Ameba-Z SDK Suspend Resume API

This document introduces usage of tickless suspend resume API.

Table of Contents

1. API Summary

Tickless API	*Introduction*
<***pmu_register_sleep_callback***>	■ Register suspend/resume call back function for one module
<***pmu_unregister_sleep_callback***>	■ Unregister suspend/resume call back function
< ***pmu_register_delay_callback*** >	■ Register resume delay call back function for one module
<***pmu_unregister_delay_callback***>	■ Unregister resume delay call back function
<***pmu_acquire_wakelock***>	■ Acquire wake lock for one module
<***pmu_release_wakelock***>	■ Release wake lock
<***pmu_set_sysactive_time***>	■ Acquire wake lock, and release wake lock automatically after timeout

2. Suspend Resume Register

2.1. pmu_register_sleep_callback

Register suspend/resume call back function for <***nDeviceId***>, suspend callback function will be called by PMU before system enter sleep mode, and resume callback function will be called after system resume.

Notice: Yield OS is not permitted in suspend/resume callback function like: taskYIELD, vTaskDelay, mutex, sema and so on

Notice: pmu_set_sysactive_time is not permitted in suspend/resume callback function.

parameter	*Type*	*Introduction*
<***nDeviceId***>	***uint32_t***	■ Device ID need suspend/resume callback ■ typedef enum { PMU_OS =0, PMU_WLAN_DEVICE =1, PMU_LOGUART_DEVICE =2, PMU_SDIO_DEVICE =3, PMU_UART0_DEVICE =4, PMU_UART1_DEVICE =5, PMU_I2C0_DEVICE =6, PMU_I2C1_DEVICE =7, PMU_USOC_DEVICE =8, PMU_DONGLE_DEVICE =9, PMU_RTC_DEVICE =10, PMU_CONSOL_DEVICE =11, PMU_ADC_DEVICE =12, PMU_WAKWLOCK_TIMEOUT=13, PMU_DEV_USER_BASE =16,

		PMU_MAX =31 } PMU_DEVICE;
<***sleep_hook_fun***>	***PSM_HOOK_FUN***	■ Suspend call back function
<***sleep_param_ptr***>	***void****	■ Suspend call back function parameter
<***wakeup_hook_fun***>	***PSM_HOOK_FUN***	■ Resume call back function
<***wakeup_param_ptr***>	***void****	■ Resume call back function parameter

2.2. pmu_unregister_sleep_callback

Unregister suspend/resume call back function for <***nDeviceId***>.

parameter	*Type*	*Introduction*
<***nDeviceId***>	***uint32_t***	■ The same as pmu_register_sleep_callback

2.3. pmu_register_delay_callback

Register resume delay call back function for <***nDeviceId***>, Delay resume callback function will be called after system resume.

Notice: Yield OS is permitted in resume delay callback function.

Notice: pmu_set_sysactive_time is permitted in resume delay callback function.

parameter	*Type*	*Introduction*
<***nDeviceId***>	***uint32_t***	■ The same as pmu_register_sleep_callback
< ***late_resume_hook_fun*** >	***PSM_HOOK_FUN***	■ Resume delay call back function
<***late_resume_param_ptr*** >	***void****	■ Resume delay call back function parameter

2.4. pmu_unregister_delay_callback

Unregister resume delay call back function for <***nDeviceId***>.

parameter	*Type*	*Introduction*
<***nDeviceId***>	***uint32_t***	■ The same as pmu_register_sleep_callback

3. Wake Lock

3.1. pmu_acquire_wakelock

Wakelock is a 32-bit map. Each module own 1 bit in this bit map. FreeRTOS tickless reference the wakelock and decide that if it can or cannot enter sleep state.
If any module acquire and hold a bit in wakelock, then the whole system won't enter sleep state.

parameter	*Type*	*Introduction*
<nDeviceId>	***uint32_t***	■ The same as pmu_register_sleep_callback

3.2. pmu_release_wakelock

Release BIT[***nDeviceId***] of wakelock bit map, If wakelock equals to 0, then the system may enter sleep state after system idle.

parameter	*Type*	*Introduction*
<nDeviceId>	***uint32_t***	■ The same as pmu_register_sleep_callback

3.3. pmu_set_sysactive_time

This function set system active time, system cannot sleep before timeout.

parameter	*Type*	*Introduction*
<timeout>	***uint32_t***	■ system cannot sleep before timeout ■ unit is ms.

4. Example

Step1: Register suspend/resume/late_resume callback functions
Step2: Set system active for 5000ms
Step3: Release os wakelock
Step4: After 5000ms system will enter sleep, and “uart_suspend” will print
Step5: Uart Rx wakeup system and “uart_resume” will print
Step6: “uart_late_resume” will print
Step7: After 5000ms system will enter sleep again

```
u32 uart_suspend(u32 expected_idle_time, void *param)
{
    DBG_8195A("uart_suspend \n");
    return TRUE;
}

u32 uart_resume(u32 expected_idle_time, void *param)
{
    DBG_8195A("uart_resume \n");
    return TRUE;
}

u32 uart_lateresume(u32 expected_idle_time, void *param)
{
    DBG_8195A("uart_late_resume \n");
    pmu_set_sysactive_time(5000);
    return TRUE;
}

void psm_sleep_uart(void)
{
    // mbed uart test
    serial_init(&sobj_g,UART_TX,UART_RX);
    serial_baud(&sobj_g,38400);
    serial_format(&sobj_g, 8, ParityNone, 1);

    uart_send_string(&sobj_g, "UART IRQ API Demo...\r\n");
    uart_send_string(&sobj_g, "Enter sleep!!\n");
    uart_send_string(&sobj_g, "\r\n8195a$");
    serial_irq_handler(&sobj_g, uart_irq, (uint32_t)&sobj_g);
    serial_irq_set(&sobj_g, RxIrq, 1);
    serial_irq_set(&sobj_g, TxIrq, 1);

    pmu_sysactive_timer_init();

    pmu_register_sleep_callback(PMU_UART0_DEVICE, uart_suspend, NULL, uart_resume, NULL);
    pmu_register_delay_callback(PMU_UART0_DEVICE, uart_lateresume, NULL);

    pmu_set_sysactive_time( 5000);
    pmu_release_wakelock(PMU_OS);

    vTaskDelete(NULL);
}
```