pwr_handler[pwr_mgr_on_e][pwr_mgr_lowest_power_e] = pwr_on2lowest

r4 asic 从pwr_mgr_on_e向pwr_mgr_lowest_power_e切换的handler

```
static bool pwr_on2lowest(pwr_mgr_level_t current_level, pwr_mgr_level_t target_level, pwr_mgr_cmd_t cmd)
{
    static uint32_t low2on = 0;
    uint32_t on2low = asic_pwr_time_get();

    if (low2on && !asic_pwr_starting_session)
    {
        asic_pwr_add_sample(on2low, low2on);
    }
    asic_pwr_starting_session = false;
    ASIC_PWR_SLOG("asic %s entry\n",__FUNCTION__);
    asic_r4_to_lpp();
    low2on = asic_pwr_time_get();
    pwr_our_level = target_level;
    ASIC_PWR_SLOG("asic %s exit\n",__FUNCTION__);
    return true;
}
```

```
/**
 * asic_r4_to_lpp - handles transition of control from ASIC R4
 * processor to LPP processor
 */
static void asic_r4_to_lpp( void )
{
    uint32_t value = 0;

    uint32_t cpu_int = cpu_disable_interrupts();

    // set the target power alarm
    asic_pwr_set_target_power_alarm();
    // allow for manual override
    if (asic_pwr_max_sleep_ticks)
    {
        asic_pwr_set_cmd(e_lpp_sys_alarm_ticks,12345678);  // backdoor to cl
ear any previous
        asic_pwr_set_cmd(e_lpp_sys_alarm_ticks,asic_pwr_max_sleep_ticks);
    }
    asic_pwr_max_sleep_ticks = asic_pwr_max_sleep_ticks_default;
    // allow for manual override
    if (asic_pwr_min_sleep_ticks)   asic_pwr_set_cmd(e_lpp_sys_min_sleep_tic
ks,asic_pwr_min_sleep_ticks);

    // if cmd has not already by placed, place our version -- usually debug.
.
// current model, LP has a default, otherwise place a mailbox command.
//    if (FAIL == asic_pwr_get_cmd(e_lpp_sys_mode_flags,NULL))
//    {
//        asic_pwr_set_cmd(e_lpp_sys_mode_flags,asic_pwr_cmd);
//    }
    // if cmd has not already by placed, place our version -- usually debug.
.
    if (FAIL == asic_pwr_get_cmd(e_lpp_sys_debug_uart_num,NULL))
    {
        asic_pwr_set_cmd(e_lpp_sys_debug_uart_num,hwGetDebugUARTNumber());
    }

    // we don't need to track usec rollovers, mainly due to using APMU timer
s - share int complicates LP int routine..
    //asic_pwr_set_cmd(e_lpp_sys_usec_rsp,0);         // we don't need this
sent to AP, so we put place holder

①  asic_pwr_set_cmd(e_lpp_sys_cmd_divide,0x12345678);   // marks the final i
ncoming command - all after this are response commands.

    // debug hook to allow time for debug connection...
    if (asic_pwr_sleep_delay_time_in_seconds)
    {
        uint32_t delay = asic_pwr_sleep_delay_time_in_seconds;
        DPRINTF(PWR_DBG_LVL, ("PWR: DEBUG sleep delay=%ds.\n",asic_pwr_sleep
_delay_time_in_seconds));
        while (delay--)
```

```
46.         {
47.             cpu_spin_delay(1000000);
48.         }
49.     }
50.
51.     //uint32_t cpu_count = cpu_get_ccount();
52.     asic_pwr_test_usec_index=0;
53.     asic_pwr_test_usec_stamp();
54.     ASIC_PWR_GPIO_CHKPT_TOGGLE(50);
55.     // if if test we are skipping our sleep/warmboot....
56.     if (!asic_pwr_debug_skip)
57.     {
58. ②       transition_cpu();
59.     }
60.
61.     asic_pwr_test_usec_stamp();
62.     // calculate the overhead...
63.     //dbg_printf("transition: preLD %d; postLD %d; RET %d; HERE %d \n",(asic
    _pwr_test_usec[1]-asic_pwr_test_usec[0]),(asic_pwr_test_usec[2]-asic_pwr_tes
    t_usec[0]),(asic_pwr_test_usec[3]-asic_pwr_test_usec[0]),(asic_pwr_test_usec
    [4]-asic_pwr_test_usec[0]));
64.     //cpu_set_ccount(cpu_count);
65.     ASIC_PWR_GPIO_CHKPT_TOGGLE(50);
66.
67.     cpu_restore_interrupts(cpu_int);
68.
69.     asic_pwr_get_cmd(e_lpp_sys_usec_rsp,&value);
70.     if (value)
71.     {
72.         asic_pwr_sleep_usec_rollover+=value;
73.     }
74.     asic_pwr_sleep_ticks = 0;
75.     asic_pwr_get_cmd(e_lpp_sys_sleep_ticks_rsp,&asic_pwr_sleep_ticks);
76.
77.     asic_pwr_wakeup_iterations++;
78.     asic_pwr_cumulative_sleep_ticks += asic_pwr_sleep_ticks;    // track our
     cumulative sleep ticks
79.     asic_pwr_get_cmd(e_lpp_sys_alarm_ticks,&value);             // determine
     if alarm woke us up
80.     asic_pwr_set_cmd(e_lpp_sys_alarm_ticks,12345678);           // clear ala
    rm
81.     if (value == 0xFFFFFFFF)
82.     {
83.         asic_pwr_alarm_wakes++;
84.         //dbg_printf("asic_pwr_alarm_wakes %d\n",asic_pwr_alarm_wakes);
85.     }
86.     if (asic_pwr_wakeup_iterations_print && ((asic_pwr_wakeup_iterations % a
    sic_pwr_wakeup_iterations_print) == 0))
87.     {
88.         dbg_printf("asicpwr DRV WAKEUP iterations: %d; alarms %d; last sleep
     ticks: %d \n",(uint32_t)asic_pwr_wakeup_iterations, asic_pwr_alarm_wakes, a
    sic_pwr_sleep_ticks);
89.         dbg_printf("asicpwr system %dt = sleep %dt + wake %dt + overhead\n",
    asic_pwr_time_get(), asic_pwr_cumulative_sleep_ticks,tx_time_get());
```

```
90.         dbg_printf("asicpwr usec timer: R4ro %d + M3ro %d + %dus\n",timer_ge
    t_time_usec_rollover(),asic_pwr_sleep_usec_rollover,timer_get_time_usec());
91.
92.         asic_pwr_this_session_sleep_ticks = asic_pwr_cumulative_sleep_ticks
    - asic_pwr_last_session_sleep_ticks;
93.         asic_pwr_this_session_iowake_ticks = tx_time_get() - asic_pwr_last_s
    ession_iowake_ticks;
94.         dbg_printf("asicpwr this sleep session: sleep %dt ; wake %dt ; %d%%
    asleep                \n",asic_pwr_this_session_sleep_ticks,asic_pwr_this_ses
    sion_iowake_ticks,(asic_pwr_this_session_sleep_ticks*100)/(asic_pwr_this_ses
    sion_sleep_ticks+asic_pwr_this_session_iowake_ticks+1));
95.
96.         dbg_printf("asicpwr past %d seconds: sleep %dt ; wake %dt ; %d%% asl
    eep ; samples %d  \n", (asic_pwr_duty_cyle.log[asic_pwr_duty_cyle.head-1].ti
    mestamp-asic_pwr_duty_cyle.log[asic_pwr_duty_cyle.tail].timestamp)/100 ,asic
    _pwr_duty_cyle.tally_sleep,asic_pwr_duty_cyle.tally_iowake,(asic_pwr_duty_cy
    le.tally_sleep*100)/(asic_pwr_duty_cyle.tally_sleep+asic_pwr_duty_cyle.tally
    _iowake),asic_pwr_duty_cyle.samples);
97.
98.         dbg_printf("asicpwr schema: %s; action: %s\n",asic_pwr_schema->descr
    iption, asic_pwr_target_power_description);
99.         dbg_printf("action:    IOmin %d; SLmin %d; SLmax %d\n",asic_pwr_targ
    et_power_action->iowake_min,asic_pwr_target_power_action->sleep_min,asic_pwr
    _target_power_action->sleep_alarm);
100.        dbg_printf("overrides: IOmin %d; SLmin %d; SLmax %d\n",asic_pwr_min_
    iowake_ticks,asic_pwr_min_sleep_ticks,asic_pwr_max_sleep_ticks);
101.    }
102.    DPRINTF(PWR_DBG_LVL, ("ASIC PWR DRV WAKEUP iterations = %d; sleep ticks
    = %d \n",(uint32_t)asic_pwr_wakeup_iterations,asic_pwr_sleep_ticks));
103.
104.    // debug hook to allow time for debug connection...
105.    if (asic_pwr_wakeup_delay_time_in_seconds)
106.    {
107.        uint32_t delay = asic_pwr_wakeup_delay_time_in_seconds;
108.        DPRINTF(PWR_DBG_LVL, ("PWR: DEBUG wake delay=%ds.\n",asic_pwr_wakeup
    _delay_time_in_seconds));
109.        while (delay--)
110.        {
111.            cpu_spin_delay(1000000);
112.        }
113.    }
114.
115.    // this implementation is always a full wake
116.    pwr_mgr_go_active_nowait(PWRMGR_UID_FULL_WAKE);   // turn everything back
     on
117.
118. }
```

①

e_lpp_sys_cmd_divide command把asic_low_power_cmd_table[]分成 2 部分，之前的是command，之后的是"response"。

r4-threadx中的asic_low_power_cmd_table[],在lpp中是mailbox.

②

r4-threadx ==> lpp

③

从low power回到ready

```
/**
 * transition_cpu - handle process to transition to/from low
 * power cpu
 */
static void transition_cpu(void)
{
#ifdef DEBUG
    if (asic_pwr_stripe_sram)
    {
        // stripe most LCM (TEXT + DATA), skip mailbox...
        uint32_t *dst = (uint32_t*)LPP_VECTORS_ADDR;
        while (dst < (uint32_t*)(LPP_VECTORS_ADDR+(64*1024)))
        {
            if (dst >= (uint32_t*)LPP_MAILBOX_ADDR && dst < (uint32_t*)(LPP_MAILBOX_ADDR+LPP_MAILBOX_SIZE))
            {
                dst += (LPP_MAILBOX_SIZE/(sizeof(dst)));
                continue;
            }
            *dst++=0xdeadbeaf;
        }
    }
#endif
    memcpy((void *)LPP_MAILBOX_ADDR,asic_low_power_cmd_table,(sizeof(asic_low_power_table_t)*ASIC_LOW_POWER_MAX_CMDS));    ④

    // handle low level warmboot routine for asic
    WarmBoot_CPU(lpp_ucode_bin,lpp_ucode_bin_length);        ⑤

    memcpy(asic_low_power_cmd_table,(void *)LPP_MAILBOX_ADDR,(sizeof(asic_low_power_table_t)*ASIC_LOW_POWER_MAX_CMDS));    ⑥
    asic_pwr_table_transition = true;
}
```

④

r4-threadx与lpp共享的就是mailbox，也就是这里的asic_low_power_cmd_table[].

switch to lpp前先要把asic_low_power_cmd_table[]复制到LCM

⑤

switch to lpp

lpp_ucode_bin是lpp binary code,其size为lpp_ucode_bin_length

⑥

把lpp的mailbox从LCM复制回DDR RAM

in ccsgit/r4/common/asic/88pa6220/lowpower/src/warmBoot.S

```
;#*****************************************************************
;# WarmBoot_CPU
;# Purpose - handle handshake transition of control R4 -> lpp -> R4
;#    We do in ASM due to special needs:
;#      - BUS bug - needs special load TCM sequence
;#      - Ease of saving registers / restore registers
;# R0 - LCM Source address of ucode CPU code
;# R1 - length of ucode CPU code
;#
;#*****************************************************************
    .global WarmBoot_CPU
WarmBoot_CPU:
;# save our LR
        push {r0-r12}       ;# save current regs
        push   {LR}

;# save/change LP mpu region's....
        bl     Save_lpp_region

;# load ucode(lpp) f/w - hold in reset
        bl     Load_ucode_bin                ⑦

;# save key registers
        bl     Save_Registers

;# Transfer control to lpp
        bl     Handle_Warm_Boot              ⑧

;# restore key registers
        bl     Restore_Registers

;# restore LP mpu region
        bl     Restore_lpp_region

;# recover LR
        POP    {LR}
        pop {r0-r12}       ;# restore regs
        BX     LR
```

⑦

把lpp code load到LCM中去，因为lpp只会在LCM中运行，不能访问DDR RAM(这是DDR RAM处于self refresh state)

⑧

```
Handle_Warm_Boot:
        push   {LR}

;# clean cache to push ucode into memory
        bl     cpu_dcache_writeback_all

;# Sync Memory Controller - flush pipeline....
;# N/A for G2, since we leave cache on...?  And could use DDR in LP
;#        bl     Sync_Memory

        mov   r0, #0
        MCR p15, 0, r0, c7, c5, 0      @ Invalidate entire instruction cache
        ISB
        movw    R0,#:lower16:LPP_VECTORS_ADDR
        movt    R0,#:upper16:LPP_VECTORS_ADDR

;# SP WILL be preserved for us
⑨      blx   r0                    ;#***   EXE from LPP .... at some point w
e shall return ***
;# SP WILL be restored for us

        mov   r0, #0
        MCR p15, 0, r0, c7, c5, 0      @ Invalidate entire instruction cache
        ISB

        mEnterSVC

        bl      asic_pwr_test_usec_stamp

        POP    {LR}
        BX      LR                  ;#Return to Caller
```

⑨

> blx r0

跳转到LCM中的lpp的入口去执行

这个入口在ccsgit/r4/common/asic/88pa6220/lowpower/lpp_v1_0/os/src/lpp_startup.S

```
startup:    (A)
;# clear our stack space, carefully...
#ifdef DEBUG
    mov   r8, lr                        @ save LR
    ldr   r0, BOT_STACKS                @ First arg: start of memory block
    mov   r1,#0xa5                      @ Second arg: fill value
    ldr   r2, TOP_STACKS                @
    sub   r2, r2, r0                    @ Third arg: length of block
    bl    Memset                    @ this is OUR version of MEMSET, it does
 NOT USE STACK!!!
    mov   lr, r8
#endif

;# setup our stack and save PREVIOUS SP
    ldr   r0, SVC_STACK
    mov   r1, sp
    mov   sp, r0
    push {r1}                           ;# save previous SP on our new SP
    push {lr}

    LDR a3, IRQ_STACK                   ;# Pickup IRQ stack pointer
    MOV     a1, #IRQ_MODE                   ;# Build IRQ mode CPSR
    MSR     CPSR_c, a1                      ;# Enter IRQ mode
    MOV     sp, a3                          ;# Setup IRQ stack point
    MOV     sl, #0                          ;# Clear sl
    MOV     fp, #0                          ;# Clear fp

    LDR a3, XIQ_STACK                   ;# Pickup XRQ stack pointer
    MOV     a1, #ABT_MODE                   ;# Build ABT mode CPSR
    MSR     CPSR_c, a1                      ;# Enter ABT mode
    MOV     sp, a3                          ;# Setup ABT stack point
    MOV     sl, #0                          ;# Clear sl
    MOV     fp, #0                          ;# Clear fp

    LDR a3, XIQ_STACK                   ;# Pickup XRQ stack pointer
    MOV     a1, #UND_MODE                   ;# Build UND mode CPSR
    MSR     CPSR_c, a1                      ;# Enter UND mode
    MOV     sp, a3                          ;# Setup UND stack point
    MOV     sl, #0                          ;# Clear sl
    MOV     fp, #0                          ;# Clear fp


    MOV     a1, #SVC_MODE                   ;# Build SVC mode CPSR
    MSR     CPSR_c, a1                      ;# Enter SVC mode

;# clear our BSS
    ldr   r0, BSS_START             @ First arg: start of memory block
    ldr   r2, BSS_END
    mov   r1,#0                     @ Second arg: fill value
    sub   r2, r2, r0                @ Third arg: length of block
    bl    Memset_32B

    bl   main           (B)
```

```
53.
54.     pop  {lr}
55.     pop  {r1}                       ;# restore the PREVIOUS SP
56.     mov  sp,r1
57.     bx   lr                         ;# exit LPP exe    (C)
```

(A)

这就是lpp的入口

(B)

在为进入c language的main()准备好环境后就跳入main

(C)

从lpp退出，也就是退出lowest power mode