Design Document for LPTMR Driver

1 Outline

This document describes the LPTMR (Low-Power Timer) driver in Linux kernel of MVF TOWER BOARD (XTWR-VF600) with MVF SoC. LPTMR driver provides Low Power Timer function by API for various drivers in kernel.

2 Existing code to be changed

All source code is newly written.

3 API of new functions

Define 6 APIs to control timer from driver.

3.1 lpt_alloc_timer function

Assign LP timer.

Drivers employing this timer use this function to obtain and control TimerHandle.

Prototype: int lpt_alloc_timer(void)

Return value: Negative value: Error
Positive value: TimerHandle

3.2 lpt_param_set function

Set timer by parameter, and register callback function for timer interrupt.

Prototype: int lpt_param_set (int timer_handle, struct mvf_lpt_request req,,
void (*event_handler)(void))

Argument: timer_handle: Handle obtained by lpt_alloc_timer
Req: Timer parameters (described below)
event_handler: Event handler (NULL can be specified)

Return value: Negative value: Error
0: Set successfully

■ struct mvf_lpt_request

Members of the structure are explained as below.

```
struct mvf_ftm_request{
        unsigned long    compare_value;
        unsigned short   timer_mode;

        unsigned short   pulse_pin_polarity;
        unsigned short   pulse_pin_select;

        unsigned short   prs_clock_sel;
        unsigned short   prs_bypass;
        unsigned short   prs_value;
};
```

・compare_value: Member to define maximum counter value of timer
Valid up to 16 bit.

・timer_mode: Member to define timer mode
Select from the following 2 parameters.
LPT_PARAM_TM_TIMECOUNTER (Timer counter mode)
LPT_PARAM_TM_PULSECOUNTER (Pulse counter mode)

・pulse_pin_polarity: Member to define pin polarity for pulse counter mode
Use when timer_mode is LPT_TM_PARAM_PULSECOUNTER.
Select from the following 2 parameters.
LPT_PARAM_PPP_ACTIVEHIGH
LPT_PARAM_PPP_ACTIVELOW

・pulse_pin_select: Member to define pin source setting in timer source.
Use when timer_mode is LPT_TM_PARAM_PULSECOUNTER.
Select from the following 4 parameters.
LPT_PARAM_PPS_INPUT0
LPT_PARAM_PPS_INPUT1
LPT_PARAM_PPS_INPUT2
LPT_PARAM_PPS_INPUT3

・prs_clock_sel: Member to define clock of prescaler

Select from the following 4 parameters

LPT_PARAM_PCS_CLOCK0

LPT_PARAM_PCS_CLOCK1

LPT_PARAM_PCS_CLOCK2

LPT_PARAM_PCS_CLOCK3

・prs_bypass: Member to define Glitch Filter

Select from the following 2 parameters

LPT_PARAM_PB_GF_ENABLE

LPT_PARAM_PB_GF_BYPASS

・prs_value: Member to define divider of prescaler/Glitch Filter detection threshold

Set divider when timer_mode is LPT_PARAM_TM_TIMECOUNTER, and set chattering elimination time when timer_mode is LPT_PARAM_TM_PULSECOUNTER.

Select from the following 16 parameters

LPT_PARAM_PV_DIV2_NA

LPT_PARAM_PV_DIV4_RISE2

LPT_PARAM_PV_DIV8_RISE4

LPT_PARAM_PV_DIV16_RISE8

LPT_PARAM_PV_DIV32_RISE16

LPT_PARAM_PV_DIV64_RISE32

LPT_PARAM_PV_DIV128_RISE64

LPT_PARAM_PV_DIV256_RISE128

LPT_PARAM_PV_DIV512_RISE256

LPT_PARAM_PV_DIV1024_RISE512

LPT_PARAM_PV_DIV2048_RISE1024

LPT_PARAM_PV_DIV4096_RISE2048

LPT_PARAM_PV_DIV8192_RISE4096

LPT_PARAM_PV_DIV16384_RISE8192

LPT_PARAM_PV_DIV32768_RISE16384

LPT_PARAM_PV_DIV65536_RISE32768

3.3 lpt_enable_timer function

Start timer.

An error occurs if it is not set by lpt_param_set function.

Prototype: int lpt_enable_timer (int timer_handle)

Argument: timer_handle: Handle obtained by lpt_alloc_timer

Return value: Negative value: Error

0: Start successfully

3.4 lpt_disable_timer function

Stop timer.

Prototype: int lpt_disable_timer (int timer_handle)

Argument: timer_handle: Handle obtained by lpt_alloc_timer

Return value: Negative value: Error

0: Stop successfully

3.5 lpt_read_counter function

Read counter value.

Counter value is 2 bytes and copy read-value of LPTMR_CNR (Counter Value) register to buffer.

Prototype: int lpt_read_counter (int timer_handle, unsigned long *counter)

Argument: timer_handle: Handle obtained by lpt_alloc_timer

Counter: Pointer of variable to obtain counter value

Return value: Negative value: Error

0: Read successfully

3.6 lpt_free_timer function

Release timer assigned by lpt_alloc_timer.

Prototype: int lpt_free_timer (int timer_handle)

Argument: timer_handle: Handle obtained by lpt_alloc_timer

Return value: Negative value: Error

0: Release successfully

4 Expected register settings

Parameters settable for 3.2 lpt_param_set function comply with processor manual.

5 Expected functionality and usage

This driver assumes that the following operations are done as a sequence from device driver.

1. Obtain handle by lpt_alloc_timer
2. Set parameter and register callback function by lpt_param_set
3. Start timer by lpt_enable_timer
4. Timer processing by callback function, or timer read and such
5. Stop timer by lpt_disable_timer
6. Release timer by lpt_free_timer at the time of driver unload

LPTMR driver employs platform framework and enables it by resource definition.
For example, when defining LTP;

```
static struct resource lpt_resources[] = {
    [0] = {
            .start = MVF_LPT_BASE_ADDR,
            .end =   MVF_ LPT_BASE_ADDR + SZ_4K-1,
            .flags = IORESOURCE_MEM,
    },
    [1] = {
            .start =MXC_INT_LPT,
            .end =MXC_INT_LPT,
            .flags = IORESOURCE_IRQ,
    },
};
static struct platform_device lpt_device = {
        .name = "lpt",
        .id = 0,
        .num_resources = 2,
        .resource = _resources,
};
```

Describe these definitions and define as platform resource by the following at startup initialization function of the kernel.

platform_device_register(&lpt_device);

6 Any other pertinent information

This driver is implemented by using framework of platform device.