青风带你玩蓝牙 nRF52832 系列教程

----------作者：青风

出品论坛： www.qfv8.com 青风电子社区

Copyright© 2016-2022 青风电子社区 ALL rights reserved

作者：　　青风
出品论坛: www.qfv8.com
淘宝店：http://qfv5.taobao.com
QQ 技术群：346518370
硬件平台：青云 QY-nRF52832 开发板

一：蓝牙程序信息 log 显示

很多读者对 nrf5x 系列 SDK 程序中的 log 显示一直很困惑，不知道如如何使用，这一讲就针对这个问题答疑解惑，下面的讲解将以 SDK15 版本为例，其他版本的内容类似。

所谓的 log 信息输出，就是为了方便调试者观察程序运行状态，提供人机交互的一种方法，和大家常用的串口 printf 的功能类似。但是在 nrf5x 芯片串口只有一个的状态下，如果你已经使用串口功能了，这时候 log 显示提供一种不占用串口的方式，也就是仿真器 jlink 的 RTT viewer 输出方式。那么如何切换和使用了下面我们来讨论下：

1.1 串口 log 输出

主函数 main 中，第一步就是初始化 log_init()功能，这个功能实际上是可以触发两个通道的：

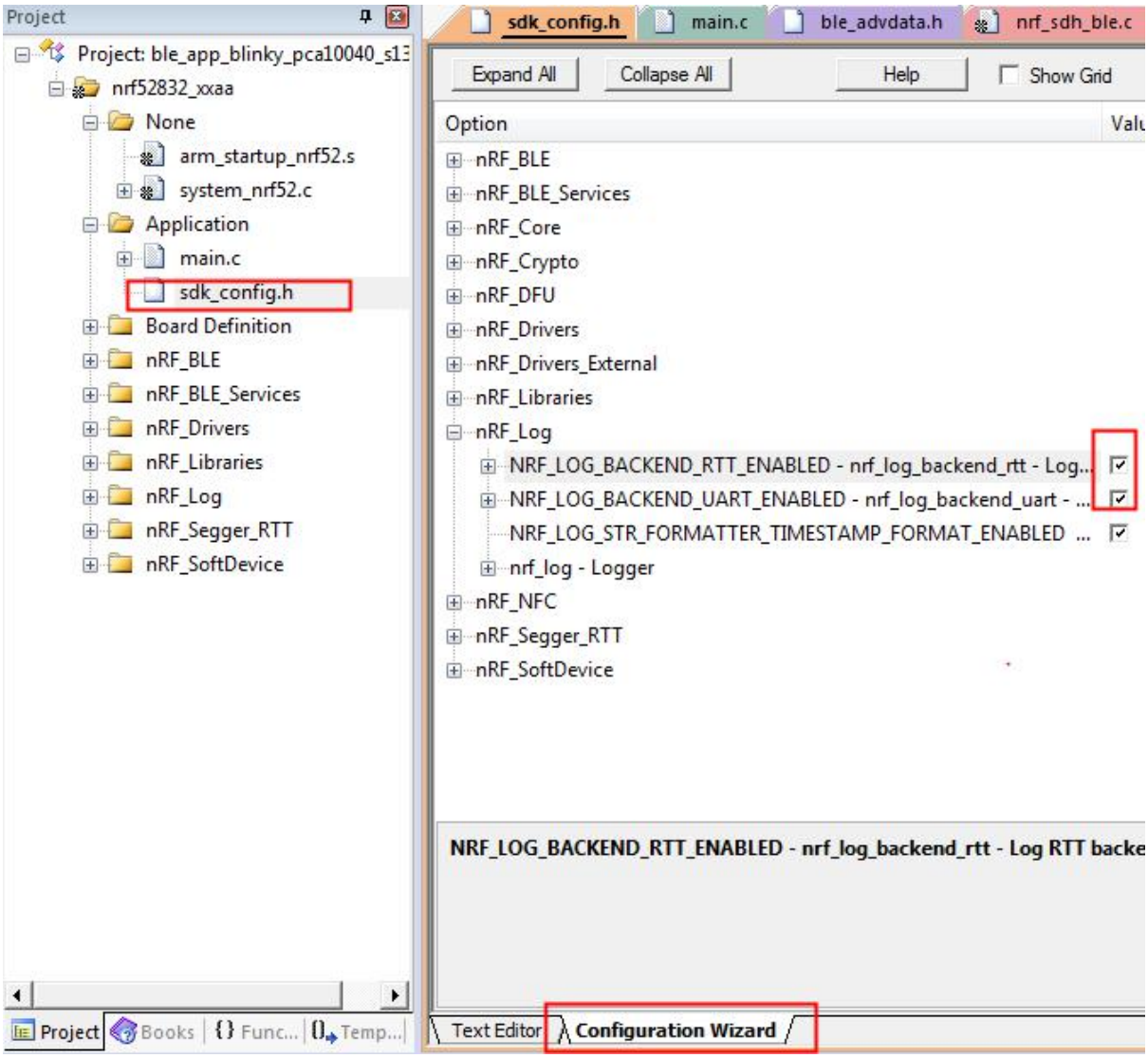
```
int main(void)
{
    // Initialize.
    log_init();
    leds_init();
    timers_init();
    buttons_init();
    power_management_init();
    ble_stack_init();
    gap_params_init();
    gatt_init();
    services_init();
    advertising_init();
    conn_params_init();
```

Copyright© 2016-2022 青风电子社区 ALL rights reserved

```
    // Start execution.
    NRF_LOG_INFO("Blinky example started.");
    advertising_start();

    // Enter main loop.
    for (;;)
    {
        idle_state_handle();
    }
}
```

首先，我们打开一个蓝牙 blinky 灯的实例，打开工程中的 sdk_config.h 文件后，选择编辑向导 Configuration Wizard 内的 nRF_log 下的勾选 NRF_LOG_BACKEND_UART_ENABLE,如下图所示：



切换回 Text Editor 界面，可以发现已经把 NRF_LOG_BACKEND_UART_ENABLE 设置为 1，也就是使能了串口，如下图所示：

Copyright© 2016-2022 青风电子社区 ALL rights reserved

```
7176
7177  // <e> NRF_LOG_BACKEND_UART_ENABLED - nrf_log_backend_uart - Log UART backend
7178  //==========================================================
7179 #ifndef NRF_LOG_BACKEND_UART_ENABLED
7180  #define NRF_LOG_BACKEND_UART_ENABLED 1
7181  #endif
7182  // <o> NRF_LOG_BACKEND_UART_TX_PIN - UART TX pin
7183 #ifndef NRF_LOG_BACKEND_UART_TX_PIN
7184  #define NRF_LOG_BACKEND_UART_TX_PIN 6
7185  #endif
7186
7187  // <o> NRF_LOG_BACKEND_UART_BAUDRATE  - Default Baudrate
7188
7189  // <323584=> 1200 baud
7190  // <643072=> 2400 baud
7191  // <1290240=> 4800 baud
7192  // <2576384=> 9600 baud
7193  // <3862528=> 14400 baud
7194  // <5152768=> 19200 baud
7195  // <7716864=> 28800 baud
7196  // <10289152=> 38400 baud
7197  // <15400960=> 57600 baud
7198  // <20615168=> 76800 baud
7199  // <30801920=> 115200 baud
7200  // <61865984=> 230400 baud
7201  // <67108864=> 250000 baud
7202  // <121634816=> 460800 baud
7203  // <251658240=> 921600 baud
7204  // <268435456=> 1000000 baud
7205
7206 #ifndef NRF_LOG_BACKEND_UART_BAUDRATE
7207  #define NRF_LOG_BACKEND_UART_BAUDRATE 30801920
7208  #endif
7209
```

在 nrf_log_default_backends.c 文件中，当我们使能了 NRF_LOG_BACKEND_UART_ENABLE 后，对应会在改文件中对串口初始化，如下图所示：

```
68
69 #if defined(NRF_LOG_BACKEND_UART_ENABLED) && NRF_LOG_BACKEND_UART_ENABLED
70      nrf_log_backend_uart_init();
71      backend_id = nrf_log_backend_add(&uart_log_backend.backend, NRF_LOG_SEVERITY_DEB
72      ASSERT(backend_id >= 0);
73      nrf_log_backend_enable(&uart_log_backend.backend);
74 #endif
75 }
76 #endif
77
```

```
72  void nrf_log_backend_uart_init(void)
73  {
74      bool async_mode = NRF_LOG_DEFERRED ? true : false;
75      uart_init(async_mode);
76  }
```

大家深入函数内部，会发现这里初始化串口的串口参数，是按照 config.h 文件里的内容进行的，同时管脚只配置了一个打印输出管：

```
203  /**@brief UART default configuration. */
204  #define NRF_DRV_UART_DEFAULT_CONFIG                                           \
205  {                                                                             \
206      .pseltxd            = NRF_UART_PSEL_DISCONNECTED,                         \
207      .pselrxd            = NRF_UART_PSEL_DISCONNECTED,                         \
208      .pselcts            = NRF_UART_PSEL_DISCONNECTED,                         \
209      .pselrts            = NRF_UART_PSEL_DISCONNECTED,                         \
210      .p_context          = NULL,                                               \
211      .hwfc               = (nrf_uart_hwfc_t)UART_DEFAULT_CONFIG_HWFC,          \
212      .parity             = (nrf_uart_parity_t)UART_DEFAULT_CONFIG_PARITY,      \
213      .baudrate           = (nrf_uart_baudrate_t)UART_DEFAULT_CONFIG_BAUDRATE,  \
214      .interrupt_priority = UART_DEFAULT_CONFIG_IRQ_PRIORITY,                   \
215      NRF_DRV_UART_DEFAULT_CONFIG_USE_EASY_DMA                                  \
216  }
217
```

Copyright© 2016-2022 青风电子社区 ALL rights reserved

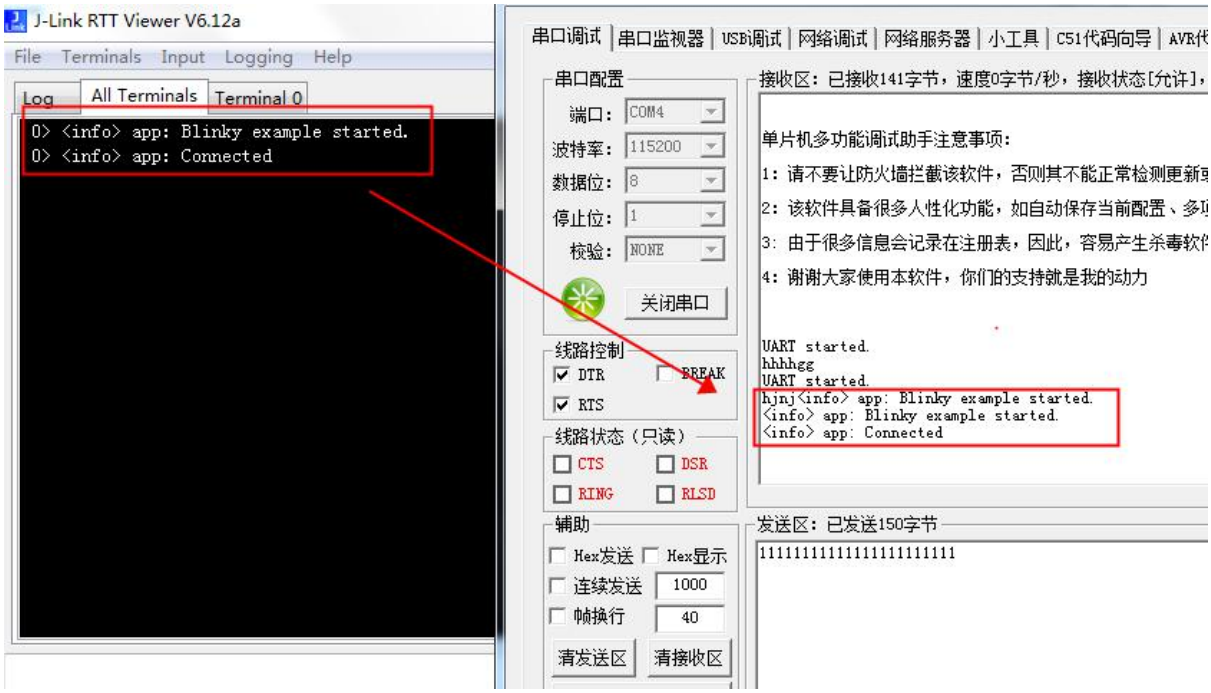
```
static void uart_init(bool async_mode)
{
    nrf_drv_uart_config_t config = NRF_DRV_UART_DEFAULT_CONFIG;
    config.pseltxd  = NRF_LOG_BACKEND_UART_TX_PIN;
    config.pselrxd  = NRF_UART_PSEL_DISCONNECTED;
    config.pselcts  = NRF_UART_PSEL_DISCONNECTED;
    config.pselrts  = NRF_UART_PSEL_DISCONNECTED;
    config.baudrate = (nrf_uart_baudrate_t)NRF_LOG_BACKEND_UART_BAUDRATE;
    ret_code_t err_code = nrf_drv_uart_init(&m_uart, &config, async_mode ? uart_evt_handler : NULL);
    APP_ERROR_CHECK(err_code);

    m_async_mode = async_mode;
}
```

注意这个时候，如果你在配置文件中已经把 NRF_LOG_BACKEND_UART_ENABLE 设置为 1，这时候就不再一次初始化配置串口。

配置完后，打开串口调制助手，按照 config.h 文件里配置的参数，波特率设置为 115200，如下图所示，当程序中有 log 输出代码，就会在串口调试助手中输出显示，代码格式如下：

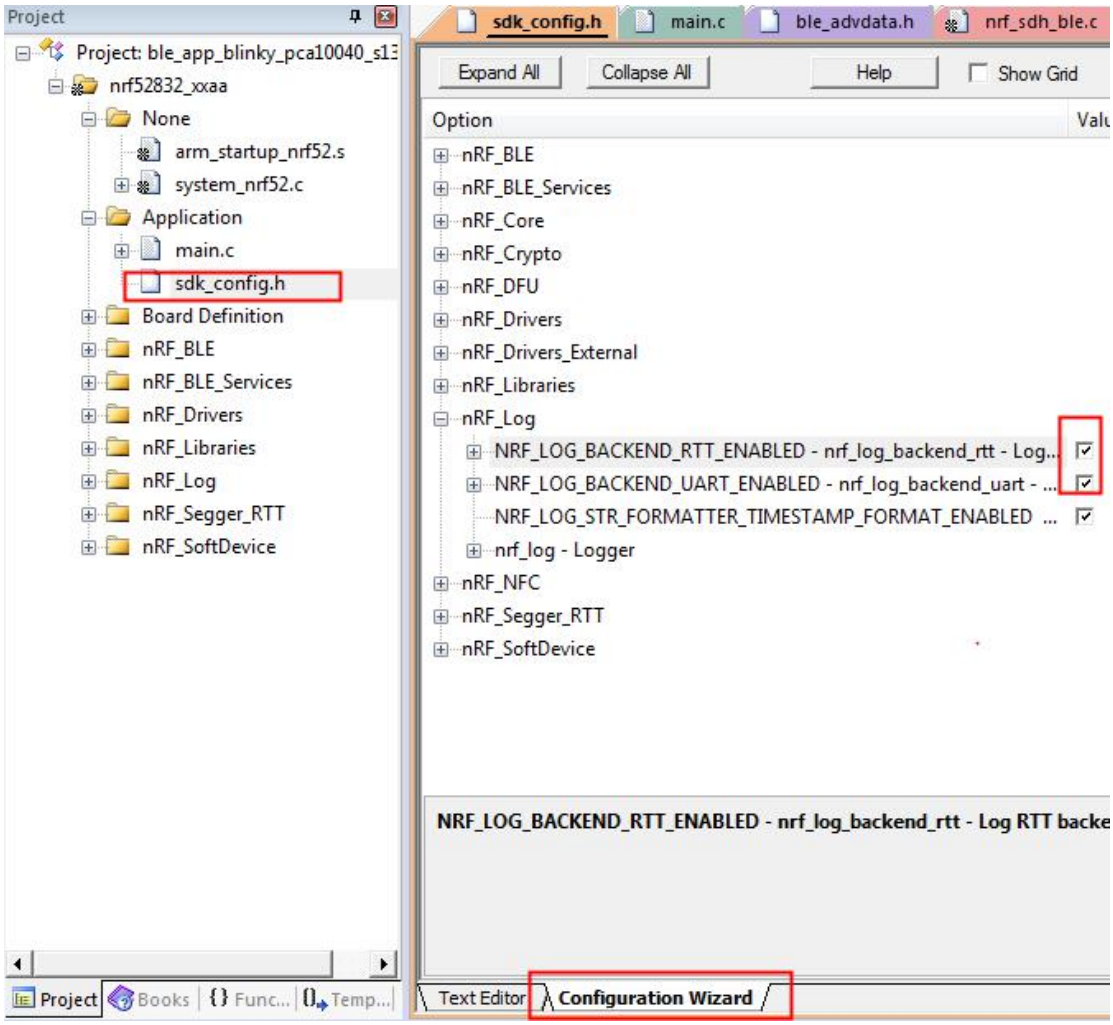
NRF_LOG_INFO("Blinky example started.");



1.2 RTT 的 log 输出

首先，还是在蓝牙 blinky 灯的实例中，打开工程中的 sdk_config.h 文件后，选择编辑向导 Configuration Wizard 内的 nRF_log 下的勾选 NRF_LOG_BACKEND_RTT_ENABLE,如下图所示：

Copyright© 2016-2022 青风电子社区 ALL rights reserved



切换回 Text Editor 界面，可以发现已经把 NRF_LOG_BACKEND_RTT_ENABLE 设置为 1，也就是使能了 RTT，如下图所示：

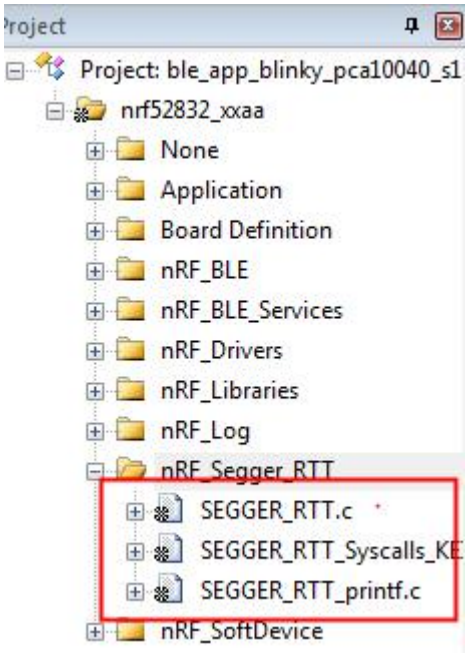
```
7143  //==========================================================
7144  // <e> NRF_LOG_BACKEND_RTT_ENABLED - nrf_log_backend_rtt - Log RTT backend
7145  //==========================================================
7146  #ifndef NRF_LOG_BACKEND_RTT_ENABLED
7147  #define NRF_LOG_BACKEND_RTT_ENABLED 1
7148  #endif
7149  // <o> NRF_LOG_BACKEND_RTT_TEMP_BUFFER_SIZE - Size of buffer for partially processe
7150  // <i> Size of the buffer is a trade-off between RAM usage and processing.
7151  // <i> if buffer is smaller then strings will often be fragmented.
7152  // <i> It is recommended to use size which will fit typical log and only the
7153  // <i> longer one will be fragmented.
7154
7155  #ifndef NRF_LOG_BACKEND_RTT_TEMP_BUFFER_SIZE
7156  #define NRF_LOG_BACKEND_RTT_TEMP_BUFFER_SIZE 64
7157  #endif
```

在 nrf_log_default_backends.c 文件中，当我们使能了 NRF_LOG_BACKEND_RTT_ENABLE 后，对应会在改文件中对 RTT 初始化，如下图所示：

Copyright© 2016-2022 青风电子社区 ALL rights reserved

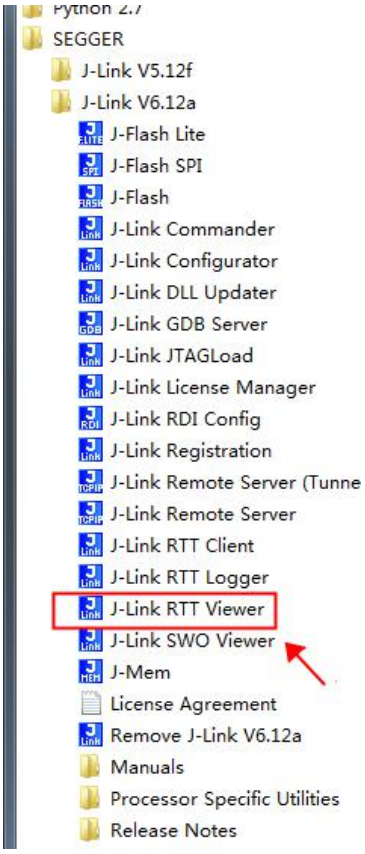
```
58  void nrf_log_default_backends_init(void)
59  {
60      int32_t backend_id = -1;
61      (void)backend_id;
62  #if defined(NRF_LOG_BACKEND_RTT_ENABLED) && NRF_LOG_BACKEND_RTT_ENABLED
63      nrf_log_backend_rtt_init();
64      backend_id = nrf_log_backend_add(&rtt_log_backend.backend, NRF_LOG_SEVERITY_DEBUG);
65      ASSERT(backend_id >= 0);
66      nrf_log_backend_enable(&rtt_log_backend.backend);
67  #endif
```

RTT 是 jlink 提供的一个专门的虚拟串口通道，不占用硬件串口，可以实现串口方式一样的观察窗口，在工程目录中，添加了 nRF_Segger_RTT 文件夹，里面包含了 3 个 RTT 工程的支持包，如下图所示：

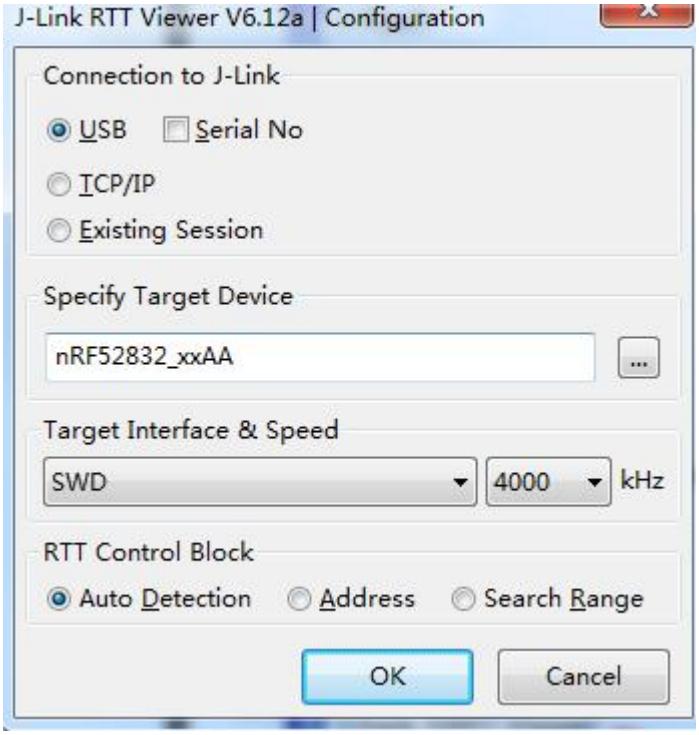


配置完后，我们在电脑开始位置，找到你的 SEGGER 的安装位置，找到驱动版本下的观察窗体 J-LINK RTT Viewer ,这个软件在你安装 jlink 驱动的时候会自动安装上，如下图所示，点击打开：

Copyright© 2016-2022 青风电子社区 ALL rights reserved

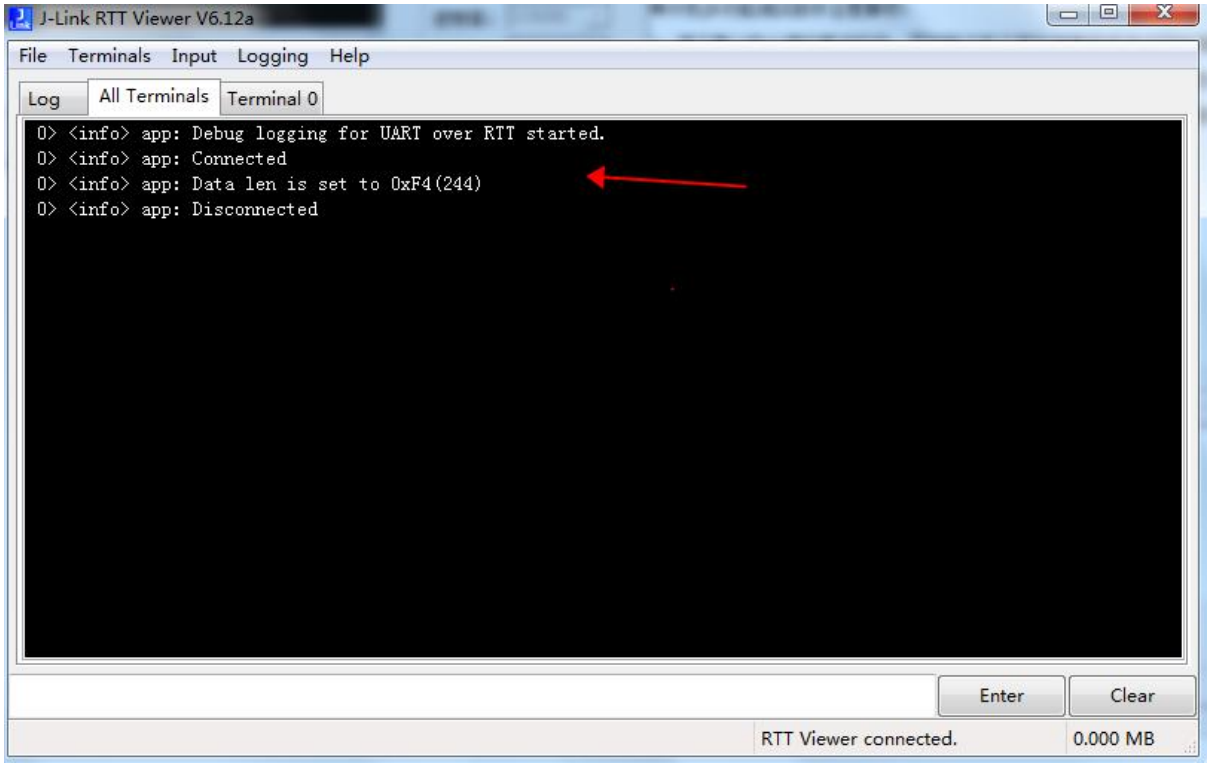


打开后弹出选择界面。这个功能我们必须接上 jlink，jlink 在线状态下调试，设置参数如下，使用 usb 端口输出：

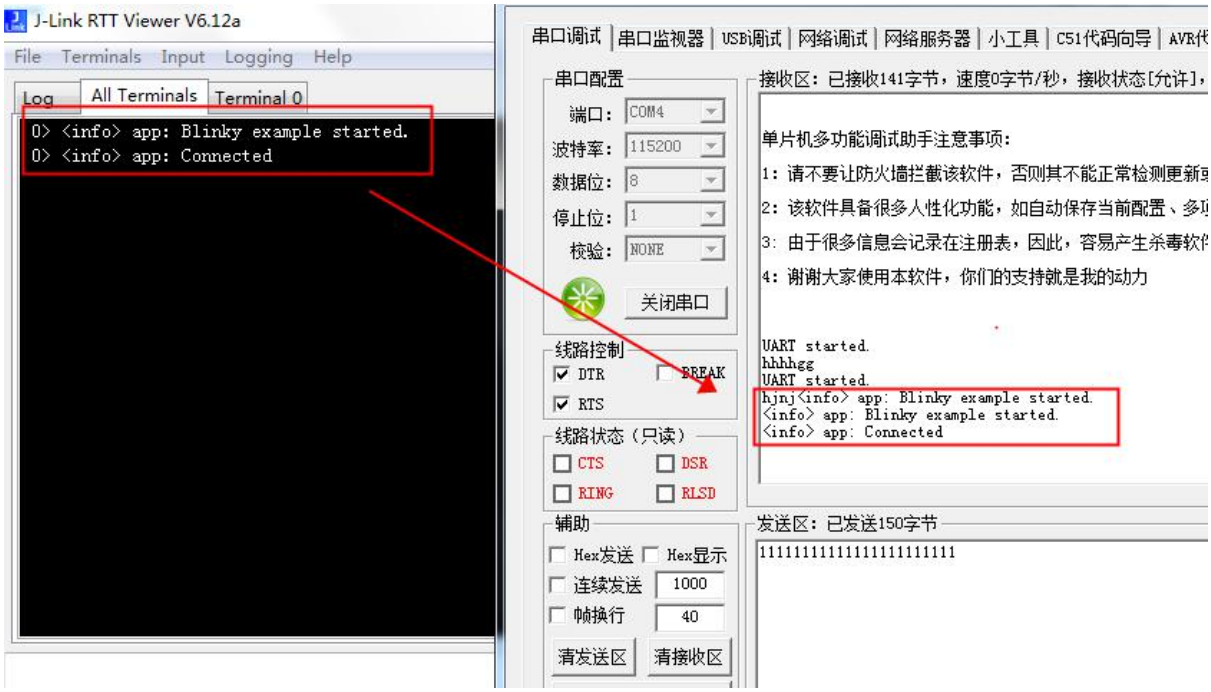


点击 OK 后，会弹出 All Terminals 终端界面，如下图所示，这个界面就是打印输出相关 log 信息的，当串口无法使用的时候，就可以选择 RTT 进行输出：

Copyright© 2016-2022 青风电子社区 ALL rights reserved



如果你串口可以使用，可以把串口和 RTT 同时使能，对比两个端口输出的数据，发现效果是一样的，如下图所示：



这篇文章就讲到这里，实际上 nRF5x 的 SDK 里提供了这个 LOG 功能，可以实现类似串口打印 prittf 的功能，同时还兼容了 RTT 的输出，在串口有限的情况下是非常管用的，很方便大家调试程序。今天在这里抛砖迎玉，大家可以灵活使用。

Copyright© 2016-2022 青风电子社区 ALL rights reserved