of 4096; amount of actually written data is stored in `actwrite`
`int ubifs_mkdir(const char *filename)`	Create a subdirectory	`filename` specifies an absolute directory path; all upper-level directories in the path must exist
`int ubifs_rmdir(const char *filename)`	Remove an empty subdirectory	`filename` specifies an absolute directory path
`int ubifs_unlink(const char *filename)`	Remove a file, hard-link or symlink	`filename` specifies an absolute path

3.3. Design: UBIFS data integrity and automatic recovery after power-cut

Support for recovery of UBIFS data after a power-cuts is a standard feature of the Linux kernel code. Both UBI and UBIFS are tolerant to power-cuts, and they were designed with this property in mind.

3.4. Design: Perform stress test of UBIFS functionality

Code of UBIFS stress test will be added to the `UBIFS_ZPL_Init` function of the `NandFlash_Test` project. The code will create 10 directories each with 10 files with generated content of size from 10Bytes to 40KBytes. The content of the files will then be validated to make sure that data is written and read correctly.

4. Test Plan

4.1. Test Set-Up

4.2. 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:

- https://www.emcraft.com/zimplistic/rm3401

The page is protected as follows:

- Login: `zimplistic`
- Password: `mee8Rae2`

4.3. Downloadable Files

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

- `nandflash_test-20190422.zip` - Source code of the `NandFlash_Test` project.

4.4. Test Set-Up

4.4.1. Hardware Set-Up

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

- Zimplistic i.MXRT1052 SOM board.

4.4.2. Software Set-Up

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

The `NandFlash_Test` project should be imported to the MCUXpresso IDE and built as described in the instructions that Zimplistic provided to Emcraft.

4.5. Detailed Test Plan

4.5.1. Test Plan: Support NAND Flash partitions in FreeRTOS

Perform the following step-wise procedure:

1. Load and start the `NandFlash_Test` application in the MCUXpresso IDE.
2. Check in the Console window of MCUXpresso IDE that the application successfully initialized the NAND device and parsed the MTD partition string:

```
---mtdparts_init---
last_ids  :
env_ids   : nand0=gpmi-nand
last_parts:
env_parts : mtdparts=gpmi-nand:512k(bcb),2m(u-boot1)ro,2m(u-boot2)ro,-(rootfs)
last_partition :
env_partition  : nand0,3
---parse_mtdids---
mtdids = nand0=gpmi-nand
get_mtd_device_nm nand0 nand0
+ id nand
---parse_mtdparts---
mtdparts = mtdparts=gpmi-nand:512k(bcb),2m(u-boot1)ro,2m(u-boot2)ro,-(rootfs)
--- current_save ---
=> partition NULL
--- index partitions ---
=> mtddevnum NULL
=> mtddevname NULL
===device_parse===
--- id_find_by_mtd_id: 'gpmi-nand' (len = 9)
entry: 'gpmi-nand' (len = 9)
dev type = 2 (nand), dev num = 0, mtd-id = gpmi-nand
parsing partitions 512k(bcb),2m(u-boot1)ro,2m(u-boot2)ro,-(rootfs)
+ partition: name bcb                    size 0x00080000 offset 0x00000000 mask flags -1
get_mtd_device_nm nand0 nand0
+ partition: name u-boot1                size 0x00200000 offset 0x00000000 mask flags -1
get_mtd_device_nm nand0 nand0
+ partition: name u-boot2                size 0x00200000 offset 0x00000000 mask flags -1
get_mtd_device_nm nand0 nand0
'-': remaining size assigned
+ partition: name rootfs                 size 0xffffffff offset 0xffffffff mask flags -1
get_mtd_device_nm nand0 nand0
```

4.5.2. Test Plan: Support UBIFS file system in NAND Flash in FreeRTOS

Perform the following step-wise procedure:

1. Boot the board to U-Boot.
2. Install a clean UBIFS image to the rootfs partition.
3. Load and start the `NandFlash_Test` application in the MCUXpresso IDE.
4. Wait the `Unmounting UBIFS volume fs!` line printed in the MCUXpresso IDE Console Window.
5. Check that the application successfully printed lists of files in the root and test directories (named `test_dirXX`). Check that there are 10 files in each test directory named `fileXX` with sizes from 10 to 36964 bytes.
6. Check that there no error messages have been printed.
7. Reboot to U-Boot and check that the test files and directories exist and are readable.

4.5.3. Test Plan: UBIFS data integrity and automatic recovery after power-cut

Perform the following step-wise procedure:

1. Boot the board to U-Boot.
2. Install a clean UBIFS image to the rootfs partition.
3. Load and start the `NandFlash_Test` application in the MCUXpresso IDE.
4. Power off the board when the test application prints lines similar to the following:

```
writing 10 bytes to /test_dir00/file00 at offset 0x00000000
writing 20 bytes to /test_dir00/file01 at offset 0x00001000
writing 30 bytes to /test_dir00/file02 at offset 0x00002000
writing 40 bytes to /test_dir00/file03 at offset 0x00003000
```

```
writing 50 bytes to /test_dir00/file04 at offset 0x00004000
writing 60 bytes to /test_dir00/file05 at offset 0x00005000
writing 70 bytes to /test_dir00/file06 at offset 0x00006000
writing 80 bytes to /test_dir00/file07 at offset 0x00007000
writing 90 bytes to /test_dir00/file08 at offset 0x00008000
writing 100 bytes to /test_dir00/file09 at offset 0x00009000
```

5. Boot the board to U-Boot.
6. Check that the rootfs partition is mounted without errors and content of files the the test applicationed created before the power outage can be read.
7. Repeat the test multiple times.

4.5.4. Test Plan: Perform stress test of UBIFS functionality

Run the following step-wise test procedure:

1. Boot the board to U-Boot.
2. Install a clean UBIFS image to the rootfs partition.
3. For 20 times, start the `NandFlash_Test` application from the MCUXpresso IDE or U-Boot.
4. Boot the board to U-Boot.
5. Check that the rootfs partition is mounted without errors and content of files the the test applicationed created can be read.