```
static bool pwr_on2reduce(pwr_mgr_level_t current_level, pwr_mgr_level_t target_level, pwr_mgr_cmd_t cmd)
{
    pwr_our_level = target_level;
    if (asic_pwr_reduced_wfi_enable)
    {
        //DPRINTF(PWR_DBG_LOUD,("WFI-"));
        dbg_printf("S1-");                              ①
#if DEBUG && defined HAVE_UART && (HAVE_UART_DEBUG < 4)
        intEnable(uartGetIntNum( uartGetDebugUart() ));      ②
#endif
        // by having WFI here, instead of in OS, we can deterministically control the time required to hunker down into WFI.
        // if we let OS handle it, other threads can run rogue and extend the timing to get into WFI.
        asm volatile
        (
            " wfi ;"                                   ③
        );
#if DEBUG
        // VERY few things should wake up from this (IPC), but we need to do the wake..  XDB attaching will cause an issue if we dont...
        pwr_mgr_go_active_nowait(PWRMGR_UID_FULL_WAKE);
#endif
        asic_pwr_reduced_wfi_enable = false;
        // this better be the INTENDED wake INTERRUPT!!!
//        DPRINTF(PWR_DBG_LOUD, ("exit\n"));
        dbg_printf("X\n");                             ④
    }
    if (asic_pwr_reduced_active_enable)
    {
        asic_pwr_handle_reduced_active();                 ⑤
        asic_pwr_reduced_active_enable = false;
        // this better be the INTENDED wake INTERRUPT!!!
        //DPRINTF(PWR_DBG_LOUD, ("X\n"));
        dbg_printf("X\n");                             ⑥
    }
    return true;
}
```

reduced mode分为2种sub-mode

1. wfi (suspend 1 state)
2. halt self-refresh (suspend 2 state)

suspend 1 state即令R4进入执行 `wfi` instruction，r4l令串口输出"S1-"
suspend 2 state即执行asic_pwr_handle_halt_sr()

```
Performs Halt SR, power down pads, clk div R4, bypass DDR PLL. Spin while (all A53 WFI) && MC idle
```

①

进入 `wfi` 前在串口输出 `S1-`

②

在串口2的terminal上按任意键会让R4退出low power,这一行就是原因。

这里enbale UART interrupt,这样按键触发UART interrupt,令R4从 `wfi` 退出

③

执行 `wfi` instruction，令R4进入low power

④

从 `wfi` 退出后打印出"X"，表示从low power退出了

⑤

进入"suspend 2 state"处理

```c
static void asic_pwr_handle_reduced_active()
{
    if (1)
    //if (MC_REG->DRAM_STATUS & (MC_DRAM_STATUS_INIT_DONE02_MASK | MC_DRAM_STATUS_INIT_DONE01_MASK))
    {
        // this is multi CS - workaround issue of 2nd CS on DIMM not coming out of SR
        // Need to run Halt SR
        // we are asked to monitor A53 cores.  When they are in WFI, we can put DDR into HALT SR and bypass DDR PLL...
        void *hsr = (void *)LCM_0_BASE;
#ifdef HSR_SRAM_START
        hsr = (void *)HSR_SRAM_START;
#endif
#ifdef DEBUG
        DPRINTF(PWR_DBG_SOFT,("HSR @ %#x\n",(uint32_t)hsr));
#else
        //dbg_printf("HSR");
#endif
        dbg_printf("S2-");                            (A)
        memcpy(hsr,asic_pwr_handle_halt_sr,SIZEOF_HSR);     (B)
        cpu_dcache_writeback_region(hsr,SIZEOF_HSR);
        cpu_icache_invalidate_all();
        ((pfn_sleep_t)hsr)(asic_pwr_local_sleep_flags,asic_pwr_local_mc_flags);    (C)
        cpu_icache_invalidate_all();
#if defined HAVE_UART && (HAVE_UART_DEBUG < 4)               (D)
        if (GIC_REG->ICD_ISPR[(INTNUM_UART_INT_UART1+HAVE_UART_DEBUG)/32] & 1 << ((INTNUM_UART_INT_UART1+HAVE_UART_DEBUG) % 32))
        {
            pwr_mgr_go_active_nowait(PWRMGR_UID_FULL_WAKE); // PWRMGR_UID_IO_WAKE_SET_MS(10000));
        }
#endif
        return;
    }
    //DPRINTF(PWR_DBG_LOUD,("S2-"));
    dbg_printf("ASR");
    // we are asked to monitor A53 cores.  When they are in WFI, we can put DDR into SR and bypass core PLL...
    // we are in DDR, but this loop is **small** (less 1KB), with interrupts disabled, so it will reside in icache, and this will keep us out of DDR
    asic_pwr_handle_auto_sr(false);
    return;
}
```

(A)

进入"suspend 2 state"前输出"S2-"

(B)

这里把asic_pwr_handle_halt_sr()的code复制到LCM中去。因为在进入"suspend 2 state"后，DDR RAM就处于self-refresh状态，不能被

access，而常规code当然是在RAM中的，所以这里先把在"suspend 2 state"要运行的code在LCM中复制一份。
(C)
在LCM中运行asic_pwr_handle_halt_sr(),在该function中令R4进入low power
(D)
从low power("suspend 2 state")中退出，要唤醒R4上的threadx system。

⑥
输出表示退出"suspend 2 state"的"X"

同样在"suspend 2 state"下，用户依然可以通过在debug terminal上按任意键令R4退出low power.原因如下

```
    while (1)
    {

    ......

#if defined HAVE_UART && (HAVE_UART_DEBUG < 4)
        if (GIC_REG->ICD_ISPR[(INTNUM_UART_INT_UART1+HAVE_UART_DEBUG)/32] & 1 << ((INTNUM_UART_INT_UART1+HAVE_UART_DEBUG) % 32))
        {
            return;
        }
#endif
    }
```

在asic_pwr_handle_halt_sr()的while loop中会检查UART interrupt是否触发，是的化就从该函数返回，即退出low power.

Question: 什么条件触发R4进入"suspend 1 state" or "suspend 2 state"？

in ccsgit/r4/common/asic/88pa6220/lowpower/src/ipc_pwr_drv.c

```c
static bool ipc_pwr_save(pwr_mgr_level_t level, pwr_mgr_cmd_t cmd)
{

    switch(level)
    {
    case pwr_mgr_on_e:
        pwr_on();
        pwr_mgr_level = level;
        pwr_mgr_powered_up(pwr_mgr_id);
        asic_pwr_ipc_send_response();
        break;
    case pwr_mgr_reduced_power_e:
        pwr_reduced();
        pwr_mgr_level = level;
        pwr_mgr_powered_down(pwr_mgr_id);
        break;
    case pwr_mgr_lowest_power_e:
        pwr_lowest();
        pwr_mgr_level = level;
        pwr_mgr_powered_down(pwr_mgr_id);
        break;
    case pwr_mgr_off_e:
        pwr_off();
        pwr_mgr_level = level;
        pwr_mgr_powered_down(pwr_mgr_id);
        break;
    default:
        break;
    }
    return true;
}
```

pwr_reduced() –> asic_pwr_ipc_reduced()

```c
error_type_t asic_pwr_ipc_reduced( void )
{
    uint32_t set_flags=0;
    static uint32_t suspend_mode = 0;
    tx_event_flags_get(&pwr_ipc_event_flags, (PWR_IPC_EVENT_TRANSITION_SUSPEND_1 | PWR_IPC_EVENT_TRANSITION_SUSPEND_2), TX_OR_CLEAR, &set_flags, TX_NO_WAIT);
    if (set_flags & PWR_IPC_EVENT_TRANSITION_SUSPEND_1)
    {
        suspend_mode = 1;
        tx_event_flags_set(&pwr_ipc_event_flags,PWR_IPC_EVENT_RESP_SUSPEND,TX_OR);
        tx_event_flags_get(&pwr_ipc_event_flags, PWR_IPC_EVENT_RESP_SUSPEND_COMP, TX_OR_CLEAR, &set_flags, TX_WAIT_FOREVER);
        DPRINTF(PWR_SOFT, ("PWR_IPC: events = %#x\n",set_flags));
    }
    else if (set_flags & PWR_IPC_EVENT_TRANSITION_SUSPEND_2)
    {
        suspend_mode = 2;
        tx_event_flags_set(&pwr_ipc_event_flags,PWR_IPC_EVENT_RESP_SUSPEND,TX_OR);
        tx_event_flags_get(&pwr_ipc_event_flags, PWR_IPC_EVENT_RESP_SUSPEND_COMP, TX_OR_CLEAR, &set_flags, TX_WAIT_FOREVER);
        DPRINTF(PWR_SOFT, ("PWR_IPC: events = %#x\n",set_flags));
    }
    else
    {
#if (HAVE_AUTO_SUSPEND==1) || (HAVE_AUTO_SUSPEND==2)
        suspend_mode = HAVE_AUTO_SUSPEND;
#endif
    }
    asic_pwr_reduced_mode(suspend_mode);
    return OK;
}
```

asic_pwr_ipc_reduced() depends on pwr_ipc_event_flags中是否设置了
PWR_IPC_EVENT_TRANSITION_SUSPEND_1 or
PWR_IPC_EVENT_TRANSITION_SUSPEND_2来设置 `suspend_mode` variable来决定进入哪一种
reduced ppwer mode。

大致流程如下：

1. in A53 Linux low_power_idle driver

in handle_light_sleep()

> send_low_power_cmd(e_lpp_sys_sleep_level,
> (void*)e_lpp_sleep_level_suspend_1,0,false,1);

即handle_light_sleep()令R4进入的是 `wfi` low power mode in handle_rtos_sleep()

> send_low_power_cmd(e_lpp_sys_sleep_level,
> (void*)e_lpp_sleep_level_suspend_2,0,true,1);

即handle_rtos_sleep()令R4进入的是 `halt self-refresh` low power mode

1. on R4 size
   in asic_pwr_ipc.c/recv_callback()

```
    case e_lpp_sys_sleep_level:
        if ((u32 == e_lpp_sleep_level_wake))
        {
            tx_event_flags_set(&pwr_ipc_event_flags, (PWR_IPC_EVENT_WAKE | PWR_IPC_EVENT_WAKE_PENDING), TX_OR);
        }
        else if ((u32 == e_lpp_sleep_level_deep))
        {
            tx_event_flags_set(&pwr_ipc_event_flags, PWR_IPC_EVENT_SLEEP_DEEP, TX_OR);
        }
        else if ((u32 == e_lpp_sleep_level_suspend_1))    ①
        {
            tx_event_flags_set(&pwr_ipc_event_flags, PWR_IPC_EVENT_SLEEP_SUSPEND_1, TX_OR);
        }
        else if ((u32 == e_lpp_sleep_level_suspend_2))    ②
        {
            tx_event_flags_set(&pwr_ipc_event_flags, PWR_IPC_EVENT_SLEEP_SUSPEND_2, TX_OR);
        }
```

①②
分别设置PWR_IPC_EVENT_SLEEP_SUSPEND_1 or
PWR_IPC_EVENT_SLEEP_SUSPEND_2 flag

1. in pwr_mgr_ipc()

```
        tx_event_flags_get(&pwr_ipc_event_flags, PWR_IPC_EVENT_FLAGS, TX_OR_CLEAR, &set_flags, TX_WAIT_FOREVER);

        ......

        if (set_flags & PWR_IPC_EVENT_SLEEP_SUSPEND_1)
        {
#if (HAVE_AUTO_SUSPEND==1)
            ipc_send(my_ipc,e_lpp_sys_sleep_level,(void*)e_lpp_sleep_level_suspend1_auto,0);
#else
            if ((sleep_level == e_lpp_sleep_level_wake)||(sleep_level == e_lpp_sleep_level_suspend_2))
            {
                sleep_level = e_lpp_sleep_level_suspend_1;
                pwr_mgr_set_enable_pwr_mgmt(true);      // this allows us to AUTO RESLEEP if we WOKE on temp basis without the WAKEUP cmd.

 ③           tx_event_flags_set(&pwr_ipc_event_flags,PWR_IPC_EVENT_TRANSITION_SUSPEND_1,TX_OR);  // flag that we need to send a response when ready...

                pwr_mgr_set_module_pwr_level(PWRMGR_DRV_NAME_IPC, PWRMGR_IPC_LEVEL);
                pwr_mgr_set_module_pwr_level(PWRMGR_DRV_NAME_ASIC, PWRMGR_ASIC_LEVEL);        // stay in threadx, just WFI
                pwr_mgr_set_module_pwr_level(PWRMGR_DRV_NAME_INTERRUPT, PWRMGR_INTR_LEVEL);
                pwr_mgr_set_module_pwr_level(PWRMGR_DRV_NAME_WDTIMER, PWRMGR_WDT_LEVEL);

                pwr_mgr_go_low_power(true);
            }
#endif
        }
        if (set_flags & PWR_IPC_EVENT_SLEEP_SUSPEND_2)
        {
#if (HAVE_AUTO_SUSPEND==2)
            ipc_send(my_ipc,e_lpp_sys_sleep_level,(void*)e_lpp_sleep_level_suspend2_auto,0);
#else
            if ((sleep_level == e_lpp_sleep_level_wake)||(sleep_level == e_lpp_sleep_level_suspend_1))
            {
                sleep_level = e_lpp_sleep_level_suspend_2;
                pwr_mgr_set_enable_pwr_mgmt(true);      // this allows us to AUTO RESLEEP if we WOKE on temp basis without the WAKEUP cmd.

④            tx_event_flags_set(&pwr_ipc_event_flags,PWR_IPC_EVENT_TRANSITION_SUSPEND_2,TX_OR);  // flag that we need to send a response when ready...

                pwr_mgr_set_module_pwr_level(PWRMGR_DRV_NAME_IPC, PWRMGR_IPC_LEVEL);
                pwr_mgr_set_module_pwr_level(PWRMGR_DRV_NAME_ASIC, PWRMGR_AS
IC_LEVEL);        // stay in threadx, just WFI
                pwr_mgr_set_module_pwr_level(PWRMGR_DRV_NAME_INTERRUPT, PWRMGR_INTR_LEVEL);
                pwr_mgr_set_module_pwr_level(PWRMGR_DRV_NAME_WDTIMER, PWRMGR_WDT_LEVEL);

                pwr_mgr_go_low_power(true);
            }
#endif
        }
```

③④
设置PWR_IPC_EVENT_TRANSITION_SUSPEND_1 or PWR_IPC_EVENT_TRANSITION_SUSPEND_2 flag

1. in asic_pwr_ipc_reduced()

```
error_type_t asic_pwr_ipc_reduced( void )
{
    uint32_t set_flags=0;
    static uint32_t suspend_mode = 0;
    tx_event_flags_get(&pwr_ipc_event_flags, (PWR_IPC_EVENT_TRANSITION_SUSPEND_1 | PWR_IPC_EVENT_TRANSITION_SUSPEND_2), TX_OR_CLEAR, &set_flags, TX_NO_WAIT);
    if (set_flags & PWR_IPC_EVENT_TRANSITION_SUSPEND_1)
    {
        suspend_mode = 1;        ⑤
        tx_event_flags_set(&pwr_ipc_event_flags,PWR_IPC_EVENT_RESP_SUSPEND,TX_OR);
        tx_event_flags_get(&pwr_ipc_event_flags, PWR_IPC_EVENT_RESP_SUSPEND_COMP, TX_OR_CLEAR, &set_flags, TX_WAIT_FOREVER);
        DPRINTF(PWR_SOFT, ("PWR_IPC: events = %#x\n",set_flags));
    }
    else if (set_flags & PWR_IPC_EVENT_TRANSITION_SUSPEND_2)
    {
        suspend_mode = 2;        ⑥
        tx_event_flags_set(&pwr_ipc_event_flags,PWR_IPC_EVENT_RESP_SUSPEND,TX_OR);
        tx_event_flags_get(&pwr_ipc_event_flags, PWR_IPC_EVENT_RESP_SUSPEND_COMP, TX_OR_CLEAR, &set_flags, TX_WAIT_FOREVER);
        DPRINTF(PWR_SOFT, ("PWR_IPC: events = %#x\n",set_flags));
    }
    else
    {
#if (HAVE_AUTO_SUSPEND==1) || (HAVE_AUTO_SUSPEND==2)
        suspend_mode = HAVE_AUTO_SUSPEND;
#endif
    }
    asic_pwr_reduced_mode(suspend_mode);
    return OK;
}
```

⑤⑥
PWR_IPC_EVENT_TRANSITION_SUSPEND_1 ==> suspend_mode = 1
PWR_IPC_EVENT_TRANSITION_SUSPEND_2 ==> suspend_mode = 2

1. in asic_pwr_reduced_mode()

```
error_type_t asic_pwr_reduced_mode(uint32_t mode)
{
    if (mode == 1)
    {
        asic_pwr_reduced_wfi_enable = true;
        asic_pwr_reduced_active_enable = false;
    }
    if (mode == 2)
    {
        asic_pwr_reduced_active_enable = true;
        asic_pwr_reduced_wfi_enable = false;
    }
    return OK;
}
```

Summary:

handle_light_sleep()使得R4进入reduced `wfi` mode;而handle_rtos_sleep()使得R4进入reduced `halt self-refresh` mode。