PAN1080 LP Sample Application Note

PAN-CLT-VER-B0, Rev 0.2

PanchipMicroelectronics

www.panchip.com

修订历史

版本	修订日期	描述
V0.1	2021-10-17	初始版本创建
V0.2	2023-09-29	更新文档格式

目录

第1章 例程演示内容

1.1 测试内容

1. 低功耗进入与唤醒
   a) 进入 DeepSleep 模式并唤醒
   b) 进入 Standby Mode 1 模式并唤醒
   c) 进入 Standby Mode 0 模式并唤醒
2. 底电流测试
   a) 测试 DeepSleep 状态下底电流
   b) 测试 Standby Mode 1 模式下底电流
   c) 测试 Standby Mode 0 模式下底电流

1.2 环境准备

1.2.1 软件环境

1.2.1.1 待测代码

测试工程文件：

<PAN1080-DK>\03_MCU\mcu_samples\LP\keil\LP.uvprojx

测试源文件目录：

<PAN1080-DK>\03_MCU\mcu_samples\LP\src

1.2.1.2 软件工具

1、 SecureCRT（用于显示 PC 与 EVB 的交互过程，打印 log 等）

2、 Nordic Power Profiler Kit II（电流测试工具，包括硬件与上位机软件）

1.2.2 硬件环境

1、 PAN1080 EVB 1 块

   a) UART0（测试交互接口，TX: P00，RX: P01，波特率: 921600）

   b) SWD（用来调试和烧录程序，SWDCLK: P46，SWDIO: P47）

2、 Nordic Power Profiler Kit II（PPK2 电流测试工具）

3、 JLink（SWD 调试与烧录工具）

第2章 例程演示流程

2.1 环境配置

2.1.1 测试程序编译烧录

打开测试工程，确保可以编译通过。

2.1.2 硬件接线

接线方面，不同的测试项连线稍有不同：

1. 在测试不同低功耗模式进入与唤醒功能时，可将 EVB 板的 TX0 和 RX0 分别使用跳线帽连接到 P00 和 P01，然后在 PC 上通过串口工具进行测试交互。

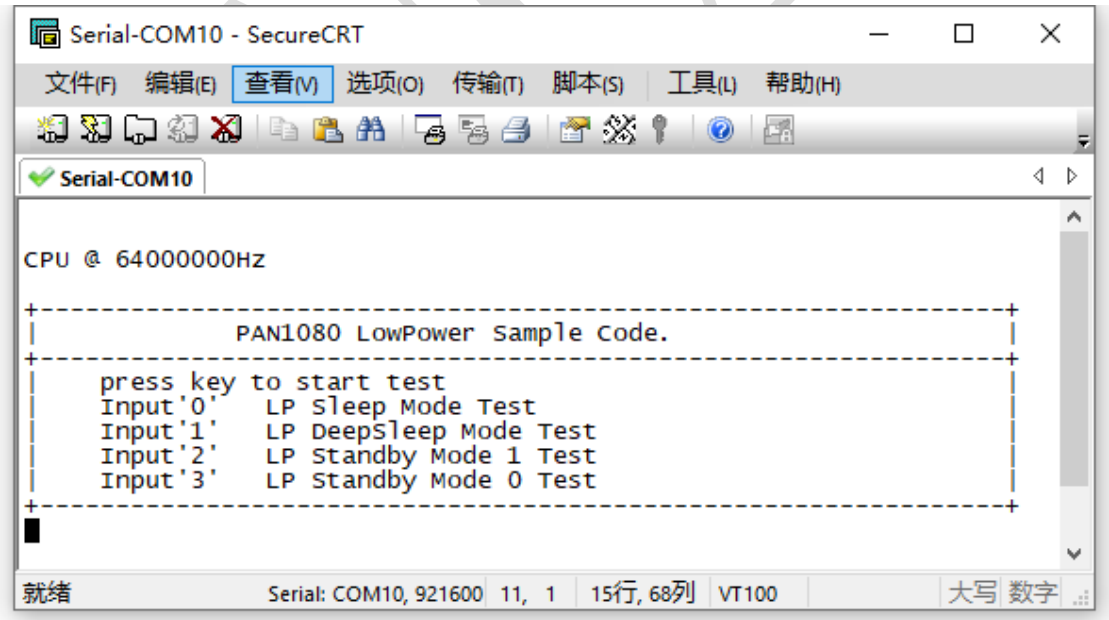
2. 在测试不同低功耗模式下底电流时，应将 EVB 子板从底板上拔出，然后仅将 VBAT/GND 引脚连接至 PPK2 工具上，进行电流测试。

2.2 低功耗流程

参考 User Manual 文档。

2.3 测试程序初始化

硬件连线完成并烧录测试程序后，EVB 上电，观察串口是否正常打印测试主菜单。



2.4 低功耗进入与唤醒功能测试

2.4.1 进入 DeepSleep 模式并唤醒

在主菜单下，输入‘1’命令 进入 Subcase 菜单：

```
1
+------------------------------------------------------------------+
|    Press key to test specific function:                          |
|                                                                  |
|    Input 'A'    32K LP Timer Wakeup.                             |
|    Input 'B'    Normal Timer Wakeup.                             |
|    Input 'C'    GPIO Wakeup.                                     |
|    Press ESC key to back to the top level case list.             |
+------------------------------------------------------------------+
```

2.4.1.1 32K LP Timer 唤醒

测试目的：

验证 DeepSleep 状态下使用 32K LP Timer 唤醒功能是否正常。

测试预期：

LP Timer 可以正常将芯片从 DeepSleep 状态下唤醒。

测试现象：

输入‘A’命令，可以从 Log 看到芯片进入低功耗状态，并在 2s 左右成功唤醒：

```
aTest LP Timer Wakeup..
Successfully waked up by LP Timer!

+------------------------------------------------------------------+
|    Press key to test specific function:                          |
|                                                                  |
|    Input 'A'    32K LP Timer Wakeup.                             |
|    Input 'B'    Normal Timer Wakeup.                             |
|    Input 'C'    GPIO Wakeup.                                     |
|    Press ESC key to back to the top level case list.             |
+------------------------------------------------------------------+
```

测试分析：

程序配置 LP Timer 唤醒时间为 2 秒，然后芯片进入 DeepSleep 状态，当 LP Timer 超时时间到达后，芯片被唤醒，**程序从睡眠的位置继续执行**。

2.4.1.2 Normal Timer 唤醒

测试目的：

验证 DeepSleep 状态下使用 APB 普通外设 Timer 唤醒功能是否正常。

测试预期：

APB 普通外设 Timer 可以正常将芯片从 DeepSleep 状态下唤醒。

测试现象：

输入‘B’命令，可以从 Log 看到芯片进入低功耗状态，并在 2s 左右成功唤醒：

```
bTest Normal Timer Wakeup..
Timer0 ISR in..
Timer0 interrupt event detected.
Timer0 wakeup event detected.
Successfully waked up by Normal Timer!

+---------------------------------------------------------------+
|    Press key to test specific function:                       |
|                                                               |
|    Input 'A'    32K LP Timer Wakeup.                          |
|    Input 'B'    Normal Timer Wakeup.                          |
|    Input 'C'    GPIO Wakeup.                                  |
|    Press ESC key to back to the top level case list.          |
+---------------------------------------------------------------+
```

测试分析：

程序配置普通的 Timer0（位于 APB）唤醒时间为 2 秒，然后芯片将进入 DeepSleep 状态，当 Timer 超时时间到达后，芯片被唤醒，**程序从睡眠的位置继续执行**。

2.4.1.3 GPIO 唤醒

测试目的：

验证 DeepSleep 状态下使用普通 GPIO 唤醒功能是否正常。

测试预期：

普通 GPIO 可以正常将芯片从 DeepSleep 状态下唤醒。

测试现象：

输入‘C’命令，可以从 Log 看到芯片进入低功耗状态，此时按一下 EVB 底板上三个按键的任意一个（KEY1、KEY2、WKUP），发现芯片被唤醒：

```
cTest GPIO Wakeup..
GPIO ISR in..
Successfully waked up by GPIO!

+---------------------------------------------------------------+
|    Press key to test specific function:                       |
|                                                               |
|    Input 'A'    32K LP Timer Wakeup.                          |
|    Input 'B'    Normal Timer Wakeup.                          |
|    Input 'C'    GPIO Wakeup.                                  |
|    Press ESC key to back to the top level case list.          |
+---------------------------------------------------------------+
```

测试分析：

程序将 3 个 GPIO 配置为唤醒源（P04/P05/P56），唤醒电平为低电平唤醒，然后芯片将进入 DeepSleep 状态，当上述任意一个 GPIO 电平拉低后，芯片被唤醒，**程序从睡眠的位置继续执行**。

2.4.2 进入 Standby Mode 1 模式并唤醒

在主菜单下，输入‘2’命令 进入 Subcase 菜单：

```
2
+-----------------------------------------------------------------+
|    Press key to test specific function:                         |
|                                                                 |
|    Input 'A'    32K LP Timer Wakeup from standby mode 1.        |
|    Input 'B'    GPIO Wakeup from standby mode 1.                |
|    Press ESC key to back to the top level case list.            |
+-----------------------------------------------------------------+
```

2.4.2.1 32K LP Timer 唤醒

测试目的：

验证 Standby Mode 1 状态下使用 32K LP Timer 唤醒功能是否正常。

测试预期：

LP Timer 可以正常将芯片从 Standby Mode 1 状态下唤醒。

测试现象：

输入‘A’命令，可以从 Log 看到芯片进入低功耗状态，并在 2s 左右成功唤醒：

```
aTest LP Timer Wakeup from standby mode 1..
STANDBY ISR in..
SoC is waked up from standby mode 1!
Wake up source is LP Timer!
CPU @ 64000000Hz

+-----------------------------------------------------------------+
|             PAN1080 LowPower Sample Code.                       |
+-----------------------------------------------------------------+
|    press key to start test                                      |
|    Input'0'   LP Sleep Mode Test                                |
|    Input'1'   LP DeepSleep Mode Test                            |
|    Input'2'   LP Standby Mode 1 Test                            |
|    Input'3'   LP Standby Mode 0 Test                            |
+-----------------------------------------------------------------+
```

测试分析：

程序配置 LP Timer 唤醒时间为 2 秒，然后芯片将进入 Standby Mode 1 状态，当 LP Timer 超时时间到达后，芯片被唤醒**并复位，程序从头开始执行**。

2.4.2.2 GPIO 唤醒

测试目的：

验证 Standby Mode 1 状态下使用普通 GPIO 唤醒功能是否正常。

测试预期：

普通 GPIO 可以正常将芯片从 Standby Mode 1 状态下唤醒。

测试现象：

输入‘B’命令，可以从 Log 看到芯片进入低功耗状态，此时按一下 EVB 底板上三个按键的任意一个（KEY1、KEY2、WKUP），发现芯片被唤醒：

```
bTest GPIO Wakeup from standby mode 1..
STANDBY ISR in..
SoC is waked up from standby mode 1!
GPIO ISR in..
CPU @ 64000000Hz

+-----------------------------------------------------------------+
|             PAN1080 LowPower Sample Code.                       |
+-----------------------------------------------------------------+
|    press key to start test                                      |
|    Input'0'   LP Sleep Mode Test                                |
|    Input'1'   LP DeepSleep Mode Test                            |
|    Input'2'   LP Standby Mode 1 Test                            |
|    Input'3'   LP Standby Mode 0 Test                            |
+-----------------------------------------------------------------+
```

测试分析：

程序将 3 个 GPIO 配置为唤醒源（P04/P05/P56），唤醒电平为低电平唤醒，然后芯片将进入 DeepSleep 状态，当上述任意一个 GPIO 电平拉低后，芯片被唤醒**并复位，程序从头开始执行**。

2.4.3 进入 Standby Mode 0 模式并唤醒

在主菜单下，输入‘3’命令 进入 Subcase 菜单：

```
3
+-----------------------------------------------------------------+
|    Press key to test specific function:                         |
|                                                                 |
|    Input 'A'    P56 Wakeup from standby mode 0.                 |
|    Press ESC key to back to the top level case list.            |
+-----------------------------------------------------------------+
```

2.4.3.1 P56 唤醒

测试目的：

验证 Standby Mode 0 状态下使用特殊的 GPIO P56 唤醒功能是否正常。

测试预期：

GPIO P56 可以正常将芯片从 Standby Mode 0 状态下唤醒。

测试现象：

输入‘A’命令，可以从 Log 看到芯片进入低功耗状态，此时按一下 EVB 底板上的 WKUP 按键（对应芯片 P56 引脚），发现芯片被唤醒：

```
aTest P56 Wakeup from standby mode 0..
STANDBY ISR in..
SoC is waked up from standby mode 0!
Wake up source is P56!
CPU @ 64000000Hz

+-----------------------------------------------------------------+
|             PAN1080 LowPower Sample Code.                       |
+-----------------------------------------------------------------+
|    press key to start test                                      |
|    Input'0'   LP Sleep Mode Test                                |
|    Input'1'   LP DeepSleep Mode Test                            |
|    Input'2'   LP Standby Mode 1 Test                            |
|    Input'3'   LP Standby Mode 0 Test                            |
+-----------------------------------------------------------------+
```

测试分析：

Standby Mode 0 下，芯片大部分区域掉电（包括普通 GPIO 与），此时仅支持 P56 引脚唤醒。程序将 GPIO P56 配置为唤醒源，唤醒电平为低电平唤醒，然后芯片将进入 Standby Mode 0 状态，当 P56 拉低后，芯片被唤醒**并复位，程序从头开始执行**。

2.5 低功耗状态下底电流测试

底电流测试需要将一些 IO 配置为模拟输入态（高阻，默认状态）以防止漏电，因此 UART 模块不可以使能，不宜再使用 UART 通信的方式进行交互测试。

为此，程序中提供了一些专门用于测试底电流的宏开关（位于 common.h），可以用来分别测试 DeepSleep / StandbyMode1 / StandbyMode0 等低功耗状态下的底电流。

2.5.1 测试 DeepSleep 状态下底电流

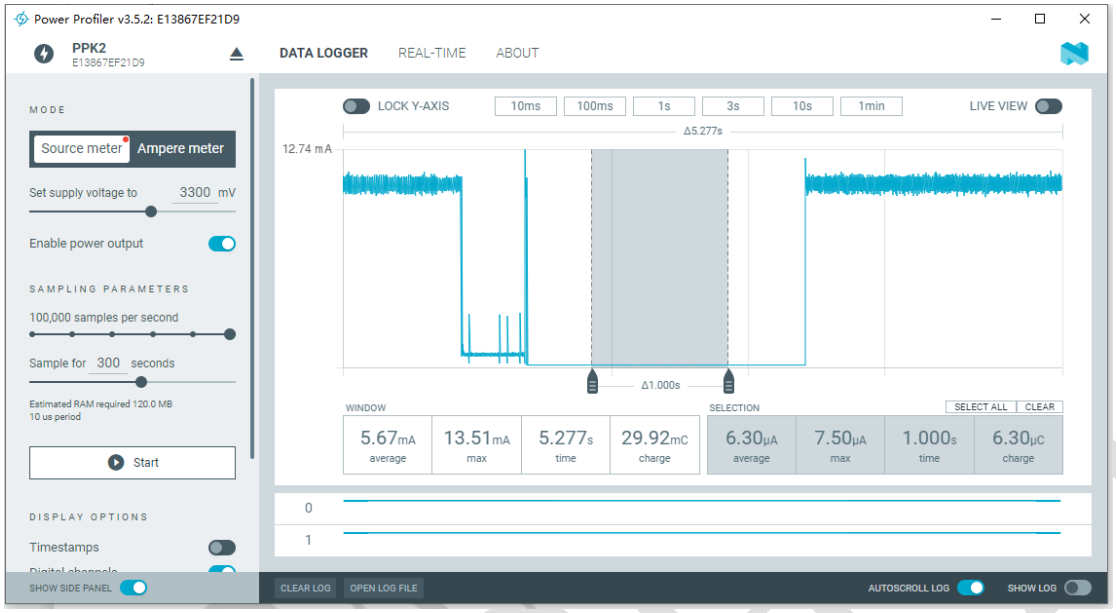
1. 在 Keil 中打开工程下的 common.h 文件，将第 29 行名为“TEST_POWER_CONSUMPTION”的宏定义取消注释；
2. 修改第 38 行，将 TEST_CURRENT_CASE 的值修改为 TEST_DEEPSLEEP_CURRENT；

修改后如下：

```
27  * Enable this macro for power (current) consumption test
28  */
29  #define TEST_POWER_CONSUMPTION
30
31  #define TEST_DEEPSLEEP_CURRENT      1
32  #define TEST_STANDBY_M1_CURRENT     2
33  #define TEST_STANDBY_M0_CURRENT     3
34
35  #ifdef TEST_POWER_CONSUMPTION
36
37  /* Change this macro for specific current test case */
38  #define TEST_CURRENT_CASE           TEST_DEEPSLEEP_CURRENT
39
```

修改完成后，程序将会直接在 main()函数比较靠前的位置直接执行 LP_TestLpTimerWakeup()函数，其行为是进入 DeepSleep 模式后，2s 后被 LP Timer 唤醒。

将修改后的程序烧录至芯片，然后拔掉芯片的所有接线，使用电流测量工具（如 PPK2）测量底电流：



从上图中可以看到，这颗芯片在 DeepSleep 状态下，1s 内的平均底电流约为 6.3uA 左右。

2.5.2 测试 Standby Mode 1 状态下底电流

1. 在 Keil 中打开工程下的 common.h 文件，将第 29 行名为“TEST_POWER_CONSUMPTION”的宏定义取消注释；
2. 修改第 38 行，将 TEST_CURRENT_CASE 的值修改为 TEST_STANDBY_M1_CURRENT；

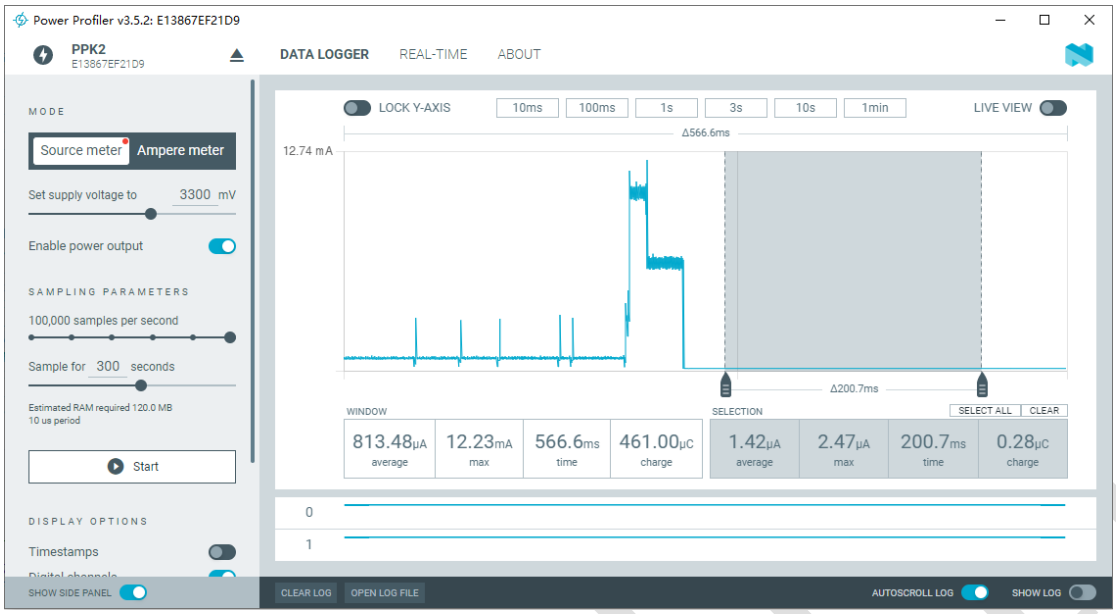
修改后如下：

```
26  /*
27   * Enable this macro for power (current) consumption test
28   */
29  #define TEST_POWER_CONSUMPTION
30
31  #define TEST_DEEPSLEEP_CURRENT      1
32  #define TEST_STANDBY_M1_CURRENT     2
33  #define TEST_STANDBY_M0_CURRENT     3
34
35  #ifdef TEST_POWER_CONSUMPTION
36
37  /* Change this macro for specific current test case */
38  #define TEST_CURRENT_CASE           TEST_STANDBY_M1_CURRENT
39
```

修改完成后，程序将会直接在 main()函数比较靠前的位置直接执行一个名为 LP_TestGpioWakeupStandbyM1()函数，其行为是进入 Standby Mode 1 模式后，等待 GPIO 唤醒。

将修改后的程序烧录至芯片，然后拔掉芯片的所有接线，使用电流测量工具（如 PPK2）测

量底电流：



从上图中可以看到，这颗芯片在 Standby Mode1 状态下，200ms 内的平均底电流约为 2.5uA 左右。

2.5.3 测试 Standby Mode 0 状态下底电流

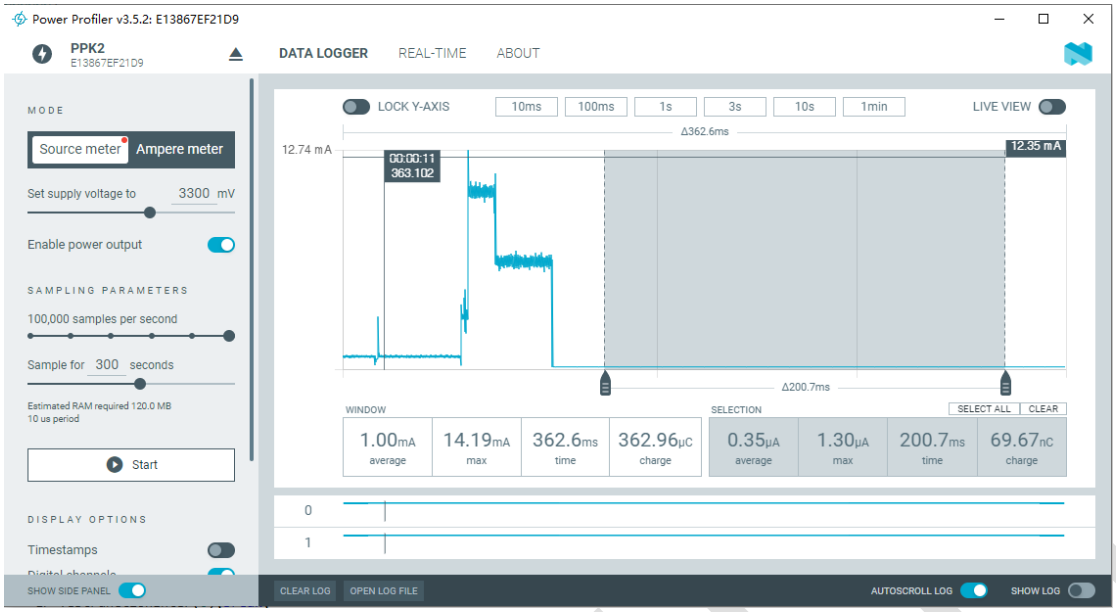
1. 在 Keil 中打开工程下的 common.h 文件，将第 29 行名为“TEST_POWER_CONSUMPTION”的宏定义取消注释；
2. 修改第 38 行，将 TEST_CURRENT_CASE 的值修改为 TEST_STANDBY_M0_CURRENT；

修改后如下：

```
26 /*
27  * Enable this macro for power (current) consumption test
28  */
29 #define TEST_POWER_CONSUMPTION
30
31 #define TEST_DEEPSLEEP_CURRENT      1
32 #define TEST_STANDBY_M1_CURRENT     2
33 #define TEST_STANDBY_M0_CURRENT     3
34
35 #ifdef TEST_POWER_CONSUMPTION
36
37 /* Change this macro for specific current test case */
38 #define TEST_CURRENT_CASE           TEST_STANDBY_M0_CURRENT
```

修改完成后，程序将会直接在 main()函数比较靠前的位置直接执行一个名为 LP_TestGpioWakeupStandbyM0()函数，其行为是进入 Standby Mode 0 模式后，等待 P56 唤醒。

将修改后的程序烧录至芯片，然后拔掉芯片的所有接线，使用电流测量工具（如 PPK2）测量底电流：



从上图中可以看到，这颗芯片在 Standby Mode 0 状态下，200ms 内的平均底电流约为 350nA 左右。

第3章 使用注意事项

1、 当希望使用 GPIO 唤醒系统时，应确保进入低功耗前相应的 GPIO 电平状态为非唤醒电平，例如，若配置某个 IO 引脚为低电平唤醒，则应保证进入低功耗前此 IO 的输入为高电平。

2、 若发现进入低功耗后底电流偏大，则应考虑检查芯片各个模块状态（特别是 GPIO），看是否有 IO 漏电发生，例如，UART Rx 功能被开启，但是进入低功耗前此引脚为浮空状态，则会导致低功耗状态下底电流变大。