Test Plan for FTM

Test Plan for FTM

1 Outline

This document is for the FTM (Flex Timer Module) in Linux kernel of MVF TOWER BOARD (XTWR-VF600) with VF6XX SoC, and describes test plan for each API/feature of such unit.

2 Test Environment

Toolchain:	The latest Linaro toolchain
Bootloader:	u-boot 2011.12
Kernel:	Freescale i.MX Linux 3.0.15 kernel
Rootfs:	rootfs on NFS

3 Target Module of the Test

FTM Driver

4 Test Plan

Create testing driver and use it for the test since FTM driver does not have the operational interface for application.

5 Testing Method

1. Preparation

Have the following setting in kernel configuration ON.
Character devices --->
[*] Flex Timer Module support

Copy test_program/mvf_testmodule.c to drivers/char/.
そThen add the following to the end of drivers/char/Makefile.

```
obj-y   += mvf_testmodule.o
```

2. Test of each timer

Test runs automatically as booting the kernel built by #1 above.

Test Plan for FTM

Datails

No.	Head	Item	Procedure	Points to be checked	Judge	Note
1		Timer allocation	Call ftm_alloc_timer function by ftm_channel=FTM0orFTM1 via testing driver.	Timer handle is obtained.		
2			Continue from the test above. Call ftm_alloc_timer function by ftm_channel=FMT_AVAILABLE_CHANNEL via testing driver.	Timer handle not allocated by #1 is obtained.		
3		Timer start	Continue from the test above. Call ftm_enable_timer function by TimerHandle: FTM0 via testing driver.	Negative value is returned and an error occurs (since setting by ftm_param_set function is not done.)		
4			Continue from the test above. Call ftm_param_set function via testing driver and set value, then call ftm_enable_timer function by TimerHandle: FTM0.	Successful timer start is returned.		
5	Interrupt	Periodic event	Continue from the test 2 above. Call ftm_param_set function via testing driver and set start/end value of event handler and timer as 0/0xffff, then call ftm_enable_timer function by TimerHandle: FTM0.	Event handler is called for each specified count.		
6		Change in timer value	Continue from the test 2 above. Call ftm_param_set function via testing driver and change the start/end value of timer to0/0x7fff, then call ftm_enable_timer function by TimerHandle: FTM0.	Event handler is called for each specified count. Event occurs twice more often than the operation of #5 above.		
7		Change in frequency division	Continue from the test 2 above. Call ftm_param_set function via testing driver. Set frequency division value to FTM_PARAM_DIV_BY_2 in addition to the setting #5 above, then call ftm_enable_timer function. Set frequency division value to FTM_PARAM_DIV_BY_16 and call ftm_enable_timer function (with other settings remain the same).	Event handler is called for each specified count. Frequency of event occurrence changes depending on the value of frequency division.		
8		Change in clock source	Continue from the test 2 above. Call ftm_param_set function via testing driver. Set clock source value to FTM_PARAM_CLK_SYSTEMCLOCK in addition to the setting #5 above, then call ftm_enable_timer function. Set clock source value to FTM_PARAM_CLK_EXTERNAL and call ftm_enable_timer function (with other settings remain the same).	Event handler is called for each specified count. Frequency of event occurrence changes depending on the value of clock source.		
9	Output Test	Timer read	Continue from the test above. Call ftm_read_counter function via testing driver.	Have the return value of 0 and have the timer value for pointer.		
10		Timer stop	Continue from the test above. Call ftm_disable_timer function by TimerHandle: FTM0.	Successful timer stop is returned. No event handler call (set at #5 above) occurs after stopping.		

11		Timer release	Continue from the test above. Call ftm_free_timer function by TimerHandle: FTM0.	Successful timer release is returned.		