

讯联电子nRF51822蓝牙4.0开发实战

RTC

V:1.0







申明:本教程版权归讯联电子所有。本教程仅供内部客户交流之用。如需引用,请注明出处。由于工程师水平有限,文档难免有所疏漏和错误,由此造成的损失,讯联电子不承担任何责任。



作者: 不离不弃 qq 574912883



RTCO 被协议栈使用了。所以在跑蓝牙程序的情况下。RTCO 不能使用。

RTC 相关寄存器如下:

EVTEN, EVTENSET, EVTENCLR.

这三个寄存器用来设置是否使能某个事件。(TICK, OVRFLW, COMPAREO-3 事件)

INTEN, INTENSET, INTENCLR.

这三个寄存器用来设置某个事件发生时是否触发 RTC 中断。

PRESCALER

该寄存器用来设置 RTC 的时钟分频

分频公式: fRTC [kHz] = 32.768 / (PRESCALER + 1)

RTC 原理跟 TIMER 基本一致。所以程序的设置也是基本一样

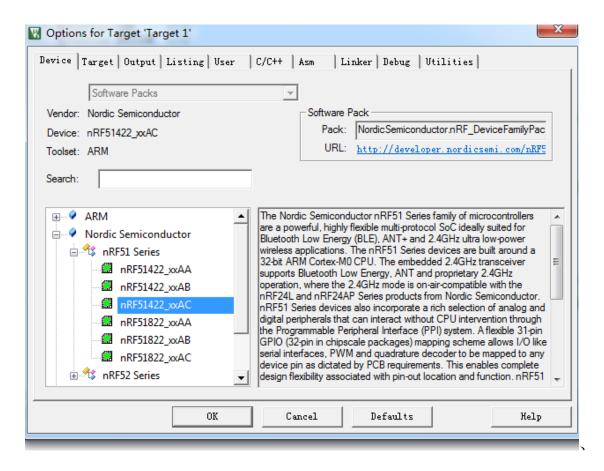
比如需要设置 RTC 的每个时钟滴答为 10ms,即内部 counter 计数一次为 10ms 事件。只需设置 PRESCALER = 327; 那么要做一个 1s 的定时,只需选定一个 compare 寄存器如 compare [0]设置为 100 就可以了。

关于事件,RTC 除了有 compare0-3 事件 (COUNTER 计数值与 cc [0-3] 中的值相等时产生),还存在一个 OVRFLW 事件,在 COUNTER 溢出是产生。以及一个 TICK 事件,即时钟滴答事件。如果使能了这个事件 (EVTEN 中使能),那么在每个时钟滴答 (即 COUNTER 计数一次)都会产生这个事件。比如,上面设置 PRESCALER = 327,则没 10ms,COUNTER 的技术值就会加 1,同时 TICK 事件也会触发。

RTC 的一些 task,如 clear, stop, start 存在 us 级和 ns 级的延迟,使用 RTC 来计时应该考虑这些可能的延迟。具体的延迟事件参考用户手册中 RTC 部分给的说明。



新建工程选择自己板子使用的芯片型号:



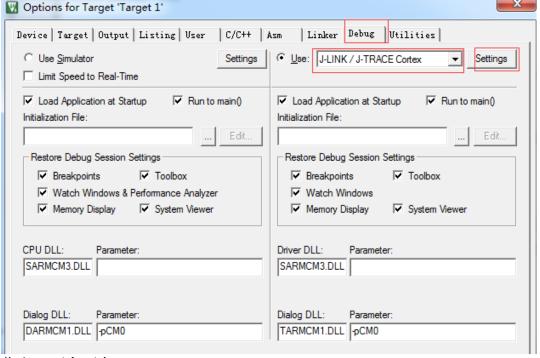
教程中为了更直接的理解模块的使用。不使用 sdk 中提供的库函数,而直接操作寄存器来实现。

所以运行时环境勾选下必要的 CMSIS 下的 CORE, Device 下的 Startup。因为用了 gpio 的函数 勾选一下 nRF_Drivers 下的 nrf_gpio 就可以了。



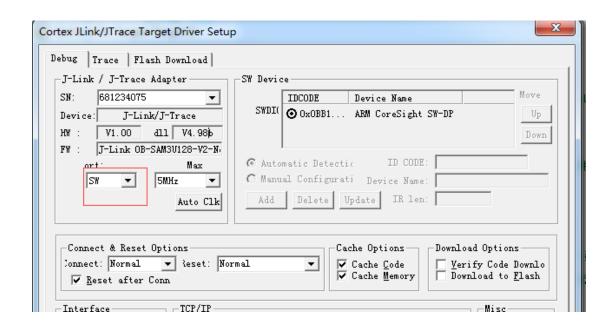
Software Component	Sel.	Variant	Version	Description
⊕ ❖ Board Support				Generic Interfaces
□ ◆ CMSIS				Cortex Microcontr
	<u></u>		3.40.0	CMSIS-CORE for C
DSP			1.4.2	CMSIS-DSP Library
⊕ ◆ RTOS (API)			1.0	CMSIS-RTOS API f
CMSIS Driver				Unified Device Dri
Device				Startup, System Se
Startup	7		8.0.3	System Startup an
🗓 💠 File System		MDK-Pro	6.2.0	File Access on vari
Graphics		MDK-Pro	5.26.1	User Interface on
• Network		MDK-Pro	6.2.0	IP Networking usi
⊕ ◆ RTOS		Clarinox	1.0.0	Clarinox implemen
🗓 💠 Third Parties				
⊕ ◆ USB		MDK-Pro	6.2.0	USB Communicati
• Wireless		Clarinox	2.0.0	Clarinox Wireless I
⊕ ◆ nRF_ANT				
nRF_BLE				
nRF_Drivers				
nRF_Drivers_External				
nRF_Libraries				
nRF_Properitary_RF				
nRF_Serialization				
nRF_SoftDevice				

然后配置 jlink 的设置(我的板子使用的是 jlink 的 sw 方式下载程序)。

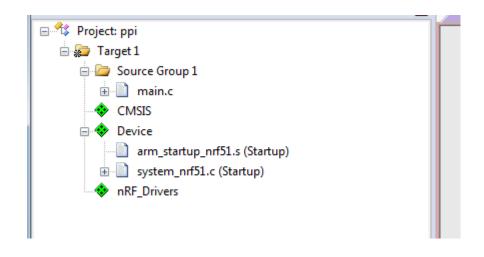


作者: 不离不弃 qq 574912883





创建 main. c 文件, 然后添加到工程中



实际使用 RTC 时,如果是跑的裸板程序的话,是需要自己主动打开 LFCLK.

51822 中的 在上电后 会自动启动 HFCLK RC oscillator 提供给系统所需的时钟。 当系统的需要由 HFCLK crystal oscillator 来提供时钟时,需要用户主动开启 HFCLK crystal oscillator。

作者: 不离不弃 qq 574912883



而对于 LFCLK, 上电后默认是不开启的。RTC 的时钟源是由 32768 分频得到的。 所以在写 RTC 裸机程序的时候需要主动设置 LFCLK 的时钟源, 并启动 LFCLK。

```
下面介绍 main. c 代码细节。
int main(void) {
   nrf_gpio_cfg_output(LED);
   nrf gpio pin set(LED);
   //选择 LFCLK 时钟源为 32.768 kHz crystal oscillator
   //并启动 LFCLK,等待直到其运行了
   NRF CLOCK - > LFCLKSRC = 1;
   NRF CLOCK->TASKS LFCLKSTART = 1;
   while( (NRF CLOCK->LFCLKSTAT&0x01) != 1);
//分频, 使每个 counter 为 10ms
   RTC->PRESCALER = 327;
//使能 compare0 事件
   RTC->EVTENSET = 1 << 16;
//设置 compare 事件产生时触发 RTC 中断。
   RTC->INTENSET = 1 << 16;
   RTC->CC[0] = 100; //1s 定时
   RTC->TASKS START = 1;//启动RTC
   //设置 NVIC 的 RTC 中断寄存器, 开启 RTC 中断
   NVIC_SetPriority(RTC1_IRQn, 1);
   NVIC ClearPendingIRQ(RTC1 IRQn);
   NVIC EnableIRQ(RTC1 IRQn);
   while (1):
   return 0:
void RTC1 IRQHandler(void) {
```

作者: 不离不弃 qq 574912883



```
//注意这里需要清除事件。
RTC->EVENTS_COMPARE[0] = 0;
nrf_gpio_pin_toggle(LED);
//注意这里要主动清 0, 让其重新计数。
RTC->TASKS_CLEAR = 1;
}
```