Title: How to debug power consumption issues

Author: Pinghua Zhang, Nina Li, Tony Yue

1.0 Introduction

This document mainly discusses debugging approaches of power consumption issues. It introduces power consumption testing procedure, HW/SW debugging steps and shares some customer power consumption issues on TI chipset platform.

2.0 I-Sample power consumption testing procedure

2.1 Hardware Setup

The role of test is to validate the power consumption of our chipsets, so we need to consider the basic components and exclude the impact of other peripherals devices like AGPS, Bluetooth and interface devices.

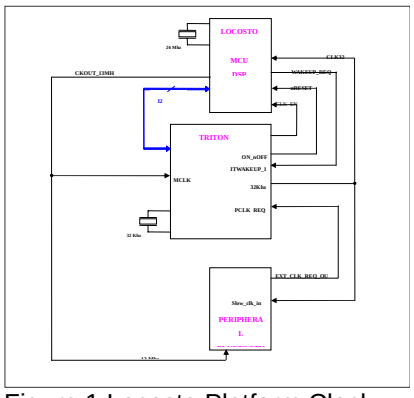

Figure 1 Locosto Platform Clock Implementation

- I-sample Version: V2.5
  - (Locosto ES2.0, Triton lite PG2.2, PA RFMD 7115)

- Test SIM card
  A test SIM card is required to communicate with CMU200

- Power voltage supply:
  Type: main power supply of DC 3.8V~4.2V.

- Current measurement kit:
  NI current acquisition equipment and PC measurement software.

- Test point to measure current (cf. **Error! Reference source not found.**):
  - J409 near DB35 connector for Handset
  - J409.1 is connected with the main power(with current measurement),plug out this jump connector.
  - J409.2 is used for plugged in second power(provided power for some external device, such as UART , USB , LCD)

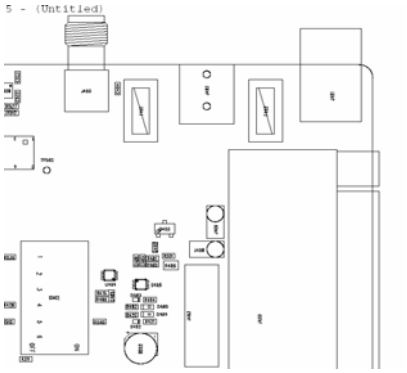

Figure 2.0 Measurement location on I-sample

- Ref Block
  - contains:
    Triton, Locosto, Memories MCP-RAM/Mirror-NOR, NAND , PA/RF-FEM, Wled, SIM

- does not contain:

LCD, secondary Back light, UART driver ,BT, Secondary Nand Flash

Notices:

Basically Locosto software version have two configuration for trace type, one is USB , the other one is UART. Each version has different hardware switch configuration on I-Sample.

For UART version.
ON: 1, 5
OFF: 2 , 3, 4, 6

One case of testing USB trace version,
ON: None
OFF: 1,2,3,4,5,6

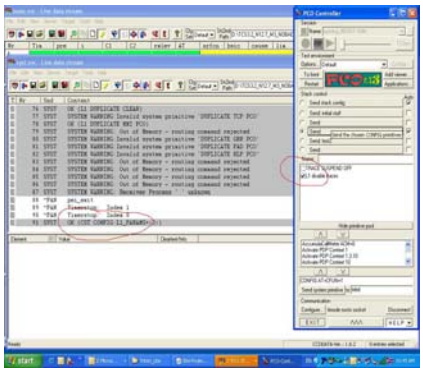

Figure 3.0 How to disable L1 trace with PCO tools

Please be noted on some versions system can not go into deep sleep if USB cable are connected.

When testing UART version with trace enabled, it's better to plug in UART cable after system has reset and is into idle state.

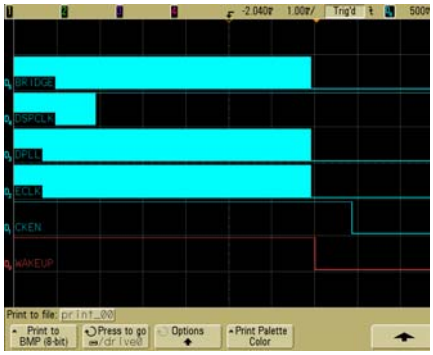

Figure 4.0 Locosto signals – ACTIVE to DEEP SLEEP sequence

2.2 Prepare a "no-L1-trace" version for testing

Please be noted to discable the L1 trace when testing power consumption cause it highly increases power consumption. Based our test experience, we can down 2mA average current if we disable L1 trace.

One way to disable L1 traces is through below configuration:
<property name="L1_SW_TRACE_TYPE" value="0"/>
<property name="TRACE_TYPE" value="0"/>

Another way to disable L1 trace is to use PCO tool ,as reference figur3.0.
Note, If you are compiling a no l1 trace version for your customer, you will meet a fatal link problem, don't worry it , take this action as follow, Open your file cst_pei.c file , and comment this function l1_cst_l1_parameters(s), it is only used for transfer some l1 configure parameter with PCO tool.

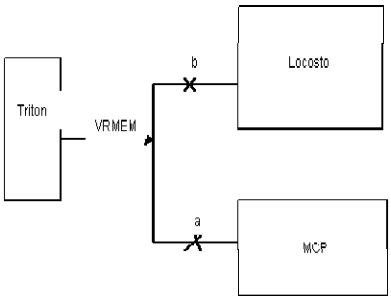

Figure 5.0 test point in VRMEM domain

3 Hardware check procedure

Firstly, we have to introduce some hardware feature. The main way is how to detect the sleep state according of some hardware signal. The core mechanism for save power consumption is cut off all core 13M clock and switch to 32K clock.

3.1 Test CLK 13M signal

Before testing the 13M clock, make sure the test points of 13M_clkout and WAKEUP_REQ exist on layout broad. 13M clock is output from Locosto to Triton. WAKEUP_REQ is the signal of controlling system sleep or wake-up. Other internal related signals can be referred from the below figure.
Once all the clocks are disabled the ULPD goes in Deep Sleep mode and set WAKEUP_REQ output signal to '0

3.2 Test WAKE_UP signal

Just refer to last section #3.1

3.3 Detect leak current from MCP

If Intel series MCP or other no SPANISH product is used, should refer this

section. In Locosto platform many new features have been supported like burst mode/sync access, muxed address/data bus. The primary reference document of how to configure your MCP could refer to APN215. Some additional suggestions are listed below.

1, Before doubting MCP cause current leakage, make sure all other peripherals and DBB IO setting are all ok. Usually MCP configuration has been well validated. And the measurement is also not so easy for MCP power consumption.
2, If MCP is doubted, below register setting need to be checked:

CLKM_IRQ_DIS | CLKM_TIMER_DIS| CLKM_CPORT_EN | CLKM_BRIDGE_DIS | 0x8000);

This signal bit must be set correct after system reset, it can be found in file init.c PDE, PWD_EN are sure to set as bit 1 . Refer to your function about how to configure your EMIF register in your MCP profile code.
Function , SET_EMIF_CONF_REG(0,0,2,1,1). On here, PDE and PWD_EN all set as enable. Then our flash can DMEM can enter into sleep automatically according of the system state.
Another hardware way to detect the MCP leak current is cut off the ABB power domain line on layout broad, detailed test approach is as below.
Using an external power supply MCP, on-board power only supplies locosto chipset. In case of on-board current is low enough, it is obviously that the MCP part have current leakage.
On the base of above test result, can try to replace with new MCP. If you ensure your all software configure for MCP is correct maybe need check MCP quality.

4.0 SW debugging procedure

In each frame interrupt, L1 sleep manager will check each peripheral like backlight, SIM, UART … state and decide whether system can go into deep sleep. With help of L1 trace we can check which peripheral can not meet the sleep requirement and prevent the whole system of entering deep sleep.

Below is an example of L1 trace that indicate sleep failed:

010422 15:07:33.132 FSL: 32392 1 blght 1
010423 15:07:33.162 FSL: 32401 L1S
010424 15:07:33.182 ADC :32406 0
010425 15:07:33.232 NP_I 32411 0 29 130 1757 -1784 14 14 14 14
010426 15:07:33.242 IBA_I 32414 7 45 (26 24)(27 20)(29 45)(32 27)(6 18)(19 17)(20 19)(25 24)
010427 15:07:33.252 FSL: 32412 1 i2c 1
010428 15:07:33.262 FSL: 32414 1 blght 1

32414 : Frame number
1: Means Check_Peripheral_App
Blight , Means “back light” not sleep .
1 means WAKEUP FOR L1 TASK,

Here, we can get the information from L1 trace that the reason of sleep failed is back light module can’t be into deep sleep state by 32414 1 blght 1

If handset enters deep sleep, trace contains key words of “deep_sleep” can be seen from the L1 trace

L1 power manager define a function array like below to indicate all external device module sleep call back function ,which is used by L1 sleep task. Every detailed

function is declared in every themselves module. And there are three main parameter used for SLEEP , WAKE_UP , MAKE .what's different means is SLEEP show this function is called for let it enter sleep state. WAKE_UP parameter is used to wake up some module with this general function. The parameter of MASK is used to check if the module is sleep or not.

```
const t_peripheral_interface  Peripheral_interface
[MAX_PERIPHERAL]=
{
        uart_pwr_interface,
#ifdef RVM_USB_SWE
        usb_pwr_interface,
#else
        f_peripheral_interface_dummy,
#endif
        usim_pwr_interface,
        i2c_pwr_interface,
        lcd_pwr_interface,
#ifdef RVM_CAMD_SWE
        camera_pwr_interface,
#else
        f_peripheral_interface_dummy,
#endif
        backlight_pwr_interface,
        f_peripheral_interface_dummy,
        audio_madc_sleep,
```

So according to these function rules, we could debug some special external device module. For example we need to check function lcd_pwr_interface() in file lcd_pwr.c when found the module of LCD can't enter into deep sleep mode.

Function Check_Peripheral_App() is the entry of system checking the "sleep" condition of peripherals. Below is an example for removing the LCD module from being checked before system going into deep sleep.

```
Check_Peripheral_App()
{
…
#if 0
ret_value = Peripheral_interface[LCD_ID](CLK_MASK);
   if(ret_value)
   {
     l1_pwmgr_debug.fail_id = LCD_ID;
     l1_pwmgr_debug.fail_ret_val = ret_value;
     return(DO_NOT_SLEEP);
   }
#endif
}
```

5. Optimize the average idle current

Next step is to optimize the power consumption. Base line current is about 1mA, maybe you get a average current about 10mA , First steps is check all unused GPIO ,make a list , configure as below ,

```
GPIO_DIRECTION_OUT(17);    GPIO_CLEAR_OUTPUT(17); //
CONF_GPIO_17= MUX_CFG(0, PULLOFF); //
```

The principle is: 1) DO NOT let any unused pin float, which means to set pin output mode or input mode with pull-up/pull-down; 2) for input-mode-only pin an external pull-up/pull-down is MUST; 3) avoid mode conflict for connected pins, e.g. can't set a couple of connected pins both in output mode.
For Locosto platform the detailed pin configurations can refer to APN206.xls

Suggest all unused GPIO configured as OUT state, and internal PULL_OFF. If your system base line current is over 1mA , please check them. According of our experience, the base line current is below 0.5mA on locosto platform.

Besides the above configure GPIO, Another note must be checked is all other IO pin ,which include those unused device pin, such as CAMERA , MMC ,and so on. These detailed IO configure pin , you are able to find and configure them.
List in file init.c, all function about GPIO is pin_configuration_xxx(). Suggest to configure them according of your hardware on your layout device broad.

6. Some shared case from customer

6.1 Inappropriate MCP configuration cause higher power consumption

Original issue is that AMOI report their power consumption is about 20mA average current on real network. Their target is about 5mA, we almost can't believe their test result. So firstly we doubt their system not go into deep sleep at all. But now we are confident to tell you their system have gone into deep sleep state. A direst reason is if system no into deep sleep , whose average is about over 40mA.
After we checked their L1 trace and some hardware signal , we gave some suggestion based on last section test way.
1, Check all external device driver, which include MCP .
2, Calibration their DRP , since we found dome cell drop off happed occasionally within about ten minutes.
3, try to exclude other hardware device from their layout broad.
4, we test our original MMI version for it's power consumption on I-sample.

After we take this action, we found the issue is affected by many kind of factors. The first issue is our original version has bad power consumption. so we check all original released version one by one. At last, found the main important reason is some compile flag involved some un-matured feature, which is "L1_SAIC" .
The followed action is help and guide customer to check their driver software and MMI part code with those suggestion, test ways written on before section.

6.2 Incorrect GPIOs configuration cause higher power consumption

Incorrect GPIOs setting was often found in customer applications. It resulted in uncertain floor current depending on different chipset lot number. This phenomenon often makes customer doubt chipset quality. The most often asked question is: why different chip brings different floor current. But in all past such cases the cause was always incorrect pin setting either by software or by hardware.

One example from Lenovo is: at first a locosto-plus based board worked well, while another project switched locosto-plus to locosto-base on the same PCB. Higher power consumption was found on locosto-base board even with the same software. Customer doubted some unknown difference exists between locosto-plus and locosto-base which brings different power consumption.

After checking GPIOs setting found some unused pins were left floating (set as input without pull-up/pull-down). For locosto-plus it happened this floating status made small impact on current leakage while for Locosto-base it unfortunately caused big current leakage.
The conclusion is: making pin floating can result in an UNCERTAIN current leakage, maybe very low, maybe very high.

6.3 MMI configuration causing more L1 activity and higher power consumption

It is very common for customer to report abnormal higher power consumption when they are performing field test, sometimes, it is due to the configuration of local network and customer's MMI which causes L1's extra activity. Besides the DRX settings, some network have cell broadcasting, if customer's MMI have configured cell broadcasting, L1 will have to receive CB_I blocks frequently. This can be observed by L1 trace "CB_I". Under such condition we can ask customer to close the Cell broadcasting function to reduce the power consumption.

6.4 Bad RF performance or bad network situation cause higher power consumption.

If RF performance is not so good, or network situation is bad, handset will perform frequent cell reselection or network drop and searching. This will highly increase the power consumption of handset.
We can observe from L1 trace, CEL_R indicates the cell reselection, while LOS_R indicates the network drop. If frequent CEL_R and LOS_R is seen from trace we need to do compare tests on other handset and then check the RF performance.

6.5 Inappropriate audio path configuration cause higher power consumption

It is customer MMI's responsibility to open/close audio path before/after the audio activity. Some customers would like to maintain the audio path even in idle mode, this will cause unnecessary power consumption.
So we always suggest customer to disable the audio path when handset enters sleep mode, this can be easily checked by ETM command aur 6 and aur 14, both should return "0" for audio path all shut.
In code, below code will shut all the mono and stereo audio path.
mfw_unset_stereo_path (AUDIO_SPEAKER_NONE)
mfw_set_stereo_path (AUDIO_SPEAKER_NONE)

6.6 Interference on UART signal cause sleep failure

UART RX and CTS must be pulled up with external resistors at the same time disable internal pull-up. With standard software release Smart Idle Mode is selected for chip internal modules and bus interfaces, in smart idle mode RX and CTS without external pull-up will prevent system from entering sleep.
If there's no external pull-up, L1 trace will always show UART is busy and can't enter sleep. After adding external pull-up at RX and CTS sleep will become ok again.

6.7 Complexity of debugging power consumption issue in dual-mode phone

In some customer dual-mode phone (GSM+CDMA) designs Calypso/ Locosto platform is used only as GSM modem. In such applications the power consumption need more considerations and the related issue becomes more complex than normal phone. How to locate unexpected power consumption is from GSM side or from CDMA side or by the interconnection between both sides? How to judge the unexpected wake-up is caused by GSM software or by CDMA side? The debug is becoming challenging.

Customer Hisense is using Cylipso as GSM modem and Qualcomm CDMA platform as master controller. The case is:
1) system floor current is about 6mA;
2) average idle current is about 20mA. Too high idle power consumption.
Customer found with early lot number of calypso the floor current is correct with about 1.1mA (GSM 0.4mA + CDMA 0.4mA + peripheral 0.3mA); while with new lot number of Cylipso the floor current is as high as about about 6~10mA. Therefore TI chipset quality is doubted by customer.
After fully checking customer design and times tests below results are gotten:

1) Because Cylipso is used only as GSM modem, many IO pins are not used and left float. It made different lots number of Calypso may result in different floor current, it easily made user misunderstand the chipset quality.
2) Cylipso UART is connected with CDMA chipset, incorrect configuration in CDMA side would wake up Cylipso with more times which would increase average current.
3) Incorrect peripheral (SIM card switch) control caused current leakage from GSM side: a CDMA controlled power supply SIM switch. During sleep mode the switch power was off, it caused SIM signals current leakage to ground.

6.8 Vocoder incorrect setting cause sleep failure

Sometimes customer report after an active call, handset fails to enter sleep mode. We found it is always caused by Vocoder incorrect setting via MMI or ACI bug. To check the Vocoder state we need to search the “VDS_C” (represent Vocoder close) and “VEN_C” (indicate Vocoder open) from L1 trace.
After call disconnected, Vocoder should be closed otherwise system will not enter deep sleep.To disable the Vocoder we need to send MMI_TCH_VOCODER_CFG_REQ to L1 from MMI.

6.9 Inappropriate MMI timer cause higher power consumption even lose network

This can be observed from L1 trace like below:
NP_I
IBA_I
deep_sleep
deep_sleep
deep_sleep
deep_sleep
NP_I
IBA_I
deep_sleep
deep_sleep
deep_sleep
deep_sleep

Too frequent wake up due to MMI timer will increase the power consumption, and if timer is too frequent, handset may lose network. As deep sleep means handset switch from the 13Mhz and 32 Khz clocks. Each time handset switch of clock it loses some accuracy in the Mobile time base. Adding a timer will wake up the system more often, thus causing adding a small error in the time base each time. This error is normally corrected by the Paging decoding thanks to the DSP demodulation which corrects this drift of the time base

6.10 Inaccurate 13MHz and 32KHz clock cause power consumption increase even handset lose network

Inaccurate 13MHz and 32KHz clock will cause higher gauging failure then sleep failure, thus the handset will sleep less than expected, and it will result in higher average power consumption.
Inaccurate 13MHz and 32KHz clock will also cause DSP decoding error and network lost.
To debug this issue we can connect standard clock to customer PCB to compare test

7.0 Conclusion

In general, there are many factors affecting power consumption, here we only wish this document could give you some guide when you meet this kind of customer issue.