Freescale MQX™ RTOS FFS Porting Guide

PRODUCT:	Freescale MQX™ RTOS
DESCRIPTION:	Freescale MQX RTOS FFS Porting Guide
RELEASE DATE:	July, 2014

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

Document Number: MQXFFSPG
Rev. 0, 07/2014

Table of Contents

1 Introduction

This document is to guide user in a process of porting Flash File System (FFS) between flash memory systems. This document should be used in conjunction with Freescale MQX™ Flash File System User Guide (MQXFFSUG).

The current FFS is only applied to a memory system consisting of one MCU and one NAND flash memory chip. This guide will show two possible examples in porting FFS. The first is to integrate FFS into a system with 2 NAND flash memory chips – this is limited only by the number of NAND flash memory chips concurrently supported by the NAND flash controller inside MCUs. The second example is to port FFS into a system with one NOR flash memory chip. In the market there are two types of flash memory NAND flash memory and NOR flash memory. This document is to cover the availability of these flash memories and should be updated for upcoming flash memory technologies.

2 FFS porting process overview

2.1 FFS overview

We firstly look at how FFS is organized and what should be modified to suite new flash memory system. For those boards supporting FFS we can see FFS is comprised of 3 main components as illustrated in Figure 1.

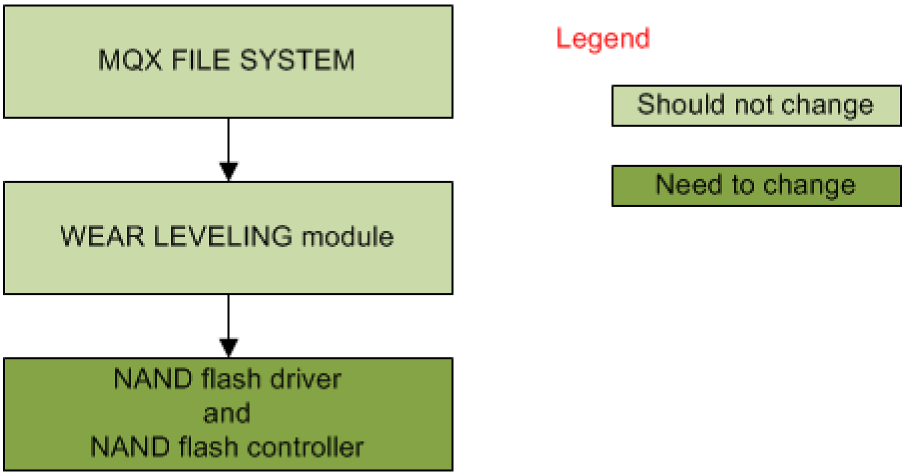

Figure 1: FFS components

This figure is obtained by re-organizing sub-blocks of figure Figure 3-3 in MQXFFSUG. In which the sub-block WEAR LEVELING module of Figure 1 comprises sub-blocks Media manager and Mapper manager in Figure 3-3 in MQXFFSUG.

2.2 FFS porting process overview

To port FFS to new development board the new MCU must have the flash controller either NAND flash controller or NOR flash controller. The following steps should be roughly followed.

1. Clone FFS – makes a copy of the current FFS, name does not need to be modified as FFS is not specific to any development board name.
2. Modify FFS source code mostly in folders

ffs\source\wearleveling\hal
ffs\source\wearleveling\nfc_wl

and some files inside folder *ffs\source* but not in folder *ffs\source\wearleveling*.

3. Modify I/O setting of the MCU to support new flash controller I/O interface.
4. Port example application to test new FFS – mostly not necessary to modify as the current example application only interact with the top layer of FFS – the MQX File System interface.

3 Cloning a Freescale MQX Flash File System

1. Because the FFS is dependent on the flash memory type and the flash controller supported in the MCU user needs only to select FFS folder in the MQX release that supports FFS.
2. Clone that FFS folder and modify the clone as needed.

4 Modify FFS source code

Depending on the characteristic of new flash memory system, the logical flow of functions in sub-blocks of FFS (refer to Figure 3-3 of MQXFFSUG) should not be modified. The modification is mainly carried out in the configuration and definition of constants and global data structures. This will be discussed in more detail in section 6 below where two porting examples will be examined.

5 Modify I/O setting of MCU

For different flash memory system with new development board the MCU could very well be different from the board currently supporting FFS. Therefore the I/O setting for flash controller (integrated in side MCU) should be modified in file ***init_gpio.c***.

6. Examples

This section discusses two examples. The first example is to port the FFS for NAND flash memory from development board with one NAND flash memory chip to development board with two NAND flash memory chips. The second example is to port the FFS from NAND flash memory to NOR flash memory in which FFS for development board with one NAND flash memory chip is ported to development board with one NOR flash memory chip.

6.1 NAND flash memory system with two NAND flash memory chips

The flash memory systems for board Vybrid twr-vf65gs10, twrk70f120m, twrk60f120m, twrmcf54418 all have only one NAND flash memory chip and the MCUs all have the NAND flash controller.

6.1.1 Cloning FFS

This step is described in section 3 of this document.

6.1.2 Modify FFS source code

To save effort during porting FFS the easiest way to extend the memory space of flash memory system based on the current flash memory system of existing development board is to use the same type of NAND flash memory chip in the new flash memory system. For example in board twrk70f120m, NAND flash memory chip MT29F2G16 is used. It is a wise choice to use two chips MT29F2G16 in the hardware design of new flash memory system. And this example is to follow this decision because little modification to FFS is required. In addition the same NAND flash controller is reused. Following modification is necessary to current FFS to support two NAND flash memory chips.

Modify files:

ffs\source\wearleveling\hal\ddi_nand_hal_nfcphymedia.cpp

ffs\source\wearleveling\hal\ddi_nand_hal_init.cpp

ffs\source\wearleveling\nfc_wl\ nfc_wl.cpp

This is to change the setting of reading/writing/erase functions to select which NAND flash memory chip to select during operation. As a result the core component of **NandFlash HAL** sub-block in NAND Flash Drivers (Figure 3-3 in MQXFFSUG) – the *g_nandHalContext* variable is updated to hold information of 2-chip NAND flash memory system.

6.1.3 Modify file *init_gpio.c*

To enable the NFC_CE I/O of NAND flash controller.

6.2 NOR flash memory system

This example is to port FFS into development board with MCU having NOR flash controller, and a NOR flash memory chip. The NOR flash memory chip has parallel I/O interface. This system is similar to the current NAND flash memory system however the internal structure of NOR flash memory is different from NAND flash memory in following points:

- There is no separation between main data area and spare area within a page.
- No default location for bad block marking.

However we reserve the concept of bad block in FFS for NAND flash memory system. And the other concepts including physical page, virtual page, logical block, logical sector, physical block, metadata, etc. are still applied.

6.2.1. Cloning FFS

This step is described in section 3 of this document.

6.2.2. Modify FFS source code

What need to be changed are all files in folders:

ffs\source\wearleveling\hal

ffs\source\wearleveling\nfc_wl

and files inside folder *ffs\source* but outside all other sub-folders.

The functionalities of functions are unchanged but we need to change the label from nand to nor. For example instead of *g_nandHalContext* we use *g_norHalContext*.

Clone folder mqx\source\io\nandflash into folder mqx\source\io\norflash and update all the files in that new folder according to the physical configuration of the NOR flash memory. Write new read/write/erase functions to operate on new NOR flash memory chip although the same logical flow of NAND flash memory is applied.

6.2.3. Modify file *init_gpio.c*

The NOR flash memory chip (parallel I/O interface) requires about 25~28 pins more than the NAND flash memory chip. This means if the NOR flash controller is embedded inside the MCU, the MCU I/O pins for NOR flash controller increases dramatically. We need to update file ***init_gpio.c*** to support NOR flash controller I/O interface.

7 Conclusion

Freescale MQX is a scalable RTOS that complements Freescale's broad family of micro-controllers. The MQX FFSs which are provided can be ported across all supported Freescale devices using the steps outlined in this guide. For the most up-to-date MQX information and documentation or to download MQX, visit the freescale.com/mqx.