基于LGTSDK Builder

LGT8F690A 快速开发系列教程

第七篇：静态功耗控制

概述

本篇为系列教程的第八篇。如果需要了解教程相关的软件硬件环境，请参考本系列教程的第一篇：《LGT8F690A快速开发系列教程第一篇_急速上手》

在本系列教程的第七篇中，我们已经介绍了LGT8F690A的动态功耗控制部分。现在我们主要介绍LGT8F690A的静态功耗控制， 休眠与唤醒。

LGT8F690A支持是三种休眠模式， 分别为待机模式，省电模式和深睡眠模式。这三种休眠模式在休眠功耗，支持的唤醒源以及唤醒时间等均有差别，适用于不同的应用要求。

LGT8F690A的休眠模式，可以在配置位中设置；也可以由软件通过设置PCON寄存器的DPSM位来动态的临时改变。以下为这三种功耗模式的配置和相关特性：

PCON : DPSM	休眠模式	功能说明
00	待机模式 IDLE	待机模式首先让内核进入休眠状态，然后关闭内核时钟。系统时钟源并没有关闭，同时其他一些外设的异步时钟也没有关闭。此模式支持比较多的唤醒源， 并能够快速的唤醒。
01	省电模式 SAVE	省电模式在待机模式的基础之上，关闭了内部高频时钟源。但保留内部32KHz低功耗时钟。 此模式支持基于内部32KHz定时器/看门的定时唤醒，以及RA/B的引脚电平变化唤醒。省电模式可以在唤醒模式和低功耗上达到一个比较好的平衡。
1X	深睡眠模式 DEEP	深睡眠模式将会关闭系统所有的内部时钟。但工作于外部低速晶振的TMR1也可以继续工作，因此深睡眠模式也可以选择支持定时唤醒。另外深睡眠模式也支持RA/B引脚电平变化唤醒。此模式能够获得最低的休眠功耗，但唤醒时间也会相对慢一些。

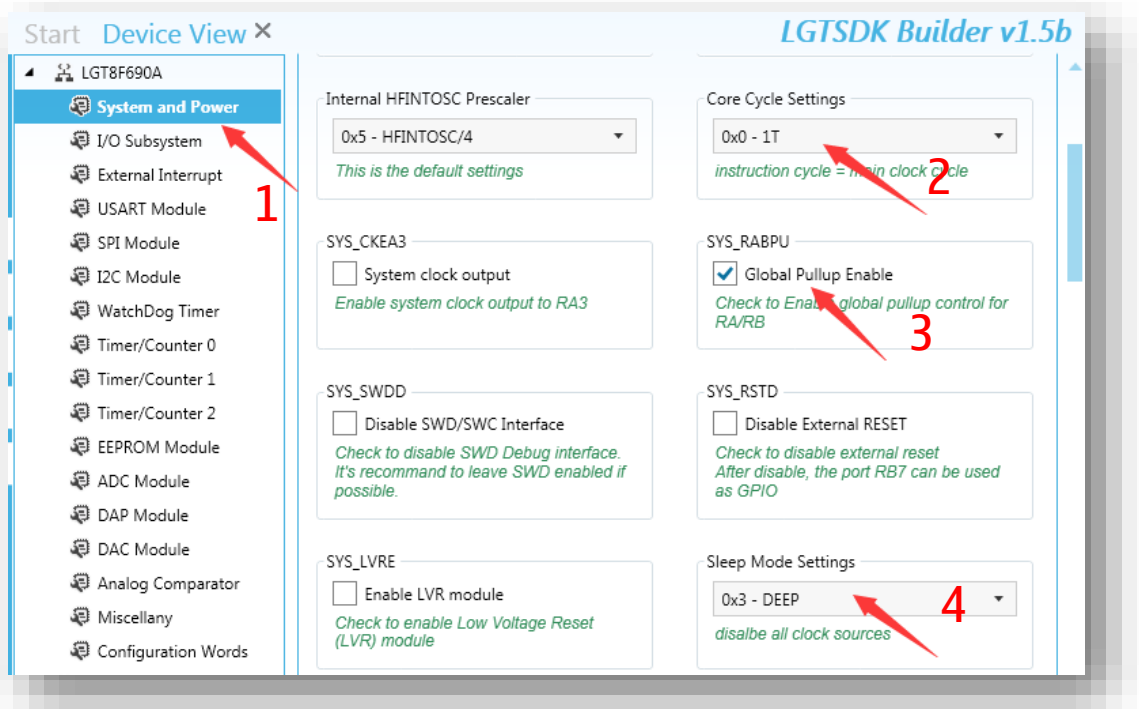
在实际应用中，在满足支持所需的唤醒源的前提下， 我们希望MCU工作在更低的功耗模式下。深睡眠模式可以支持定是唤醒，也支持外部引脚电平变化唤醒，基本上可以满足大部分应用的需求。接下来，我们先以深睡眠模式为例， 分别示例如何实现休眠以及唤醒控制。

开始工作！ 首先启动LGTSDK Builder， 新建一个工程，选择目标芯片LGT8F690A。
项目名称：lgt8f690a_dpsm

简单说明下我们将要实现的例程：系统上电后，进入正常的工作模式，我们用一个I/O， 比如RA0，输出一个固定的频率来指示系统的工作状态。使用另外一个I/O，比如RC0， 作为进入休眠模式的控制输入。当RC0拉低后，系统进入预设的休眠模式。休眠的唤醒可以使用看门狗做定时唤醒，也可以使用RA/B组I/O的电平变化唤醒。 为了演示不同模式下的最低功耗性能，我们将分两个程序，分别演示引脚电平变化唤醒和定时唤醒两种情况。

LGT8F690A的RA和RB两组I/O都可以产生引脚电平变化事件。这里，我们使用RB4/5/6/7四个I/O作为唤醒源。定时唤醒将使用看门狗定时器实现。

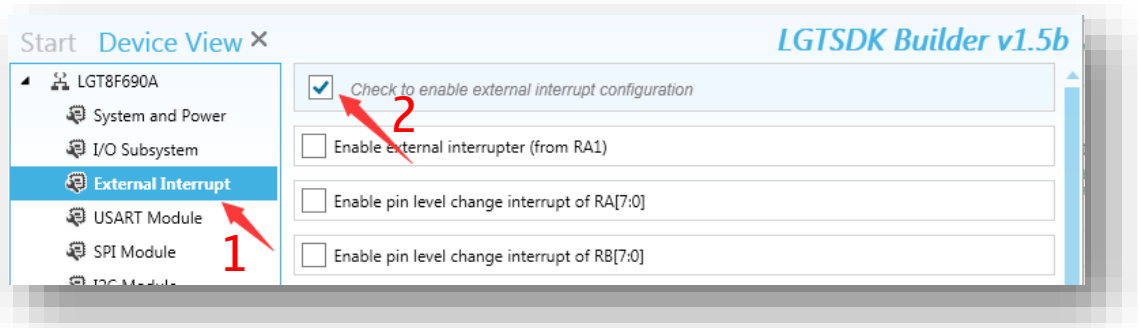
首先是设置休眠模式，这个设置在[System and Power]配置页中：



配置说明：
2： 指令周期模式这里配置为1T，选择其他配置也可以；
3： 使能全局上拉。我们需要将没有使用到的I/O设置为输入上拉，避免浮空漏电；
4： 这里是配置休眠模式，选择我们需要的深睡眠模式(DEEP)；

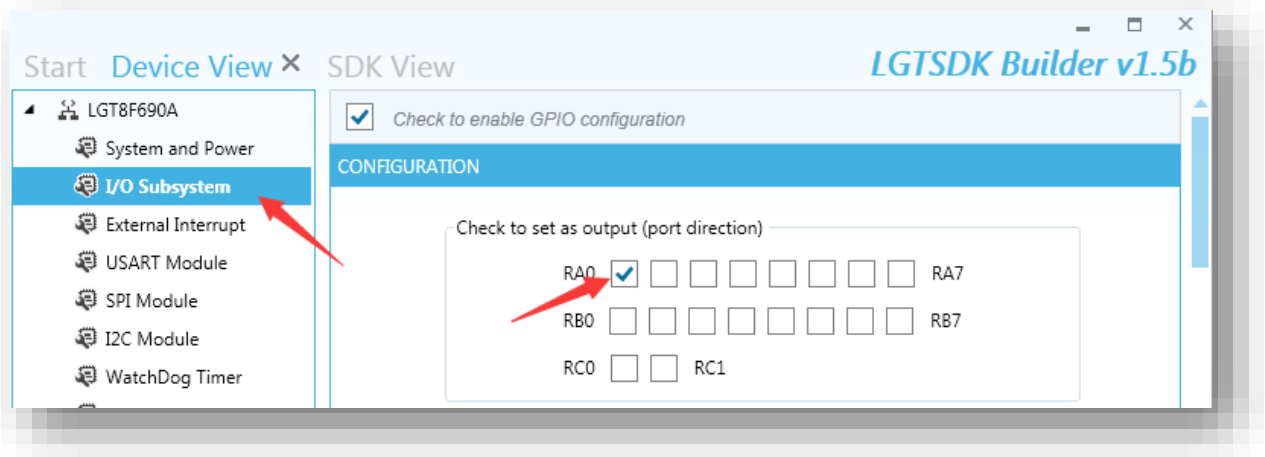
此处2/4的配置，都将产生软件代码。优先级高于配置位的设置。配置位中确保主时钟源为内部高速RC时钟即可。

接下来是唤醒源的配置。首先我们演示引脚电平变化唤醒模式。LGTSDK Builder中有关引脚电平变化的配置在外部中断相关的配置页中，我们目前不打算使用中断，因此我们将使用SDK中的接口函数使能引脚电平变化事件功能。由于引脚电平变化设置相关的接口函数在[External Interrupt]配置页中，因此，我们需要使能它，把相关函数包含到项目中：



如上图2中，勾选即可。不需要做其他配置。这样我们就将外部引脚变化相关的函数接口包含到当前工程中。

最后是I/O的配置， 我们需要用RA0作为输出工作指示。其他I/O无论是作为唤醒功能，还是没有使用到的I/O， 都保持为默认的输入状态。我们将在程序中开启他们的内部上拉电阻。



外设配置完成， 点击[Device View] 下面的 [Build] ， 生成SDK代码！

下面是我们将要使用的SDK接口函数的列表：

函数名称	功能说明	使用方法
eintEnableIOC()	使能I/O的外部引脚电平变化功能	eintEnableIOC(GPPB, P3|P4|P5|P6) 使能RB3/4/5/6的引脚电平变化功能
eintDisableIOC()	禁止I/O的外部引脚电平变化功能	eintDisableIOC(GPPB, PALL) 禁止RB组所有I/O的引脚电平变化功能
sysSleep()	在当前设置的休眠模式下进入休眠	sysSleep()
sysSleepByMode()	在指定的休眠模式下进入休眠	sysSleepByMode(SYS_SLEEP_DEEP) 进入深睡眠模式 sysSleepByMode(SYS_SLEEP_IDLE) 进入待机模式 sysSleepByMode(SYS_SLEEP_SAVE) 进入省电模式
gppPullupEnable()	使能I/O的内部上拉电阻 前提是使能全局上拉控制	gppPullupEnable(GPPA, PALL) 使能RA组I/O的上拉电阻
gppToggle()	翻转I/O的输出状态	gppToggle(GPPA, P0) 翻转RA0的输出状态
gppReadSingle()	读一个I/O的输入状态	Value = gppReadSingle(GPPC, P0) 读RC0的输入状态

休眠与唤醒

程序编译成功， 可以下载到最小开发板上测试了。

将电流表串入给芯片供电的回路上。 上电运行后， RA0上输出一个频率信号。系统正常工作。 将RC0拉低， 系统进入休眠模式。因为RC0内部有上拉电阻， 系统进入休眠后， 断开RC0的下拉， 否则会有多余漏电。 休眠后，此时的功耗应该在1uA@3.3V 左右。

在休眠模式下， 拉低RB4/5/6/7中任意一个I/O， 系统将会被唤醒， 恢复到正常的工作状态。

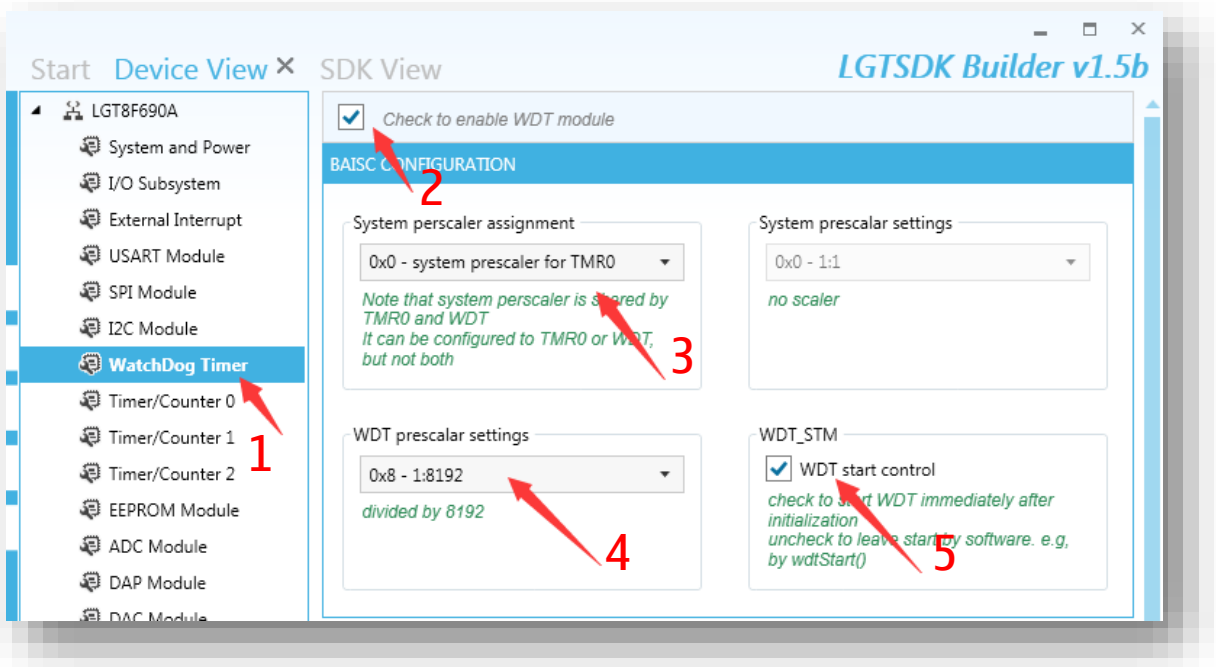
定时唤醒

下面是演示如何使用定时唤醒功能。 定时唤醒使用内部看门狗定时器实现。我们这里只需要增加看门狗的配置即可。 其他配置保持不变。这次我们将使用sysSleepByMode()进入到我们希望的省电模式(SAVE)。

这里简单介绍下LGT8F690A内部的看门狗定时器(WDT)， 这是一个16位的定时器计数器，定时器计数时钟位内部32KHz低速RC振荡器。另外， WDT与TIMER0还有一个共享的预分频器， 最大支持128分频。因此，WDT最大定时时间可达128*65536*31(us) = 260(s)。

LGT8F690A内部的32KHz 低功耗RC振荡器是没有校准的。因此频率精度较低,不能用于精确定时。

例程中，我们使用250ms左右的定时唤醒。可以不需要使用预分频器， 使用看门狗的16位计数器即可实现：



看门狗配置说明：
3： 此处不使用系统预分频器，将它分配给TMR0；
4： WDT定时设置， WDT的计数时钟是32KHz, 因此定时周期：8192*31us = 256ms；
5： 勾选此处，WDT将在初始化完成后立即运行。如果我们需要在特定的任务完成后开启WDT，可以不勾选此项。 在程序中适当的位置调用wdtStart()启动看门狗定时器。

我们也可以在配置位中使能看门狗。这里是使用软件启动看门狗，可以不用考虑配置位的设置。

外设配置完成， 点[Device View]下的[Build]， 重新生成SDK。之后来到SDK View界面。为了更清晰的展示定时唤醒功能，我们需要更新下代码：

SDK View ✕ LGTSDK Builder v1.5b

main.c ✕ init_auto.c lgt8f690a_sys.c

```
7   // Import external definitions
8   extern void init_modules(void);
9
10  int main(void)
11  {
12      // Device initialization
13      init_modules();
14
15      // enable pullup of input or unused I/O
16      gppPullupEnable(GPPA, PALL);
17      gppPullupEnable(GPPB, PALL);
18      gppPullupEnable(GPPC, (P0|P1));
19
20      // enable I/O level change funciton of RB4/5/6/7 (for wakeup)
21      eintEnableIOC(GPPB, (P4|P5|P6|P7));
22
23      // Add your code from here
24      while(1)
25      {
26          for(u8 i = 0; i < 0xff; i++)
27              gppToggle(GPPA, P0);
28
29          sysSleepByMode(SYS_SLEEP_SAVE);
30      }
31
```

使能输入I/O或者没有使用的I/O的内部上拉

使能RB4/5/6/7的电平变化唤醒功能

RA0输出一些输出工作指示

进入省电模式，等待唤醒！

代码完成，编译通过后，下载到最小开发板测试！

用示波器检测RA0的输出， 可以看到， RA0输出一串频率后，系统进入休眠。间隔260ms左右，系统被唤醒， RA0输出一串频率后再次进入休眠。

电流表上显示的电流应该在 6uA ~ 25uA @3.3V 之间来回跳动！

应用说明：

我们提供的代码和测试环境都比较理想。 软件工作后没有开启其他模拟外设。最小测试板的外部电路也非常的简单。 因此我们能够比较容易的达到最低休眠功耗。

在实际应用中， MCU的外围电路可能比较复杂， 软件在进入休眠前会开启类似ADC, 比较器或其他一些模拟外设， 这些复杂的应用环境，如果没有处理妥当，将会带来多余的耗电。因此在实际应用中，在进入休眠模式前， 请注意以下操作建议：

1. 进入休眠模式系统不会自动关闭模拟外设，比如ADC, DAC, 比较器，运放等。软件需要在进入前关闭掉所有不需要的模拟功能。
2. 特别注意在使用ADC的内部分压通道时，休眠前将分压模块关闭。
3. I/O不能处于输入浮空状态。在系统进入休眠前，需要确保I/O的配置不会带来漏电回路。
4. 唤醒源的设置。 不同模式下有不同的唤醒源， 请确认关闭掉不需要的唤醒源。