

基于LGTSDK Builder

LGT8F690A 快速开发系列教程

第五篇: USART的使用



本篇为系列教程的第五篇。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》

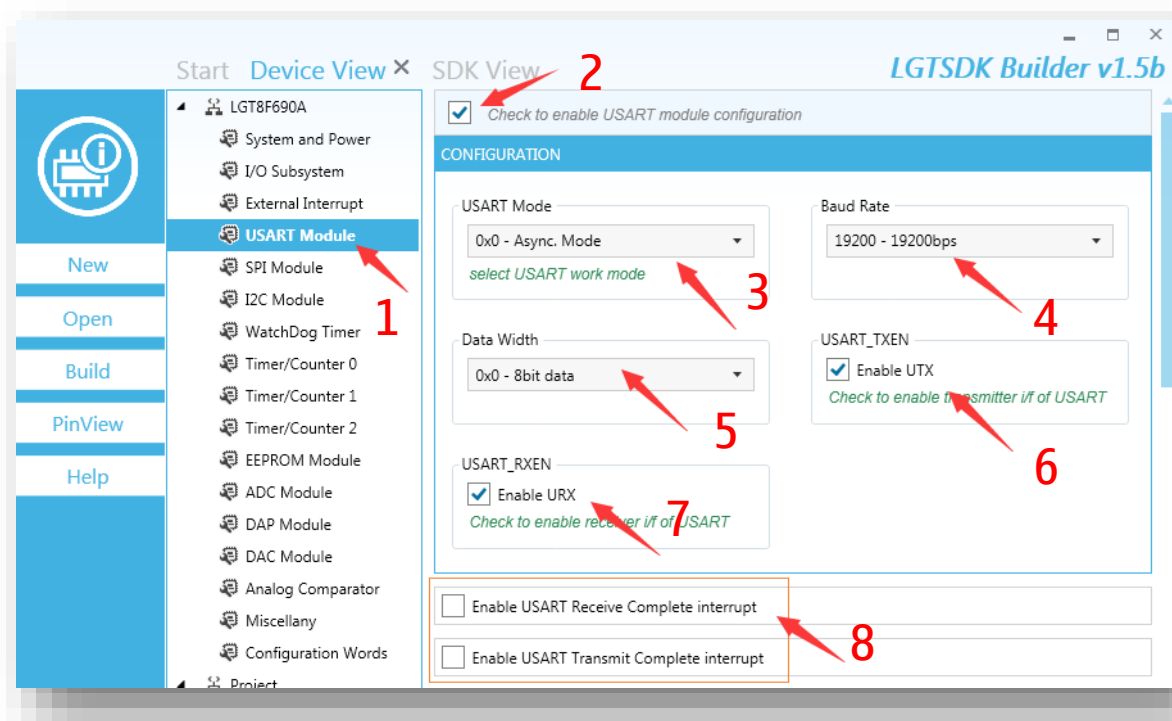
LGT8F690集成了一个支持异步/同步传输模式的通用串行收发器(USART)。其中异步传输模式为常用串口的常用工作模式。本篇教程也是演示如何使用串口的异步模式实现和上位机(PC)的交互通讯。

同步模式与异步模式相比，需要多一根时钟线(UCK)。同步模式有主从设备之分。主从设备的区别在于主设备需要产生同步时钟(UCK)，同步模式的串口非常类似SPI接口。同步模式的串口应用并不多见，因此我们暂不介绍。

下面将以LGTSDK Builder为例，介绍LGT8F690A的串口配置。

启动LGTSDK Builder, 创建一个新的工程，选择芯片类型，项目名称：lgt8f690a_uart

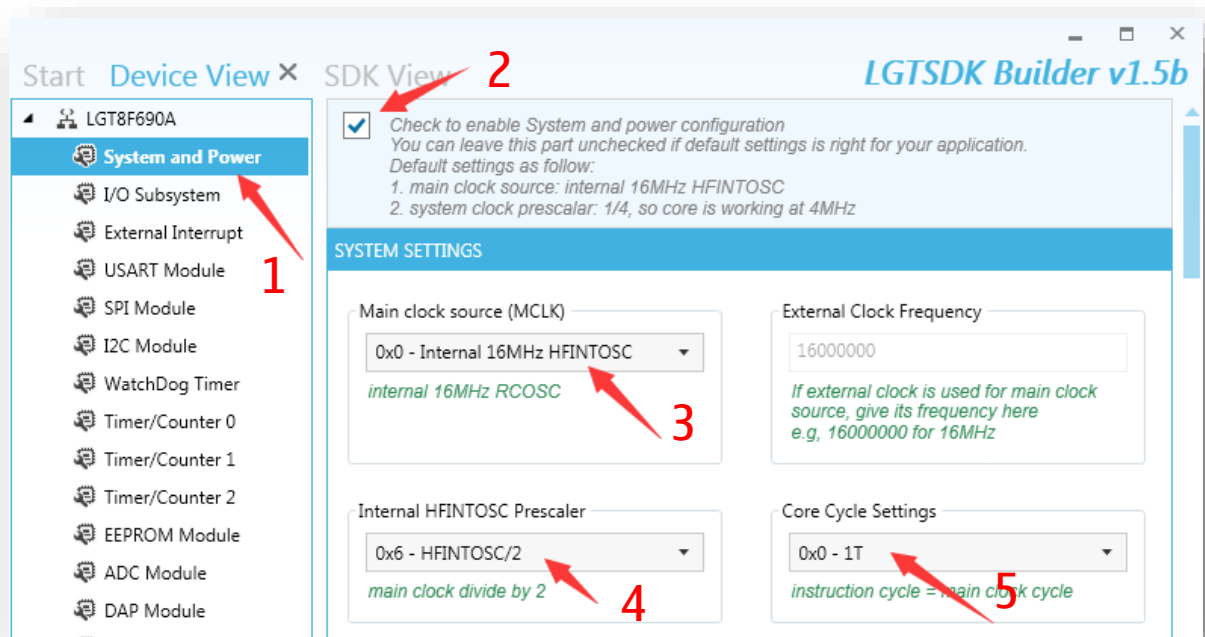
USART基本配置



USART模块的功能配置部分从3~8，都是串口模块的基本功能，下面我们分别详细说明：

	USART配置说明
3	USART工作模式，这里我们选择异步模式
4	串行通讯波特率，选择常用的19200bps。这个波特率的误差很小(<0.2%)，我们选择这个波特率还有一个目的，就是校准芯片内部的16MHz RC
5	串行通讯数据的宽度，选择常用的8位，就是一个字节
6	使能USART的发送接口，使能发送接口后，RB5作为串口的UTX输出口
7	使能USART的接收接口，使能接收接口后，RB6作为串口的URX输入口
8	这里是配置使能串口的数据收发完成中断的。串口中断在写一些异步结构的通讯程序时比较有效。这里我们主要是示例串口的基本通讯功能，中断功能将在以后介绍。

串口模式的配置已经完成！但先不着急生成SDK。因为SDK是根据系统时钟和波特率设置来计算产生波特率发生器的配置信息。因此我们还需要让SDKBuilder清楚的知道我们的系统时钟是什么。因此，我们还需要使能 [System and Power]配置页，选择我们将要使用的系统时钟：



- 3：主时钟源选择内部16MHz RC振荡器；
- 4：主时钟预分频器选择2分频，也就是我们的系统工作在 $16\text{M}/2 = 8\text{MHz}$ ；
- 5：指令周期模式选择1T模式；选择其他模式也没有问题；

重要说明：4/5项配置具有最终决定权。3的配置只是告诉SDKBuilder我们将要使用的主时钟源，时钟源的选择，需要通过配置位确定。这个可以在使用LGTMix_ISP下载程序时将主时钟源选择为内部RC振荡器即可。

由于我们早期提供的LGT8F690A为工程样片，且芯片的16MHz RC可能没有校准。为了愉快的实现准确的串口通讯。我们需要首先校准内部16MHz RC。校准的方法非常的简单直接：

我们将在程序中逐次调整内部RC的校准寄存器，然后向PC发送校准信息。当RC的精度可以产生准确波特率后，我们可以在PC上准确的收到校准信息。这样我们就可以找到RC的校准值。

为了方便我们写程序，这里我们还需要SDKBuilder中的[Miscellany]提供的辅助函数，因此也在SDKBuilder中勾选上。

所有的配置都已完成，在[Device View]中点 [Build]产生SDK代码！

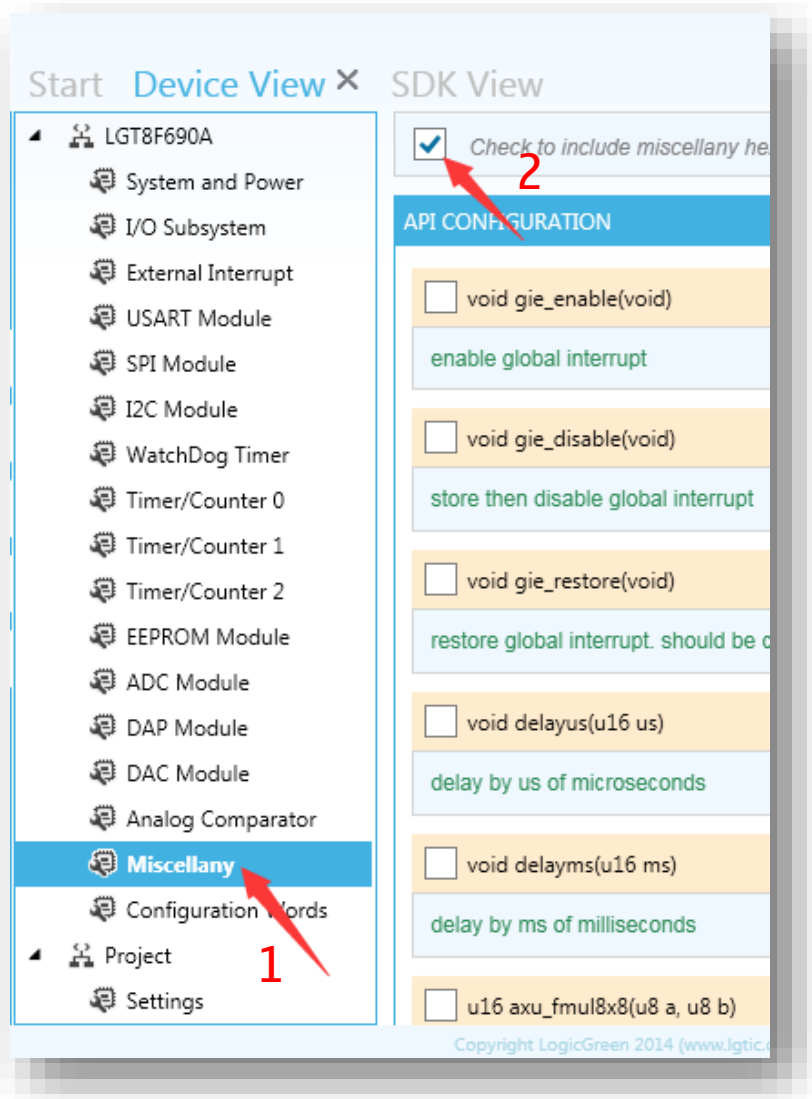
SDKBuilder来到SDK View项目管理界面，打开main.c文件，开始写代码。

首先是校准内部RC的代码：
需要一个变量记录当前的校准值，我们让这个变量在一个合适的范围内变化，然后每个校准值打印一次数据，这样我们就可以在PC观察校准的结果。

由于波特率错误时，数据的接收可能不完整，因此我们需要在每次发数据之间设定一个足够的间隔，避免相邻两次通讯之间的影响。

打印部分，我们使用了一个整形数据转字符串的辅助函数: `axu_utoa()`，这个函数本身是使用LGT8F690A内置的除法器优化的。

下面我们先例程本例程设计的一些SDK接口函数：



函数名称	功能说明	使用方法
usartPutChar()	串口发送一个字符	usartPutChar(0x55) usartPutChar('A')
usartGetChar()	等待并接收一个字符。注意这个函数将会等待，直到从串口接收到一个字符	Byte = usartGetChar()
usartPutStr()	串口发送一个字符串	usartPutStr("Hello, the world")
axu_utoa()	将一个整形转换为一个字符串，支持各种进制的数。要注意字符指针对应数据的宽度	axu_utoa(pbuf, 0x1234, 16) 执行后， pbuf="1234"

Start

Device View

Project 'lgt8f690a_usart'

Drivers

Includes

allinone.h

FSYSCLK.h

global.h

lgt8f690a.h

lgt8f690a_misc.h

lgt8f690a_sys.h

lgt8f690a_usart.h

macros_auto.h

Sources

lgt8f690a_cfw.c

lgt8f690a_misc.c

lgt8f690a_sys.c

lgt8f690a_usart.c

init_auto.c

main.c

SDK View

main.c

```

6
7  #define RCCAL_MIN 0x70
8  #define RCCAL_MAX 0xB0
9
10 // Import external definitions
11 extern void init_modules(void);
12 char caTmp[8];
13
14 int main(void)
15 {
16     u8 rccal = RCCAL_MIN;
17
18     // Device initialization
19     init_modules();
20
21     // Add your code from here
22     while(rccal < RCCAL_MAX)
23     {
24         delays(100);
25
26         usartPutStr("OSCTUNE = 0x");
27         axu_utoa(caTmp, rccal, 16);
28         usartPutStr(caTmp);
29         usartPutStr("\r\n");
30
31         OSCTUNE = rccal;
32
33         if(rccal++ > RCCAL_MAX)
34             rccal = RCCAL_MIN;
35     }
36

```

校准调整的范围

存放校准值的数组

初始化校准值为最小值

打印当前的校准值

校准RC! !

更新RC校准值

隔离两次数据发送的安全间隔

close

xic2mic: Convert from PIC14 to LGT8F690A...

xic2mic: code convert successful!

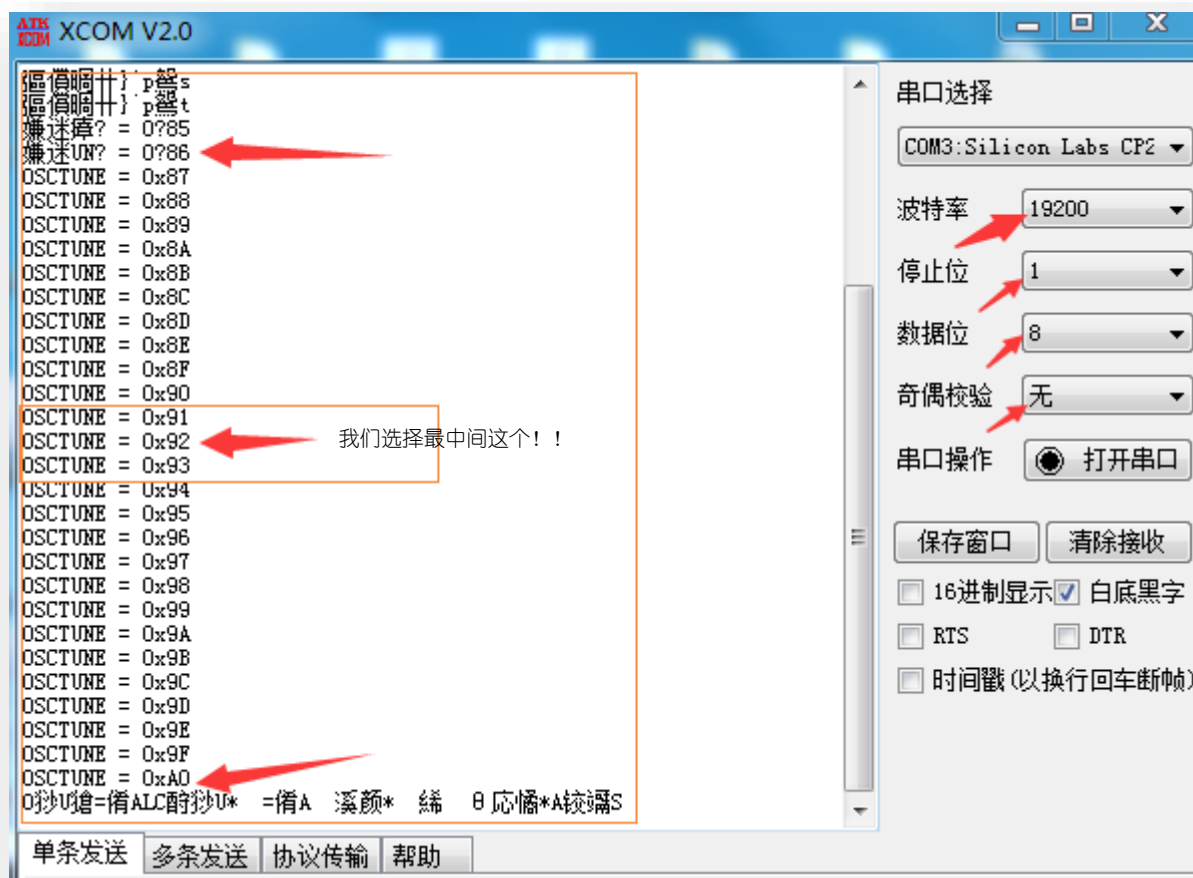
xic2mic: generate mic8-hex file successful!

xic2mic: generate mic8-abin file successful!

Copyright LogicGreen 2014 (www.lgtic.com)

代码编译成功后，可以下载到最小开发板上测试了。

将最小开发板上的串口与PC的串口或者USB转串口设备连接。在PC上打开串口助手，这里我用的是XCOM，配置好串口参数，就可以看到如下数据接收过程：



从串口打印的数据看，校准值在0x87~0xA0都可以满足通讯。我们选择最中间的0x92。记录下这个值，我们可以在初始化是加一句：OSCTUNE = 0x92；完成RC的校准！

说明：

对于量产的LGT8F690A, 或者内部已写入校准值的芯片, 不需要以上校准过程。

串口已经校准完成，接下来可以重新编辑下main.c，实现一个常规的串口打印例程：

