Design Document for PIT Driver

1 Outline

This document describes the PIT (Periodic Interrupt Timer) driver in Linux kernel of MVF TOWER BOARD (XTWR-VF600) with MVF SoC.

PIT driver provides highly-accurate timer function by API for various drivers in kernel.

Out of 8 PIT timers, PIT0 can be used as long as it is not defined by the kernel as TICK timer.

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 pit_alloc_timer function

Assign PIT timer.

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

Prototype: int pit_alloc_timer (pit_channel ch)

Argument: ch:PIT channel (refer to enum below)

Return value: Negative value: Error

Positive value: TimerHandle

Note: If the kernel assigns PIT0 as System Timer, this allocation causes an error.

■ enum pit_channel

```
typedef enum {
        PIT0, PIT1, PIT2, PIT3, PIT4,PIT5,PIT6,PIT7
        PIT_AVAILABLE_CHANNEL
} pit_channel;
```

PIT_AVAILABLE_CHANNEL is used to obtain available channel of PIT.

3.2 pit_param_set function

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

Prototype: int pit_param_set (int timer_handle, unsigned long load_value,
void (*event_handler)(int ch))

Argument: timer_handle: Handle obtained by pit_alloc_timer
load_value: 32bit timer setting value (default value of down counter)
event_handler: Event handler (NULL can be specified)

Return value: Negative value: Error
0: Set successfully

3.3 pit_enable_timer function

Start timer.

An error occurs if load_value is not set by pit_param_set function.

Prototype: int pit_enable_timer (int timer_handle)

Argument: timer_handle: Handle obtained by pit_alloc_timer

Return value: Negative value: Error
0: Start successfully

3.4 pit_disable_timer function

Stop timer.

Prototype: int pit_disable_timer (int timer_handle)

Argument: timer_handle: Handle obtained by pit_alloc_timer

Return value: Negative value: Error
0: Stop successfully

3.5 pit_read_counter function

Read counter value.

Counter value is 4 bytes and copy read-value of PIT_CVALn (Current timer value) register to buffer.

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

Argument: timer_handle: Handle obtained by pit_alloc_timer
Counter: Pointer of variable to obtain counter value

Return value: Negative value: Error
0: Read successfully

3.6 pit_free_timer function

Release timer assigned by pit_alloc_timer.

Prototype: int pit_free_timer (int timer_handle)

Argument: timer_handle: Handle obtained by pit_alloc_timer

Return value: Negative value: Error
0: Release successfully

4 Expected register settings

Parameters settable for Timer Load Value of 3.2 pit_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 pit_alloc_timer
2. Set parameter and register callback function by pit_param_set
3. Start timer by pit_enable_timer
4. Timer processing by callback function, or timer read and such
5. Stop timer by pit_disable_timer
6. Release timer by pit_free_timer at the time of driver unload

PIT driver employs platform framework and enables it by resource definition.

For example, when defining PIT;

```
static struct resource pit_resources[] = {
    [0] = {
            .start = MVF_PIT_BASE_ADDR,
```

```
                .end =    MVF_PIT_BASE_ADDR + SZ_4K-1,
                .flags = IORESOURCE_MEM,
        },
        [1] = {
                .start =MXC_INT_PIT,
                .end =MXC_INT_PIT,
                .flags = IORESOURCE_IRQ,
        },
};
static struct platform_device pit_device = {
        .name = "pit",
        .id = 0,
        .num_resources = 2,
        .resource = pit_resources,
};
```

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

```
platform_device_register(&pit_device);
```

6 Any other pertinent information

This driver is implemented by using framework of platform device.