Test Plan for PIT

Test Plan for PIT

1 Outline

This document is for the PIT (Periodic Interrupt Timer) driver 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

PIT Driver

4 Test Plan

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

5 Testing Method

1. Preparation

Have the following setting in kernel configuration ON.

```
Character devices  --->
[*] Periodic Interrupt 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 PIT

Datails

No.	Head	Item	Procedure	Points to be checked	Judge	Note
1		Timer allocation	Call pit_alloc_timer function by pit_channel=PIT0-7 via testing driver.	Timer handle is obtained.		
2			Continue from the test above. Call pit_alloc_timer function by pit_channel=PIT_AVAILABLE_CHANNEL via testing driver.	Timer handle not allocated by #1 is obtained.		
3		Timer start	Continue from the test above. Call pit_enable_timer function by TimerHandle: PIT0 via testing driver.	Negative value is returned and an error occurs (since setting by pit_param_set function is not done.)		
4			Continue from the test above. Call pit_param_set function via testing driver and set value, then call pit_enable_timerfunction by TimerHandle: PIT0.	Successful timer start is returned.		
5	Interrupt	Periodic event	Continue from the test 2 above. Call pit_param_set function via testing driver and set start value of event handler and timer, then call pit_enable_timer function by TimerHandle: PIT0.	Event handler is called for each specified count.		
6		Change in timer value	Continue from the test 2 above. Call pit_param_set function via testing driver and change the start value of timer to half of the #5 above, then call pit_enable_timer function by TimerHandle: PIT0.	Event handler is called for each specified count. Event occurs twice more often than the operation of #5 above.		
7	Output Test	Timer read	Continue from the test above. Call pit_read_counter function via testing driver.	Have the return value of 0 and have the timer value for pointer.		
8		Timer stop	Continue from the test above. Call pit_disable_timer function by TimerHandle: PIT0.	Successful timer stop is returned. No event handler call (set at #5 above) occurs after stopping.		
9		Timer release	Continue from the test above. Call pit_free_timer function by TimerHandle: PIT0.	Successful timer release is returned.		