NANO低功耗 Application Note & Sample Code

Application Note for 32-bit NuMicro® Family

Rev.1.13 — Jan.18, 2014

Document Information

Abstract	该文档介绍nano中，怎样调进入超低功耗模式时的功耗.
Apply to	Nano Series.

Table of Contents

1 介绍

Nano系列是以Cotex-m0为内核的超低功耗MCU。进入深度睡眠模式时，所有IP都关闭的情况下的功耗为1uA；RTC和段式LCD屏都工作的情况下功耗10uA左右。客户的系统中往往不止MCU一颗IC，怎样让系统进入深度睡眠模式时的功耗最低？一直是客户问的最多的问题，也是问题最多的地方。本文介绍一些原则和调试的方法，希望对大家有帮助。

1.1 寄存器

涉及到的寄存器包括：

- PWRCTL、AHBCLK 和 APBCLK 寄存器。
  - PWRCTL 查看内部 10K(LIRC)和外部 32K(LXT)是否关闭，如果不需要使用就关闭。这两个晶振硬件不会帮忙关闭，需要软件自行关闭。
  - AHBCLK 和 APBCLK 中查看开启的 IP，掉电模式下不需要工作的 IP，把相应的时钟也关闭（其实只要关闭模拟外设的时钟就可以）
- GPIO 中的 PMD（模式寄存器），OFFD（关闭数字通路寄存器），PUEN（使能内部上拉）寄存器。
- 系统中的 IO 功能设定寄存器 MFP
- CONFIG0 寄存器

1.2 原则

- 使用 Nu-Link 下载/调试/烧录 m0 之后，m0 会进入 debug 模式，该模式下 m0 无法进入 power down。只有给 m0 断电，m0 才能退出 debug 模式。所以烧录之后要先断电再上电才能看到 power down 的功耗。
- 进入 power down 时 NANO100 系列 LDO 引脚的电压会从 1.8V 变成 1.6V。
  - 如果 LDO 电压变成 1.6V 才说明 NANO 进入了低功耗模式。
  - 如果量到 1.8V，则表示或者烧录之后没有断电，或者是进入低功耗的函数不对，或者进入了低功耗但是系统又被唤醒了。
- 进入 power down 时 NANO1x2 系列 LDO 引脚的电压有 1.8V 和 1.6V 两种选择。
  - LDO_CTL 寄存器控制进入 power down 时 LDO 引脚的电压
- 不用的引脚统统设为 GPIO(不分封装)，INPUT 模式，并使能内部上拉(PUEN)；或者设为 OUTPUT 模式，并输出低电平。不分封装的意思是，不论 48/64/100 还是 128pin 封装，PUEN 寄存器 bit[15:0]都要写‘1’。例如：NANO130RxxBN 是 64pin 封装的，PC.4 脚没有出来，但是 PC->PUEN bit[4]还是要写‘1’。
- 状态需要保持的引脚（保持输出低电平/高电平，或者保持输入低电平/高电平），切成 GPIO 功能，模式维持不变，同时不要使能内部上拉。
  - 输入的引脚要特别小心，确认输入的电压是 0 或者 VDD 不能是中间的其它值，否则会漏电。如果切成输入的引脚输入的电压不确定，需要关闭数字通路(OFFD)

- 晶振引脚保持配置为晶振模式，即使没有外接晶振。在 power down 的时候都要设为晶振模式，并且不能打开内部上拉。但是如果外部没有接晶振，一定不能使能晶振（PWRCON），否则会漏电。
- ICE_DAT 和 ICE_CLK 脚切为 GPIO 功能，INPUT 模式，并打开内部上拉。
- 外接的段式 LCD 屏，power down 时需要保持显示的话，LCD 引脚需要保持 LCD 模式，并关闭相应引脚的数字通路。但是如果 power down 时不需要屏继续显示，相应 COM/SEG/DH1/DH2/ LCD_V1/LCD_V2/LCD_V3 引脚需要设为 GPIO 功能，INPUT 模式，并关闭相应引脚的数字通路。
  - 段式屏如果使用 C-Type 模式，功耗会比 R-Type 模式大约高 5～6uA（这个值跟 R-type 阶梯电阻，C-Type 充电频率有关）
    - C-Type 的好处是当电池没电时，可以将电压稳住，屏显示不会模糊，但是比较耗电
    - R-Type 芯片内部带 200K 欧姆的阶梯电阻，如果屏的驱动电流允许更小，可以外接 1M 这样漏电更小。
    - 在屏允许的范围内将 LCD 工作频率尽量调低（LCD_CTL 寄存器的 FREQ）
- 模拟外设需要软件自行将功能关闭。例如:ADC 需要将 ADCR 寄存器的 ADEN 清 0。并且将模拟引脚设为输入模式(PMD)，切成 GPIO 功能(MFP)，并关闭数字通路(OFFD)。
- 内部 10K 和外部 32K，power down 时不需要的话，软件应该自行关闭。
- 关闭片外 IC 的电源或者让其进入 power down。与片外 IC 相连的 MCU 引脚处理原则如下
  - 引脚是输出状态的，切成 GPIO 功能之后输出低电平。例如 SPI_CS, SPI_CLK，SPI_MISO 和 SPI_MOSI 4 根引脚，切成 GPIO 功能，输出模式，并输出低电平。
  - 需要片外 IC 通过 GPIO 唤醒 NANO 的，该 GPIO 引脚保持输入模式不要动，也不要打开内部上拉(PUEN)。
  - 如果片外 IC 的电源是关闭的，需要把与之相连的引脚都切成 GPIO 功能，并输出 0
- 检查板子上所有接地的部分，是否有可能从电源到地形成通路。如果必须形成通路，尽量把电阻调大一点，这样漏电会小一些
- 检查 CONFIG0 寄存器的设定
  - 特别是 CFOSC 栏位(CPU clock source select after reset)的值，防止没接外部晶振可是这里使能了外部晶振。
  - 另外查看 CBS（boot select）栏位，确认程序可以启动起来。经常发生程序烧录在 APROM 里面可是选择的从 LDROM 启动，而 LDROM 里面又没有程序可以跳到 APROM 里面，所以导致 APROM 里面的程序实际上没有被执行到。这里大家经常遇到的问题是在 keil 下 debug 的时候程序 run 的挺好，上电 run 反而 run 不起来，这是因为 keil 发现你的程序烧录在 APROM 里，会帮忙跳到 APROM 执行。

请在 WFI 指令之前设定断电，查看寄存器的设定是否如预期，涉及的寄存器如 1.1 节所述。

1.3 NANO 的行为

进入power down的时候，MCU会帮忙关闭HIRC（内部高频晶振）和HXT（外部高频晶振）两

个高频晶振，软件不要去控制这两个晶振。外部32K和内部10K需要软件自己关闭。

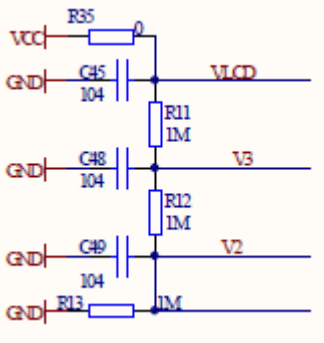
这就导致使用HIRC和HXT为时钟源的外设在进入power down的时候会自动停止工作，但是使用外部32K或者内部10K作为时钟源的外设需由软件决定在系统进入power down的时候是否仍然工作。例如：系统进入power down的时候，RTC和WDT仍然工作，这就需要不能关闭32K和10K晶振。

但是模拟外设是个特例，因为模拟电路工作不需要时钟。所以如果模拟外设功能使能，MCU虽然将晶振关闭，但是模拟电路部分仍会耗电。例如：ADC，系统进入power down之前需要写控制寄存器ADCR->ADEN将ADC功能关闭，否则会漏电

如果进入power down的时候如果引脚设定有问题，NANO中每个IO脚漏电大概20uA，这也是一个线索。

1.4 漏电电路

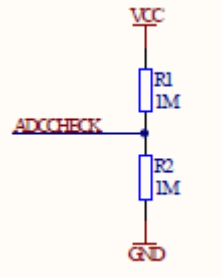
1.4.1 电路一



该电路从VCC->R35->R11->R12->R13形成一个通路。假设VCC=3.3V，这里漏电3.3/3 = 1.1uA。

这个是LCD的驱动电路，考虑到驱动能力，加1M可能已经差不多了。大家明白这里会耗电1.1uA就OK了。

1.4.2 电路二



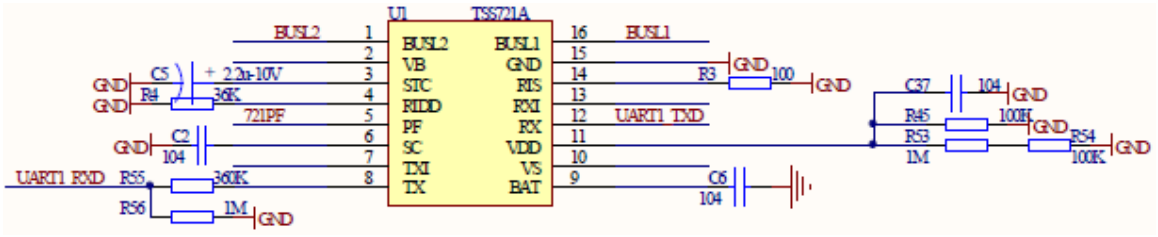
该电路从VCC->R1->R2形成通路，假设VCC=3.3V，这里漏电3.3/2 = 1.65uA。

该电路是测量电源电压的电路，ADCCHECK引脚接到ADC0模拟输入引脚，power down的时候需要将ADCCHECK引脚切为输入模式(PMD)，GPIO功能(MFP)，并关闭数字通路(OFFD)。解决

办法：

a) R1 和 R2 尽量用大一点的电阻
b) 如果 GPIO 口有剩余，将 VCC 接 GPIO，不用的时候将 IO 设成输入，关闭数字通路 OFFD，但是不要打开内部上拉。

1.4.3 电路三



该电路从UART1_RXD->R56形成通路，假设VCC=3.3V，这里漏电3.3/1 = 3.3uA。

解决办法：power down时，将UART1_RXD引脚设为INPUT模式，但是不要打开内部上拉。

1.5 常见问题

- 打开 GPIO 的内部上拉，功耗反而变高
- 打开内部上拉，功耗降低，但是耗电仍然偏高
- 功耗会飘，手靠近板子，功耗会忽高忽低
- Power down 时，LCD 和 RTC 保持显示，正常情况下功耗应该是 10uA 左右，但是测得功耗 16uA
- 大多情况下进入 power down 的功耗都是正常的 6uA 左右，但是某次进入 power down 时的功耗会高达 1mA
- 寄存器设定都正常，但是功耗还是不对

1.6 解决方法

1.6.1 打开 IO 口的内部上拉，功耗反而变高

说明引脚有下拉，这个下拉可能是内部也可能是外部。该引脚输出低电平时芯片内部是接地的，如果忘记将该引脚设为INPUT模式，内部就是下拉的，就会导致漏电。

外部有下拉不一定是明显的引脚加电阻接地，可能是外部芯片在输出低。

- 如果是 MCU 在输出低，power down 的时候不需要输出低了，就设为 INPUT 并打开内部上拉。
- 如果需要保持输出低，该引脚就维持输出不要动它。

- 如果是外部芯片拉低的，该引脚就切成 GPIO 功能，INPUT 模式，但是不要打开内部上拉。

1.6.2 打开内部上拉，功耗降低，但是耗电仍然偏高

这就要考虑两种情况

- 是否 OFFD 没有关
- 是否 ADC 没有关

OFFD 没有关，如果 LCD 使能，就会导致漏电
ADC 没有关，也会漏电

1.6.3 功耗会飘，手靠近板子，功耗会忽高忽低

这个问题的原因比较多，列出可能的3个原因

1. **可能是晶振引脚和 ICE 引脚设定不正确**

   进入power down的时候

   - 晶振引脚要维持晶振功能，不管外部是否有接晶振。但是不要打开内部上拉
   - ICE_DAT 和 ICE_CLK 引脚要设为 GPIO 功能，INPUT 模式，并打开内部上拉

   晶振引脚可能会引起大家的迷惑，
   - 如果不接晶振时，需要保持晶振模式，但是不能打开内部上拉。
   - 如果晶振引脚需要作为 GPIO 使用，power down 时，GPIO 功能不再需要的，需要切成晶振功能，但是不能打开内部上拉。
   - 如果晶振引脚需要作为 GPIO 使用，power down 时，GPIO 功能需要保持的，可以保持 GPIO。

2. **可能是晶振引脚没有接晶振但是使能了晶振(PWRCON)**
   - 解决办法就是将晶振使能关闭
3. **可能是设为 INPUT 模式的引脚输入的电压是介于 VSS 和 VDD 之间的某个电压，导致芯片内部的 MOS 漏电**
   - 解决办法就是将所有 IO 的数字通路都关闭：OFFD = 0xFFFF0000;
   - 如果电流不再飘，说明就是有引脚输入了非预想电压
   - 然后依次使能一些引脚的数字通路，最后就可以定位是哪些引脚需要关闭数字通路（OFFD）

1.6.4 Power down 时，LCD 和 RTC 保持显示，正常情况下功耗应该是 10uA 左右，但是测得功耗 16uA

如果芯片内部IO口漏电，每根IO漏电20uA左右。该板子多耗电6uA，所以一般不是芯片内部漏

电，这就需要查整个板子的电路，是否有电源到地的通路。1.4节列出了几种漏电的情况，大家可以参考看看。

1.6.5 大多情况下进入 power down 的功耗都是正常的 6uA 左右，但是某次进入 power down 时的功耗会高达 1mA

发现关闭ADC功能之后这个问题不再重现，所以锁定ADC

ADC工作频率比较低的情况下，写到ADC寄存器中的数据要等2个ADC工作时钟才会起作用，所以在进入power down时，关闭ADC时钟之前（APBCLK），需要延迟2个ADC工作时钟。不然ADC可能还没有关闭，时钟没有了，就会漏电。

1.6.6 寄存器设定都正常，但是功耗还是不对

检查引脚上是不是外接了其它仪器，查功耗的的时候，只能串电流表进来，其它的测量仪器都拿掉

1.7 调试方法

拿到板子之后，首先要确定从NANO这一路接线出来，保证只量NANO的功耗。

然后接上ICE，keil进入Debug模式，将断点放在__WFI指令这里，程序停在这里以后，逐一确认PWRCTL、AHBCLK和APBCLK以及PMD（模式寄存器），OFFD（关闭数字通路寄存器），PUEN和MFP寄存器的内容是不是和你设定的一样。设定原则就如1.2节所述。

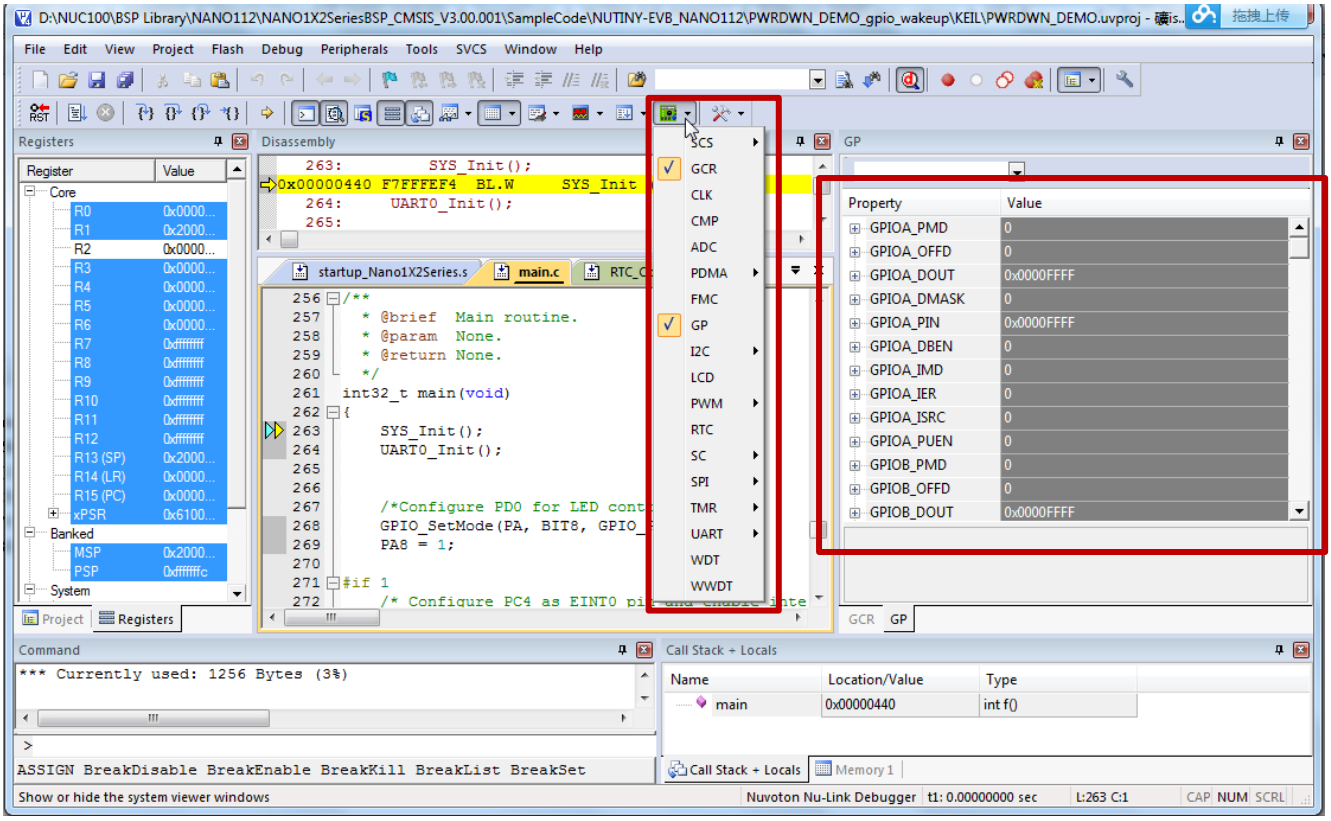


图 1-1 keil 调式额面

左边的红色框是寄存器列表，点击之后，该IP的寄存器列表会出现在右边

1.8 真实案例分析

1.8.1 客户板子进入 power down 功耗 5mA，到客户端分析原因

1) 发现客户进入低功耗时 LDO 的电压是 1.77V，明显没有进入 power down
2) 经检查代码发现是使能了 RTC 唤醒
   - RTC 的寄存器要读/写的话需要使能 ACCESS Enable 寄存器，不然一些寄存器读都是不可以的。所以进入调试模式，通过 TTR 寄存器查看是否使能了唤醒模式是不可能的。只能查 RTC 相关代码
3) 关闭 RTC 唤醒之后，此时功耗 300uA。
4) 处理进入 power down 时的 IO 状态
   - 因为客户的板子上所有的外设在 power down 时都是断电的，所以所有跟外设相连的 IO 统统设为 GPIO 功能（MFP），并输出 0。
   - 系统通过 PB.8 唤醒，该 IO 设为 INPUT 模式(PMD)，不要使能内部上拉。
   - 跟充电管理芯片相连的 IO 都设为 INPUT 模式，不使能内部上拉
   - 其他没有用到的 IO 都设为 INPUT，使能内部上拉（PUEN）

- 测量电源电压的 ADC 引脚切成 GPIO 模式，INPUT，并关闭数字通路

处理 IO 之后功耗 188uA

5) 客户板子上有一片 LDO 芯片，将 3.6V 转成 3.0V 给 MCU 使用，拿掉该 LDO 之后功耗 88uA
6) 充电电路从 VDD 到 VSS 有通路，拿掉了相连的电阻，之后功耗开始飘。低的时候到 5uA 高的时候到 20uA
   - 关闭所有 GPIO 的数字通路 OFFD，发现功耗固定在 4.2uA 不再飘
   - 依次使能 GPIO 的数字通路，发现与充电管理芯片相连的 2 跟 PB.9 和 PB.10 的数字通路一定要关闭，不然功耗就会飘
   - 测量 PB.8 和 PB.9 的波形，发现这两根引脚的电压为 2.4V，会导致漏电，关闭这两根引脚的数字通路

至此，客户整个板子在进入低功耗时耗电4.2uA

1.8.2 客户的板子放一个晚上功耗飘到了 100uA，到客户端分析原因

客户的板子上只有我们的NANO，功耗正常时2uA

经过分析发现客户没有外接高速晶振，但是使能了晶振（PWRCON）。将晶振关闭之后电流没有再飘。

2 示例代码

该板子外接段式LCD屏，进入power down的时候LCD和RTC要保持工作，另外外部还有一颗芯片需要PB.7输出低电平让其进入power down

2.1 进入低功耗函数

```
void Enter_PowerDown()
{
    UNLOCKREG();
    /* Back up original setting */
    SavePinSetting();

    /* set to INPUT mode */
    GPIOA->PMD = 0;
    GPIOB->PMD = 0x4000;/*PB.7 OUTPUT low*/
    GPIOC->PMD = 0;
    GPIOD->PMD = 0;
    GPIOE->PMD = 0;
    GPIOF->PMD = 0;
    /* Set to GPIO模式 */
    GCR->PA_L_MFP = 0x00FFFF00;
    GCR->PA_H_MFP = 0xFFFFFF00;
    GCR->PB_L_MFP = 0x0000FF00;
    GCR->PB_H_MFP = 0xFFFFFFFF;
    GCR->PC_L_MFP = 0x0000FF00;
    GCR->PC_H_MFP = 0x0F00FFFF;
    GCR->PD_L_MFP = 0x0;
    GCR->PD_H_MFP = 0x0;
    GCR->PE_L_MFP = 0x0;
    GCR->PE_H_MFP = 0x0;
    GCR->PF_L_MFP = 0x00FF00;

    /* Disable Digital path of LCD pin */
    GPIOA->OFFD = 0xFC3C0000;
    GPIOB->OFFD = 0xFF0C0000;
    GPIOC->OFFD = 0x4F0C0000;

    /* Enable GPIO pull up */
```

```
    GPIOA->PUEN = 0xFFFF;
    GPIOB->PUEN = 0xFF7F;/*PB.7 low*/
    GPIOC->PUEN = 0xFFFF;
    GPIOD->PUEN = 0xFFFF;
    GPIOE->PUEN = 0xFFFF;
    GPIOF->PUEN = 0x0033;/* exclude GPF2 and GPF3 which are HXT OUT/IN */

    SYS_SetChipClockSrc(CLK_PWRCTL_LXT_EN, 1);
    SYS_SetUpPowerDown(0);  /* Disable PDWU interrupt */
    LOCKREG();
    __WFI();      /* system really enter power down here ! */
}
```

2.2 状态保存和恢复

函数SavePinSetting();用来保存GPIO和MFP寄存器的状态，在系统唤醒之后，这些寄存器的状态需要恢复

```
__IO uint32_t  _Pin_Setting[11];       /* store Px_H_MFP and Px_L_MFP */
__IO uint32_t  _PullUp_Setting[6];     /* store GPIOx_PUEN */
__IO uint32_t  _PMD_Setting[6];        /* store GPIOx_PMD */
__IO uint32_t  _OFFD_Setting[6];       /* store GPIOx_PMD */

void SavePinSetting()
{
    /* Save Pin selection setting */
    _Pin_Setting[0] = GCR->PA_L_MFP;
    _Pin_Setting[1] = GCR->PA_H_MFP;
    _Pin_Setting[2] = GCR->PB_L_MFP;
    _Pin_Setting[3] = GCR->PB_H_MFP;
    _Pin_Setting[4] = GCR->PC_L_MFP;
    _Pin_Setting[5] = GCR->PC_H_MFP;
    _Pin_Setting[6] = GCR->PD_L_MFP;
    _Pin_Setting[7] = GCR->PD_H_MFP;
    _Pin_Setting[8] = GCR->PE_L_MFP;
    _Pin_Setting[9] = GCR->PE_H_MFP;
    _Pin_Setting[10] = GCR->PF_L_MFP;
```

```
    /* Save Pull-up setting */
    _PullUp_Setting[0] =  GPIOA->PUEN;
    _PullUp_Setting[1] =  GPIOB->PUEN;
    _PullUp_Setting[2] =  GPIOC->PUEN;
    _PullUp_Setting[3] =  GPIOD->PUEN;
    _PullUp_Setting[4] =  GPIOE->PUEN;
    _PullUp_Setting[5] =  GPIOF->PUEN;

    /* Save PMD setting */
    _PMD_Setting[0] =  GPIOA->PMD;
    _PMD_Setting[1] =  GPIOB->PMD;
    _PMD_Setting[2] =  GPIOC->PMD;
    _PMD_Setting[3] =  GPIOD->PMD;
    _PMD_Setting[4] =  GPIOE->PMD;
    _PMD_Setting[5] =  GPIOF->PMD;

    /* Save OFFD setting */
    _OFFD_Setting[0] =  GPIOA->OFFD;
    _OFFD_Setting[1] =  GPIOB->OFFD;
    _OFFD_Setting[2] =  GPIOC->OFFD;
    _OFFD_Setting[3] =  GPIOD->OFFD;
    _OFFD_Setting[4] =  GPIOE->OFFD;
    _OFFD_Setting[5] =  GPIOF->OFFD;
}

void RestorePinSetting()
{
    /* Restore Pin selection setting */
    GCR->PA_L_MFP = _Pin_Setting[0];
    GCR->PA_H_MFP = _Pin_Setting[1];
    GCR->PB_L_MFP = _Pin_Setting[2];
    GCR->PB_H_MFP = _Pin_Setting[3];
```

```
    GCR->PC_L_MFP = _Pin_Setting[4];
    GCR->PC_H_MFP = _Pin_Setting[5];
    GCR->PD_L_MFP = _Pin_Setting[6];
    GCR->PD_H_MFP = _Pin_Setting[7];
    GCR->PE_L_MFP = _Pin_Setting[8];
    GCR->PE_H_MFP = _Pin_Setting[9];
    GCR->PF_L_MFP = _Pin_Setting[10];

    /* Restore Pull-up setting */
    GPIOA->PUEN = _PullUp_Setting[0];
    GPIOB->PUEN = _PullUp_Setting[1];
    GPIOC->PUEN = _PullUp_Setting[2];
    GPIOD->PUEN = _PullUp_Setting[3];
    GPIOE->PUEN = _PullUp_Setting[4];
    GPIOF->PUEN = _PullUp_Setting[5];

   /* Restore PMD setting */
    GPIOA->PMD = _PMD_Setting[0];
    GPIOB->PMD = _PMD_Setting[1];
    GPIOC->PMD = _PMD_Setting[2];
    GPIOD->PMD = _PMD_Setting[3];
    GPIOE->PMD = _PMD_Setting[4];
    GPIOF->PMD = _PMD_Setting[5];

   /* Restore OFFD setting */
    GPIOA->OFFD = _OFFD_Setting[0];
    GPIOB->OFFD = _OFFD_Setting[1];
    GPIOC->OFFD = _OFFD_Setting[2];
    GPIOD->OFFD = _OFFD_Setting[3];
    GPIOE->OFFD = _OFFD_Setting[4];
    GPIOF->OFFD = _OFFD_Setting[5];
}
```

保存和恢复的函数将所有的寄存器都保存和恢复了，如果用户确定某个寄存器不需要保存可以将相关代码拿掉，特别在SRAM不够用的时候，buffer可以定义小一点。

Revision History

Rev.	Date	Description
1.00	1-21-2014	Initially issued.
1.10	4-11-2014	添加AHBCLK和APBCLK处理 添加一些有问题原理图分析
1.12	4-25-2014	功耗飘时多添加2种肯能原因 如果量不到LDO为1.6v原因说明

Important Notice

Nuvoton products are not designed, intended, authorized or warranted for use as components in systems or equipment intended for surgical implantation, atomic energy control instruments, airplane or spaceship instruments, transportation instruments, traffic signal instruments, combustion control instruments, or for other applications intended to support or sustain life. Furthermore, Nuvoton products are not intended for applications wherein failure of Nuvoton products could result or lead to a situation wherein personal injury, death or severe property or environmental damage could occur.
Nuvoton customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Nuvoton for any damages resulting from such improper use or sales.

Please note that all data and specifications are subject to change without notice. All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.