step 1

向R4发送的command / value pair.

```
    send_low_power_cmd(e_lpp_sys_sleep_level,   (void*)e_lpp_sleep_level_deep,0,true,1);
    // reset mailbox AFTER R4 sync resp
    send_low_power_cmd(e_lpp_sys_initialize,    (void*)0xFFFFFFFF,0,false,1);
    send_low_power_cmd(e_lpp_sys_initialize,    (void*)LOWPOWER_COMMAND_DATECODE,0,false,1);
    send_low_power_cmd(e_lpp_sys_power_service, (void*)lpi.lpp[e_lpp_level_deep].service,0,false,0);
    send_low_power_cmd(e_lpp_sys_power_wfi,     (void*)lpi.lpp[e_lpp_level_deep].wfi,0,false,0);
    send_low_power_cmd(e_lpp_sys_power_global,  (void*)lpi.lpp[e_lpp_level_deep].global,0,false,0);
    send_low_power_cmd(e_lpp_sys_ddr_mc_flags,  (void*)lpi.lpp[e_lpp_level_deep].mc,0,false,0);
    send_low_power_cmd(e_lpp_sys_avs,           (void*)lpi.lpp[e_lpp_level_deep].avs,0,false,0);
    send_low_power_cmd(e_lpp_sys_power_debug,   (void*)lpi.lpp[e_lpp_level_deep].debug,0,false,0);

    // send any commands queue for options/overrides
    send_low_power_cmd_q(NULL);

    // send AP sync to activate LPP LP mode, it will now monitor system, looking to engage shutdowns..
    send_low_power_cmd(e_lpp_sys_sleep_level,   (void*)e_lpp_sleep_level_deep_rdy,0,false,1);
```

step 2

in asic_pwr_ipc.c/recv_callback()

```
case e_lpp_sys_sleep_level:
       ......
        else if ((u32 == e_lpp_sleep_level_deep))
        {
            tx_event_flags_set(&pwr_ipc_event_flags, PWR_IPC_EVENT_SLEEP_DEEP, TX_OR);
        }
```

step 3

in pwr_mgr_ipc()

```
        if (set_flags & PWR_IPC_EVENT_SLEEP_DEEP)
        {
①          sleep_level = e_lpp_sleep_level_deep;
            pwr_mgr_set_enable_pwr_mgmt(true);      // this allows us to rec
over from IOwake (IPC mgs)
            // make sure we are awake from any possible 'suspend mode'
            pwr_mgr_go_active_wait(PWRMGR_UID_FULL_WAKE);
②          tx_event_flags_set(&pwr_ipc_event_flags,PWR_IPC_EVENT_TRANSITION_
WAKE,TX_OR);  // flag that we need to send a response when we wake up...

            pwr_mgr_set_module_pwr_level(PWRMGR_DRV_NAME_IPC, pwr_mgr_lowest
_power_e);          // transfer control to LP module
            pwr_mgr_set_module_pwr_level(PWRMGR_DRV_NAME_ASIC, pwr_mgr_lowes
t_power_e);         // transfer control to LP module
            pwr_mgr_set_module_pwr_level(PWRMGR_DRV_NAME_INTERRUPT, pwr_mgr_
lowest_power_e);    // transfer control to LP module
            pwr_mgr_set_module_pwr_level(PWRMGR_DRV_NAME_WDTIMER, pwr_mgr_lo
west_power_e);      // transfer control to LP module
            pwr_mgr_set_module_pwr_level(PWRMGR_DRV_NAME_WTM, pwr_mgr_off_e)
;          // transfer control to LP module
            asic_pwr_set_cmd(e_lpp_sys_initialize,cmd_lpp_sys_initialize);
            pwr_mgr_go_low_power(true);
        }
```

②
???

step 4

```
pwr_mgr_set_module_pwr_level(PWRMGR_DRV_NAME_IPC, pwr_mgr_lowest_power_e);
        // transfer control to LP module
pwr_mgr_set_module_pwr_level(PWRMGR_DRV_NAME_ASIC, pwr_mgr_lowest_power_e);
        // transfer control to LP module
pwr_mgr_set_module_pwr_level(PWRMGR_DRV_NAME_INTERRUPT, pwr_mgr_lowest_power
_e);    // transfer control to LP module
pwr_mgr_set_module_pwr_level(PWRMGR_DRV_NAME_WDTIMER, pwr_mgr_lowest_power_e
);      // transfer control to LP module
pwr_mgr_set_module_pwr_level(PWRMGR_DRV_NAME_WTM, pwr_mgr_off_e);          /
/ transfer control to LP module
asic_pwr_set_cmd(e_lpp_sys_initialize,cmd_lpp_sys_initialize);
pwr_mgr_go_low_power(true);
```

step 5

> pwr_mgr_go_low_power(true);

```
void pwr_mgr_go_low_power(bool force)
{
    pwr_mgr_msg_t msg;

    //DPRINTF(PWR_MGR_LOUD,("PWR: %s\n",__FUNCTION__));

    if (pwr_mgr_debug_off) dbg_printf_control(false,true);
    msg.msgType = MSG_PWR_MGR_POWER_DOWN;
    msg.param1 = force;

    pwr_mgr_send_msg(&msg);
}
```

==> tx_queue_send(&pwr_mgr_queue, (void *)msg, TX_NO_WAIT);

step 6

in pwr_mgr()

```
    while (1)
    {
        tx_queue_receive (&pwr_mgr_queue, (uint32_t *)&msg, TX_WAIT_FOREVER)
;

        state = pwr_mgr_state;

        ASSERT((uint32_t)state < pwr_mgr_initializing_e);
        ASSERT((msg.msgType >= MSG_PWR_MGR_MSG_BASE) && (msg.msgType <= MSG_
PWR_MGR_MSG_END));

        // given the STATE we are in, and MSG we received, call proper handl
er
        PFN_HANDLER pfn = pwrmgr_handler[state][PWRMGR_MSG_INDEX(msg.msgType
)].pfn;
        if (pwr_mgr_slog_on )   PWR_MGR_SLOG("PWRMGR: msg: %s\n", pwr_mgr_ms
g_name[PWRMGR_MSG_INDEX(msg.msgType)]);
        DPRINTF(PWR_MGR_DEBUG, ("PWR: %s @ %s\n",pwr_mgr_msg_name[PWRMGR_MSG
_INDEX(msg.msgType)],pwr_mgr_state_name[state]));
        if (pfn)
        {
            pfn(&msg);
        }
    }
```

> PFN_HANDLER pfn = pwrmgr_handler[state]
> [PWRMGR_MSG_INDEX(msg.msgType)].pfn;
> state = pwr_mgr_ready_e
> msg.msgType = MSG_PWR_MGR_POWER_DOWN

pwrmgr_handler[pwr_mgr_ready_e][MSG_PWR_MGR_POWER_DOWN] = pwr_mgr_ready_powerdown

==>

pwr_mgr_ready_powerdown(&msg)

step 7

in pwr_mgr.c/pwr_mgr_ready_powerdown()

```
/**
 *
 * pwr_mgr_ready_powerdown - handle POWERDOWN in state ready
 *
 * @param state
 * @param msg
 */
static void pwr_mgr_ready_powerdown(pwr_mgr_msg_t *msg)
{
    uint32_t force = msg->param1;

    // If the "force" parameter is set, then the power down
    // transition must occur - ignore if power manager is
    // enabled or if the system is online. Force power down
    // typically comes from soft power switch press.
    if ((pwr_mgr_enabled == true)  || force)
    {
        DPRINTF(PWR_MGR_SOFT, ("PWR: Got power down message! Online:  %d\n",
 pwr_mgr_sys_status_online));
        if (force)
        {
            // ignore/erase IOwake controls on force sleep
            pwr_mgr_io_up_extension_ticks=0;
            pwr_mgr_io_up_until_ticks=0;
        }
        if ((pwr_mgr_sys_status_online == true) || force)
        {
            // Only power down if fully "awake". Don't try to
            // change power levels if transitioning to wake, or
            // are already awake.
            if ((!ATIsListEmpty(&active_list)) && !force)    ①
            {
                DPRINTF(PWR_MGR_DEBUG, ("PWR: Skip power down message! Activ
e\n"));
                return;
            }
            // if timer was running, stop it now
            pwr_mgr_cancel_idle_timer();
            pwr_mgr_change_power_level(pwr_mgr_cmd_pwr_down_e,true);    ②
        }
    }
}
```

①

由于这里force = true，所以无论active_list空与非空，都会运行②

②

```
/**
 * pwr_mgr_change_power_level
 *
 * @param cmd - either power up, or power down
 * @param start - true if need to initialize starting priority.
 *
 * @return bool - true is successful, false on any error.  TBD -
 *         never fails now!!!
 */
static bool pwr_mgr_change_power_level(pwr_mgr_cmd_t cmd, bool start)
{
    bool pending;
    static pwr_mgr_pri_t priority=pwr_mgr_pri_start;
    static pwr_mgr_state_t pwr_mgr_rollback_state = pwr_mgr_transition_to_ready_e;

    // if start, we will pick which end of priority list to start at.
    if (start)                          (A)
    {
        // set some defaults
        pwr_mgr_abort_powerdown=0;        (B)
        pwr_mgr_rollback_state = pwr_mgr_transition_to_ready_e;

        if (pwr_mgr_cmd_pwr_up_e == cmd)
        {
            if (pwr_mgr_state == pwr_mgr_io_ready_e)
            {
                // we have already sent IOs pwr_IOup_e, but situations they could ignore
                // so tell IOs again to wake, this time ALL will wake to ready
                // then wake rest of drivers
                priority = pwr_mgr_pri_io_pwr_e;
                pwr_mgr_set_state(pwr_mgr_transition_to_ready_e);
            }
            else
            {
                priority = pwr_mgr_pri_system_1_e-1;  //system drivers are already up, start with next enum...
            }
        }
        else                (C)
        {
            if (pwr_mgr_state == pwr_mgr_io_ready_e)
            {
                // we have IO ready, now resleep, START at IO priority
                priority = pwr_mgr_pri_io_low_e;
                pwr_mgr_rollback_state = pwr_mgr_transition_to_io_ready_e;
            }
            else            (D)
            {
                priority=pwr_mgr_pri_lowest_e;
            }
```

```
50.             pwr_mgr_set_state(pwr_mgr_transition_to_low_power_e);    (E)
51.         }
52.     }
53.     // now loop on all priorities.
54.     // If one priority has a pending driver completion, we will exit and wai
    t for the msg CONTINUE.
55.     do
56.     {
57.         if (pwr_mgr_cmd_pwr_down_e == cmd)        (F)
58.         {
59.             pwr_mgr_check_for_abort();
60.         }
61.         // check for aborts at start of each priority level,  we can unwind
    the drivers...
62.         if (pwr_mgr_abort_powerdown)            (G)
63.         {
64.             if (pwr_mgr_cmd_pwr_down_e == cmd)
65.             {
66.                 // only handling aborts on - pwr_mgr_cmd_pwr_down_e
67.                 if (pwr_mgr_abort_powerdown == PWRMGR_UID_IO_WAKE)
68.                 {
69.                     // ignore IOWake aborts, until we reach IO priorities
70.                     if (priority >= pwr_mgr_pri_io_low_e)
71.                     {
72.                         // either we failed in our callbacks, or IO wake cam
    e in,
73.                         pwr_mgr_set_state(pwr_mgr_rollback_state);
74.                         cmd = pwr_mgr_cmd_pwr_up_e;
75.                         priority--;
76.                         pwr_mgr_abort_powerdown = 0;
77.                         pwr_mgr_go_active_nowait(PWRMGR_UID_IO_WAKE_SET_MS(1
    0)); // give a short IO wake, retry.
78.                     }
79.                 }
80.                 else
81.                 {
82.                     // either we have a job or request for full wake
83.                     pwr_mgr_set_state(pwr_mgr_transition_to_ready_e);
84.                     cmd = pwr_mgr_cmd_pwr_up_e;
85.                     priority--;
86.                     pwr_mgr_abort_powerdown = 0;
87.                 }
88.             }
89.             else
90.             {
91.                 ASSERT(0);
92.             }
93.         }
94.         if ((priority == pwr_mgr_pri_start) || (priority == pwr_mgr_pri_syst
    em_1_e))    (H)
95.         {
96.             // detected the completion of the power transition, send respect
    ive msg
97.             pwr_mgr_msg_t msg;
```

```
98.             if (pwr_mgr_cmd_pwr_up_e == cmd)
99.             {
100.                msg.msgType = MSG_PWR_MGR_WAKE_READY;
101.                msg.param1 = 0;
102.            } else
103.            {
104.                msg.msgType = MSG_PWR_MGR_DOWN_READY;
105.                msg.param1 = PWR_MGR_DOWN_READY_PHASE_1;
106.            }
107.            pwr_mgr_send_msg(&msg);
108.            break;
109.        }
110.        if ((priority == pwr_mgr_pri_io_low_e-1) &&         (I)
111.            ((pwr_mgr_state == pwr_mgr_transition_to_io_ready_e)||(pwr_mgr_s
    tate == pwr_mgr_transition_to_io_ready_continue_e)))
112.        {
113.            // detected the completion of the power IO ready transition, sen
    d respective msg
114.            pwr_mgr_msg_t msg;
115.            ASSERT (pwr_mgr_cmd_pwr_up_e == cmd);
116.            msg.msgType = MSG_PWR_MGR_WAKE_READY;
117.            msg.param1 = 0;
118.            pwr_mgr_send_msg(&msg);
119.            break;
120.        }
121.        pending = pwr_mgr_exec_callbacks(cmd, priority);    (J)
122.        if (pwr_mgr_cmd_pwr_up_e == cmd)
123.        {
124.            priority -= 1;
125.        } else                                          (K)
126.        {
127.            priority += 1;                               (L)
128.            // if we have reached IO, rollbacks will attempt to stay at IO..
    (unless full wake is required)
129.            if (priority == pwr_mgr_pri_io_low_e)
130.            {
131.                pwr_mgr_rollback_state = pwr_mgr_transition_to_io_ready_e;
132.            }
133.        }
134.    } while (!pending);
135.    return true;
136. }
```

(A)

start = true

(B)

make global variable pwr_mgr_abort_powerdown = 0

在pwr_mgr_exec_callbacks()中调用各个driver的callback，如果return fail，则会设置该global variable

in pwr_mgr_exec_callbacks()

```
success = p_node->function(level, send_cmd);
if (!success)
{
    // any ONE driver fails, means set global and deal with it at completion
 of this priority...
    DPRINTF( PWR_MGR_DEBUG,("PWR %s: ABORTED callback %s with priority %d, l
evel %d\n",(cmd == pwr_mgr_cmd_pwr_up_e)?"up":"down", p_node->name, p_node->
priority, p_node->level));
    pwr_mgr_abort_powerdown = (priority >= pwr_mgr_pri_io_low_e)?PWRMGR_UID_
IO_WAKE:PWRMGR_UID_FULL_WAKE;
}
```

(C)

cmd = pwr_mgr_cmd_pwr_down_e，so branch here

(D)

pwr_mgr_state = pwr_mgr_ready_e, so branch here.

因为是system由ready到low power down，所以通知driver的次序是从最低优先级到最高优先级。

(E)

pwr_mgr_state = pwr_mgr_transition_to_low_power_e;

pwr_mgr_state用于追踪system当前state。正在向low_power state过渡。(调用各个driver的callback期间被定义为过渡状态)

(F)

这里已经进入按照优先级调用各个driver的callback,有可能callback return fail，所以在这里要检查是否abort进入low power的过程

(G)

有driver的low power callback返回失败，就是表示该driver现在不能进入low power，这里是该情况的处理。

(H)

不是太明白???

(I)

pwr_mgr_state = pwr_mgr_pri_io_pwr_e, (E)处设置，所以if的条件不满足

(J)

按priority由低到高依次调用注册的driver的low power callback.

(K)(L)

因为是ready to low power，所以调用次序是由低到高

driver通过pwr_mgr_reg_callback()来注册low power的callback。

> uint32_t pwr_mgr_reg_callback(char *name, pfn_pwr_mgr_change_callback_t function, pwr_mgr_pri_t priority)

for example

> pwr_mgr_id = pwr_mgr_reg_callback(PWR_MGR_MY_NAME, audio_power_save,

> PWRMGR_AUDIO_PRIORITY);
> pwr_mgr_cpu_event = pwr_mgr_reg_callback("CPU", cpu_low_pwr_mode,
> pwr_mgr_pri_high_e);
> pwr_mgr_id = pwr_mgr_reg_callback(PWR_MGR_MY_NAME, gpio_power_save,
> PWRMGR_GPIO_PRIORITY);

step 8

在pwr_mgr_reg_callback()中priority应该到pwr_mgr_pri_asic_e为止啊！？
那么最后被调用的应该是asic_power_save()

> pwr_mgr_reg_callback(PWR_MGR_MY_NAME, asic_power_save,
> PWRMGR_ASIC_PRIORITY);

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

```
1.  bool asic_power_save(pwr_mgr_level_t target_level, pwr_mgr_cmd_t cmd)
2.  {
3.      // given the pwr STATE we are in (pwr_our_level), and new target level w
    e received, call proper handler
4.      PFN_HANDLER pfn = pwr_handler[pwr_our_level][target_level].pfn;
5.      DPRINTF(PWR_DBG_LVL, ("%s: @ <%s> ->> <%s>\n",__FUNCTION__,pwr_mgr_get_p
    wr_level_name(pwr_our_level),pwr_mgr_get_pwr_level_name(target_level)));
6.      if (pfn)
7.      {
8.          return pfn(pwr_our_level, target_level, cmd);
9.      }
10.     return true;
11. }
```

> pfn(pwr_our_level, target_level, cmd)

调用的应该是

> pwr_on2lowest(pwr_mgr_on_e, pwr_mgr_lowest_power_e,
> pwr_mgr_cmd_pwr_down_e)

```
1.  static bool pwr_on2lowest(pwr_mgr_level_t current_level, pwr_mgr_level_t tar
    get_level, pwr_mgr_cmd_t cmd)
2.  {
3.      static uint32_t low2on = 0;
4.      uint32_t on2low = asic_pwr_time_get();
5.
6.      if (low2on && !asic_pwr_starting_session)
7.      {
8.          asic_pwr_add_sample(on2low, low2on);
9.      }
10.     asic_pwr_starting_session = false;
11.     ASIC_PWR_SLOG("asic %s entry\n",__FUNCTION__);
12.     asic_r4_to_lpp();
13.     low2on = asic_pwr_time_get();
14.     pwr_our_level = target_level;
15.     ASIC_PWR_SLOG("asic %s exit\n",__FUNCTION__);
16.     return true;
17. }
```

> asic_r4_to_lpp();

ASIC在该function完成由r4-threadx向lpp的切换。