

基于LGTSDK Builder

LGT8F690A 快速开发系列教程

第三篇：时钟配置(1)

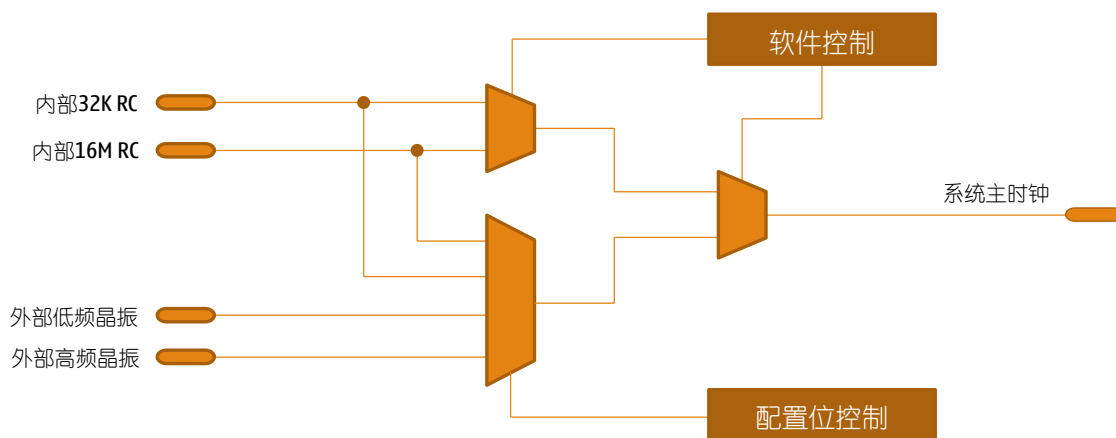


本篇为系列教程的第二篇。如果需要了解教程需要的软件硬件环境，请参考本系列教程的第一篇：《LGT8F690A快速开发系列教程第一篇_急速上手》

LGT8F690A支持内外多种时钟源。时钟源的选择主要是通过配置位控制。除配置位外，我们还可以通过软件随时强制性的将时钟源切换到内部时钟源上。

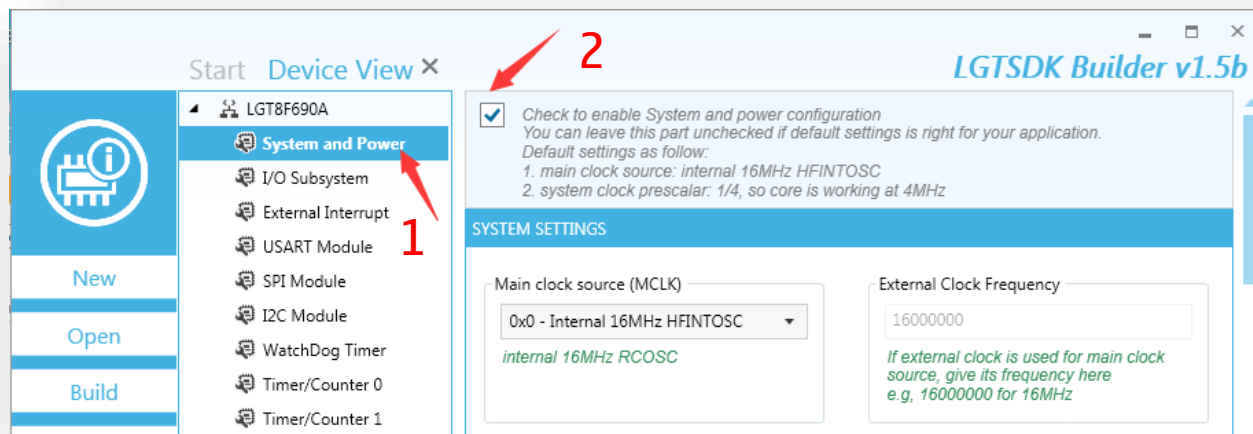
这里需要重点说明下软件控制时钟源的行为：

软件只能用来选择是用内部时钟源，还是系统配置位选择的时钟源。也就是说，软件可以强制性的将系统时钟切换到内部时钟源上。但如果要使用外部时钟源，必须通过配置位设置。下图为系统时钟的控制逻辑：



接下来，我们将以LGTSDK Builder为例，介绍如何配置选择LGT8F690A的时主时钟源和分频。在后半部分，也会介绍SDK中用于时钟控制的函数和使用方法。

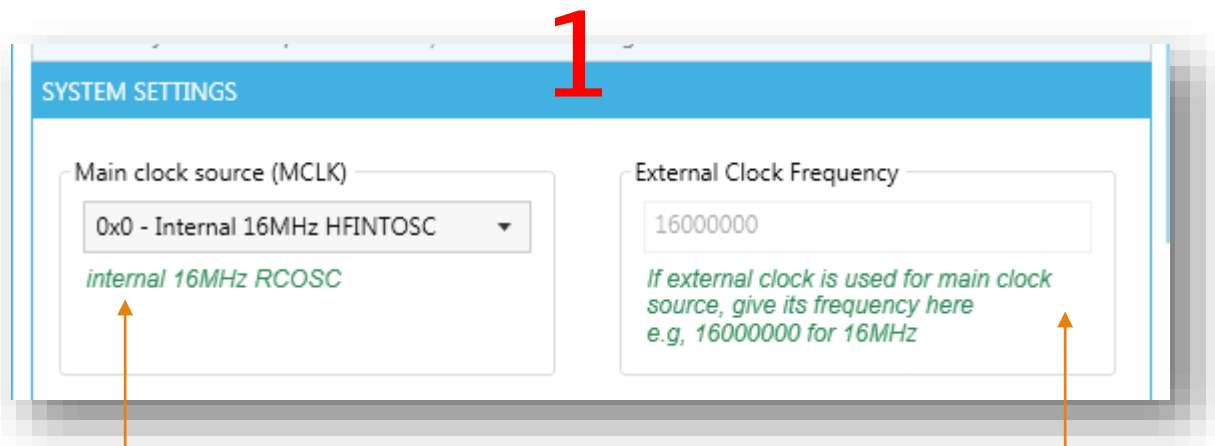
启动LGTSDK Builder，建立一个工程，工程名称：lgt8f690a_sysclk
系统时钟相关的配置，在[System and Power] 以及 [Configuration Words]两者中。



我们先关注内部时钟源的用法。芯片上电后，默认是使用内部的**16MHz RC**作为时钟源。主时钟源再经过系统时钟分频器(默认是4分频)之后，产生系统工作的时钟。

内部时钟源是芯片默认的配置，包括系统配置位，也是默认选择了内部**16MHz RC**。

下面主要介绍 [System and Power] 配置页中，和时钟有关的配置选项：

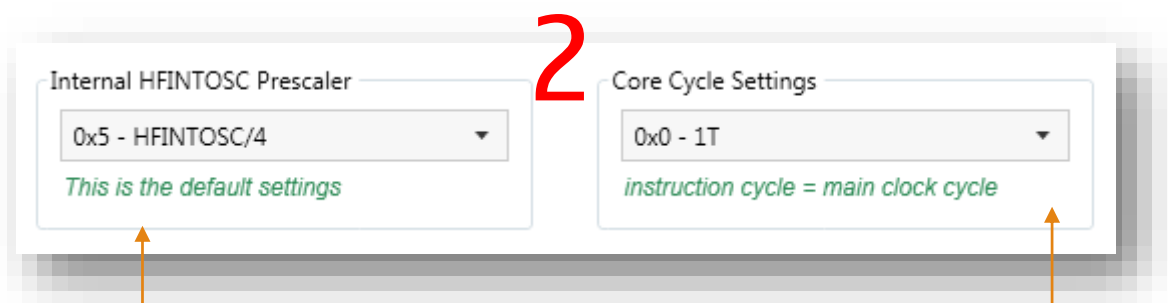


如果使用的是外部晶振，这里给出晶振的频率(单位：Hz)

主时钟源选择。

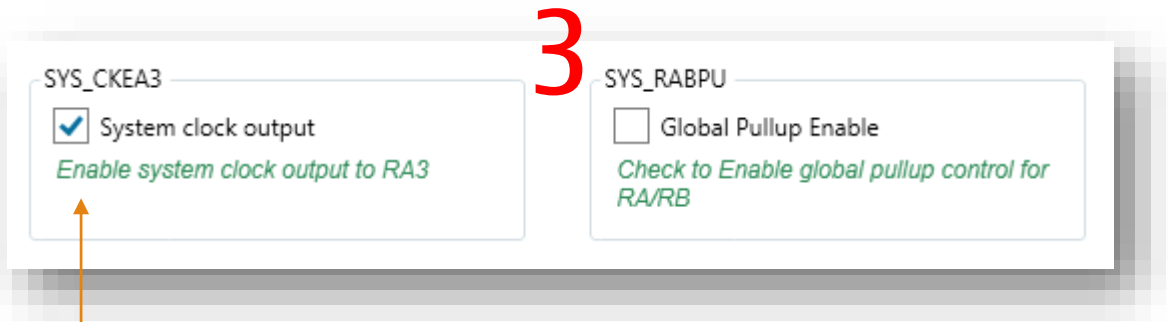
注意：只有配置位可以选择主时钟源，这里只是用于系统时间相关的计算的。比如延时函数，串口波特率计算等。我们使用默认的**16MHz**内部晶振

如上所述，此两处配置都是**SDK**用于准确计算系统时间的。并不能改变系统工作时钟。系统是时钟源的选择可以通过配置位设置，也可以通过软件切换到内部时钟源。我们这里使用默认的时钟源。因此也不需要涉及配置位的操作。



选择时钟预分频。**LGT8F690A**的预分频只对内部**16MHz RC**有效

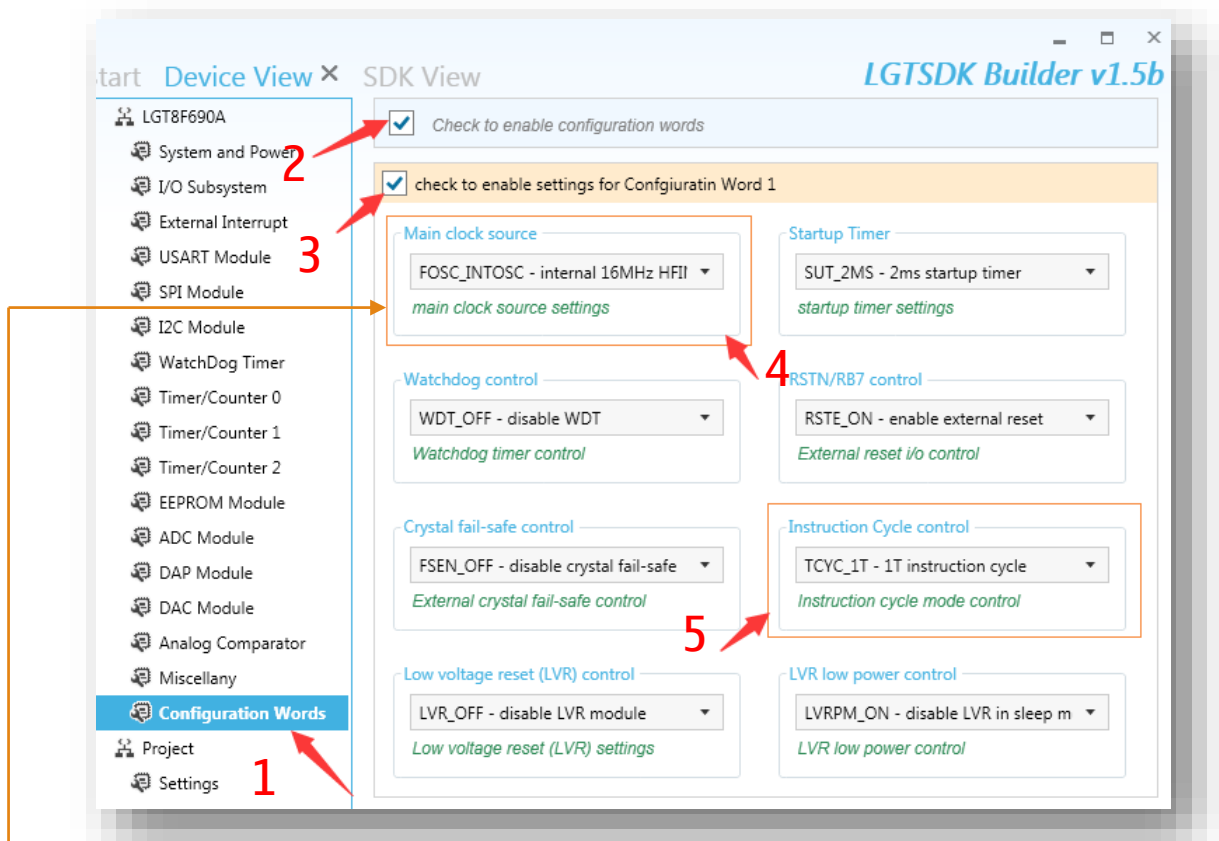
指令周期模式选择。**1T**表示一个指令周期对应一个系统时钟周期。**4T**表示一个指令周期对应4个系统时钟周期。这里我们选择**1T**模式。
注意：此处对指令周期模式的设置优先级高于配置位中的配置。



系统时钟输出到RA3。勾选后，引脚RA3上输出系统时钟。勾选此处，我们将通过示波器或者频率计检测RA3的输出，确定当前的时钟配置。

为保证芯片工作在我们需要的内部16MHz RC，我们需要确保配置位选择的主时钟源为默认的内部16MHz RC。配置位的选择可以在下载工具上配置，也可以在SDK Builder里面配置并合并到源代码中，这样下载工具可以直接从编译产生的HEX文件中读取配置信息，省去了手工设置的麻烦和可能带来的错误。

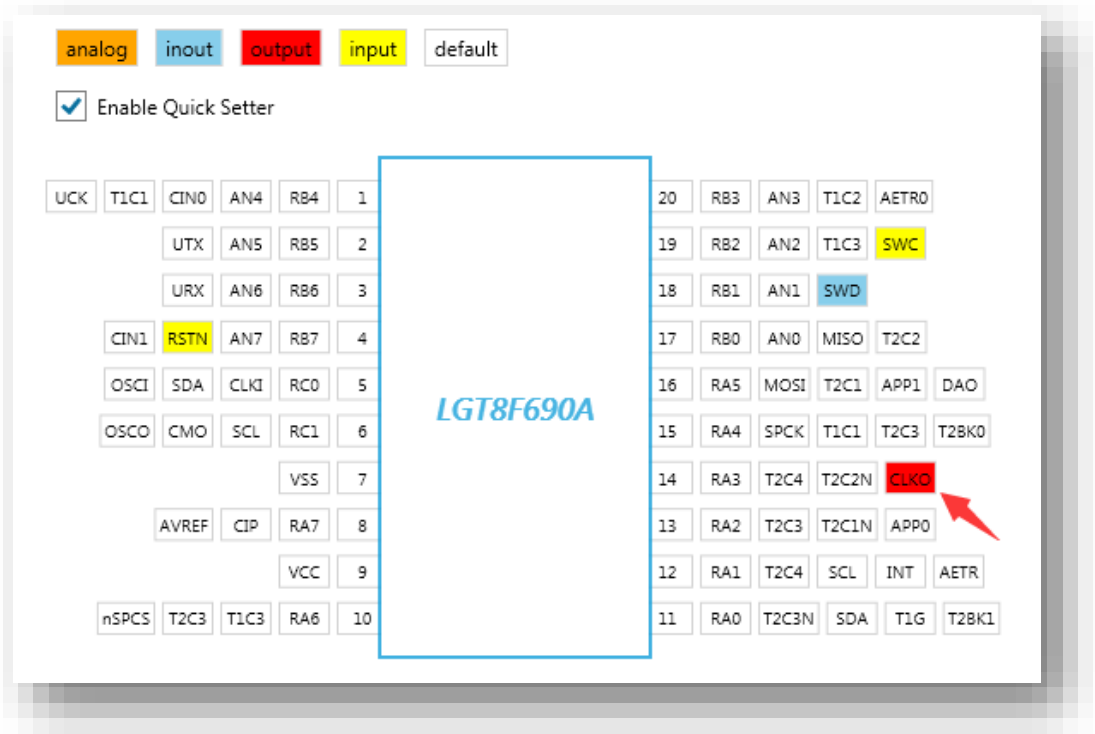
SDKBuilder中，配置位的设置在设备配置树下的 [Configuration Words]配置页中：



点1/2/3依次使能系统配置位。并确保此处主时钟源设置为默认的内部16MHz RC振荡器

配置完成！接下来点 [Device View] 下的 [Build] 按钮产生SDK 代码。

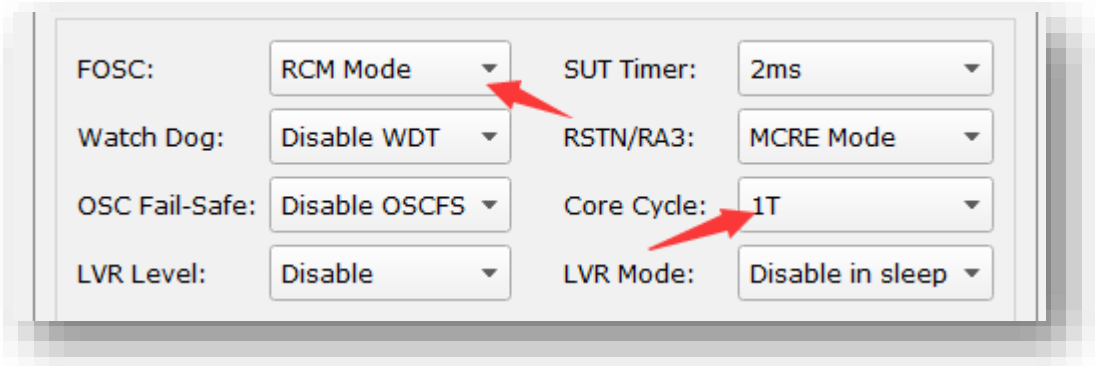
我们在配置时已经使能了系统时钟输出到RA3上。打开[PINOUT VIEW]， 应该可以看到RA3引脚此时已经标记了CLKO的输出功能：



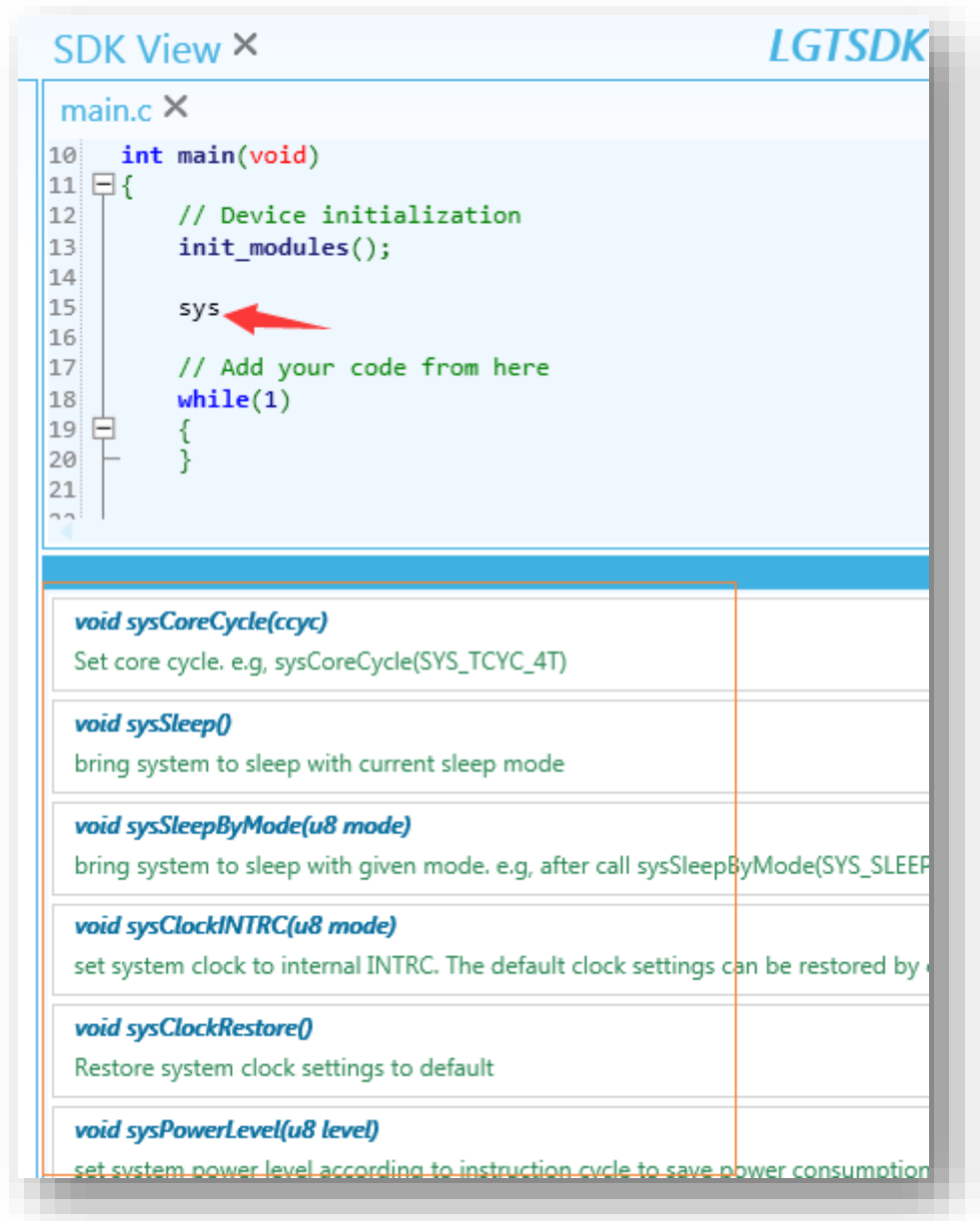
到这里，我们不用写代码，直接在 [SDK View] 下点 [Build] 编译。可以直接下载到最小开发板中测试。根据我们之前的配置，16MHz主频，默认4分频。程序下载运行后，RA3的输出频率为：

$$\text{Freq(RA3)} = 16\text{MHz}/4 = 4\text{MHz}$$

在使用LGTMix_ISP下载程序时，加载完HEX文件，可以观察配置位的变化，此时LGTMix_ISP的配置信息应该与SDKBuilder中的配置一致。可以从指令周期模式的配置确认：



接下来，我们增加代码，改变系统时钟。打开main.c，在合适的位置输入sys, 或者clock，日志窗口将会列出所有系统操作相关的SDK函数：



这里我们介绍几个系统时钟控制的相关函数。这些函数包括设置指令周期模式，设置系统时钟预分频器，使能/禁止系统时钟输出等。

函数名称	功能描述	使用方法
sysCoreCycle()	设置指令周期模式	sysCoreCycle(SYS_TCYC_4T)
sysClockINTRC()	配置系统时钟为内部时钟源	sysClockINTRC(SYS_INTRC_DIV16) 设置主时钟为内部16MHz时钟的16分频 sysClockINTRC(SYS_INTRC_RC32K) 切换主时钟为内部32KHz RC振荡器
sysClockRestore()	恢复系统时钟配置，用于在sysClockINTRC之后恢复系统时钟的默认配置	sysClockRestore()
clkoEnable()	使能系统时钟输出到RA3	clkoEnable()
clkoDisable()	禁止系统时钟输出到RA3	clkoDisable()

首先，我们实例如何使用sysClockINTRC改变主时钟预分频，代码如下：

此时，系统时钟应该为1MHz左右。

```
12 // Device initialization
13 init_modules();
14
15 sysClockINTRC(SYS_INTRC_DIV16);
16
17 // Add your code from here
18 while(1)
19 {
20
21 }
22
```

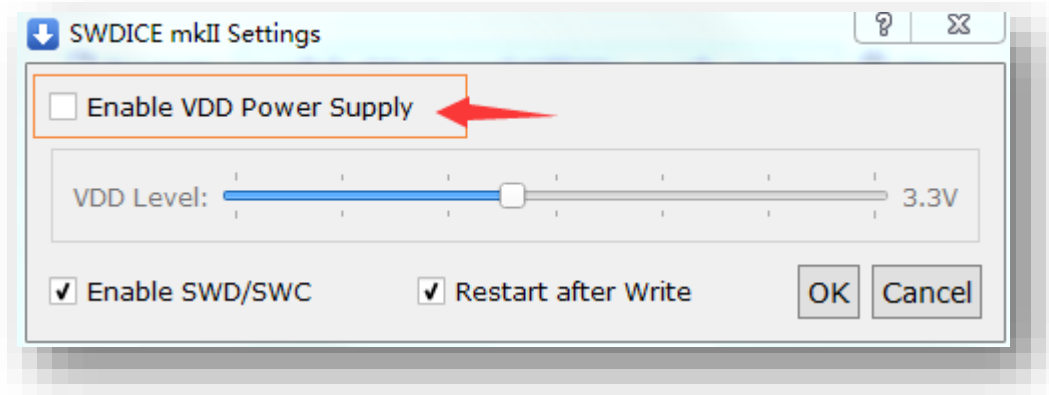
接下来，我们把系统时钟切换到内部32KHz RC上，代码如下：

此时，系统时钟应该为32KHz左右。

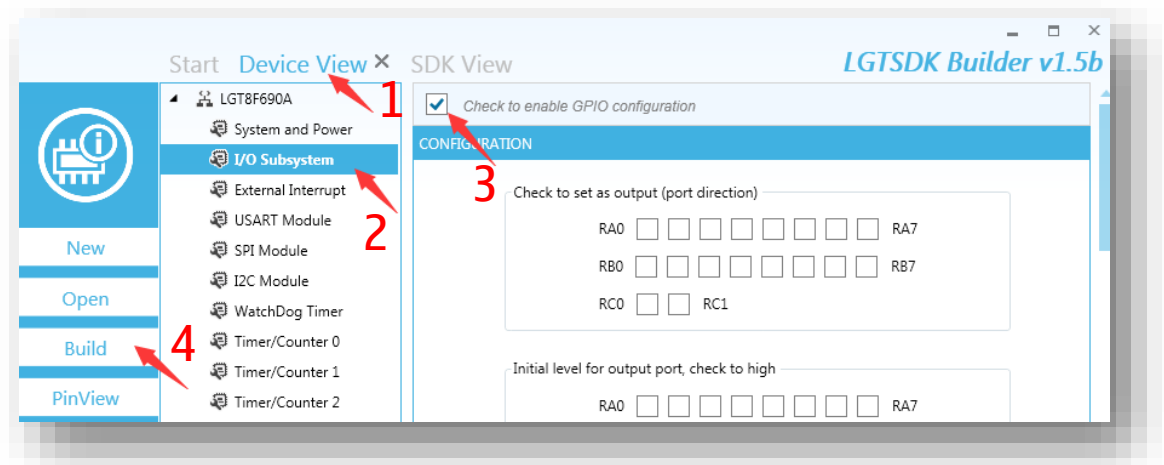
```
12 // Device initialization
13 init_modules();
14
15 sysClockINTRC(SYS_INTRC_RC32K);
16
17 // Add your code from here
18 while(1)
19 {
20
21 }
22
```

内部32KHz时钟是没有校准的。因此频率偏差可能比较大。另外内部32KHz的占空比也不是50%的。

内部时钟切换到RC32KHz后，在使用LGTMix_ISP下载程序时，可能会提示设备出错。遇到类似情况，请在LGMTmix_ISP软件上，打开Settings设置窗口，更改供电设置如下：



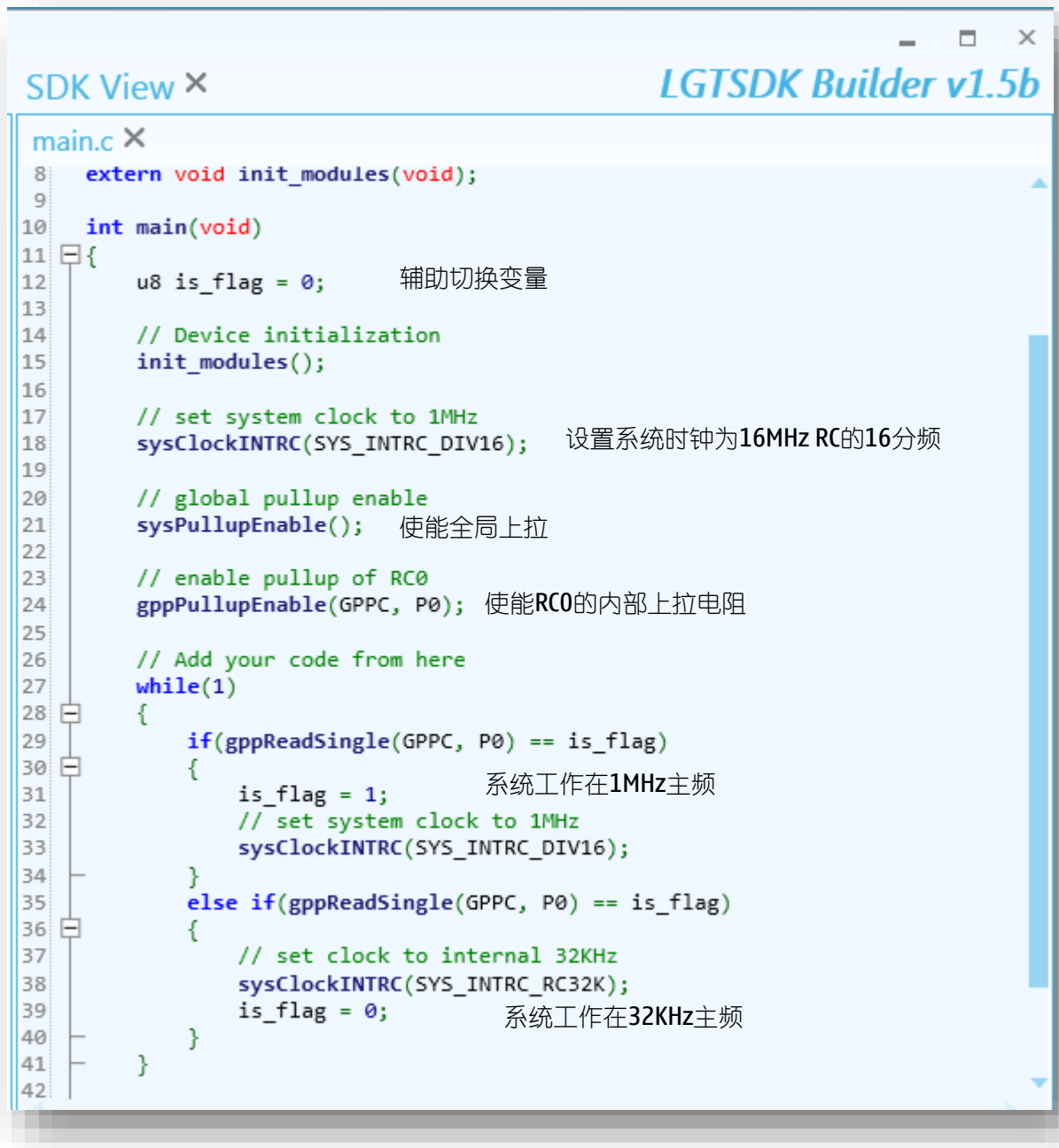
最后，我们示例下如何动态的控制时钟源之间的切换。为实现动态切换，我们需要一个输入I/O辅助触发时钟的切换。因此需要使用到I/O相关函数。为使用I/O相关函数，我们可以再次回到 [Device View] 下，通过勾选 [I/O Subsystem] 将相关SDK函数包含到工程中。步骤如下：



这里我们用RC0的输入触发时钟源切换。不要勾选I/O为输出。按照1/2/3的步骤选择。然后点步骤4，重建SDK。此时重建SDK不会更新我们修改过的主.c文件。

SDK重建后，会再次回到SDK View视图。此时可以在项目树视图下看到增加了I/O的相关文件。

下面开始编写代码。我们使用RC0的输入触发时钟切换。作为输入的I/O，我们将会开启内部上拉给他一个确定的状态。这次，我们将用函数来控制开启上拉。



```
main.c X
8   extern void init_modules(void);
9
10  int main(void)
11  {
12      u8 is_flag = 0;      辅助切换变量
13
14      // Device initialization
15      init_modules();
16
17      // set system clock to 1MHz
18      sysClockINTRC(SYS_INTRC_DIV16);    设置系统时钟为16MHz RC的16分频
19
20      // global pullup enable
21      sysPullupEnable();    使能全局上拉
22
23      // enable pullup of RC0
24      gppPullupEnable(GPPC, P0);    使能RC0的内部上拉电阻
25
26      // Add your code from here
27      while(1)
28      {
29          if(gppReadSingle(GPPC, P0) == is_flag)
30          {
31              is_flag = 1;      系统工作在1MHz主频
32              // set system clock to 1MHz
33              sysClockINTRC(SYS_INTRC_DIV16);
34          }
35          else if(gppReadSingle(GPPC, P0) == is_flag)
36          {
37              // set clock to internal 32KHz
38              sysClockINTRC(SYS_INTRC_RC32K);
39              is_flag = 0;      系统工作在32KHz主频
40          }
41      }
42  }
```

代码编写完成。编译后下载到最小开发板中测试！

通过将RC0接到低电平或者悬空，用示波器测试RA3的输出，观察系统时钟的变化！