

Test Plan for LPTMR



Test Plan for LPTMR

1 Outline

This document is for the LPTMR (Low-Power 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

LPTMR Driver

4 Test Plan

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

5 Testing Method

1. Preparation

Have the following setting in kernel configuration ON.

Character devices --->

[*] Low Power Timer Module support

Copy test_program/mvf_testmodule.c to drivers/char/.

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 LPTMR

Details

No.	Head	Item	Procedure	Points to be checked	Judge	Note
1		Timer allocation	Call <code>lpt_alloc_timer</code> function via testing driver.	Timer handle is obtained.		
2		Timer start	Continue from the test above. Call <code>lpt_enable_timer</code> via testing driver.	Negative value is returned and an error occurs (since setting by <code>lpt_param_set</code> function is not done.)		
3			Continue from the test above. Call <code>lpt_param_set</code> function via testing driver and set value, then call <code>lpt_enable_timer</code> function.	Successful timer start is returned.		
4	Interrupt	Periodic event	Continue from the test 1 above. Call <code>lpt_param_set</code> function via testing driver and set maximum counter value of event handler and timer, then call <code>lpt_enable_timer</code> function.	Event handler is called for each specified count.		
5		Change in timer value	Continue from the test 1 above. Call <code>lpt_param_set</code> function via testing driver and change the maximum value of timer to half of the #4 above, then call <code>lpt_enable_timer</code> function.	Event handler is called for each specified count. Event occurs twice more often than the operation of #4 above.		
6		Change in frequency division	Continue from the test 1 above. Call <code>lpt_param_set</code> function via testing driver. Set mode: <code>LPT_PARAM_PB_GF_BYPASS</code> and call <code>lpt_enable_timer</code> function. Change the value of <code>prs_value</code> and recall <code>lpt_enable_timer</code> function.	Event handler is called for each specified count. Frequency of event occurrence changes as the value of <code>prs_value</code> is changed.		
7	Output Test	Timer read	Continue from the test above. Call <code>lpt_read_counter</code> 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 <code>lpt_disable_timer</code> function via testing driver.	Successful timer stop is returned. No event handler call (set at #4 above) occurs after stopping.		
9		Timer release	Continue from the test above. Call <code>lpt_free_timer</code> function via testing driver.	Successful timer release is returned.		