

J-link 抓取 SystemView 数据

1、SystemView 能用来分析什么问题？

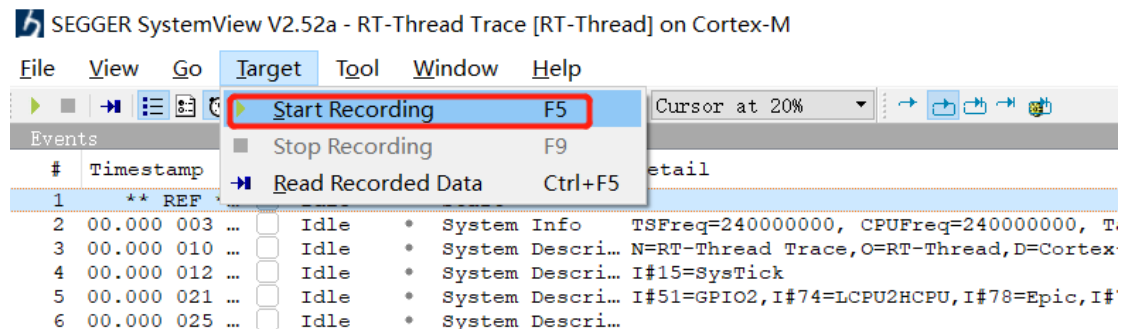
----SystemView 是一款帮助用户进行系统调试和分析的强大工具，能够显著缩短开发和调试时间，提高开发效率。RT-Thread 上提供了 SystemView 工具对系统进行调试和分析。该工具，可以详细的看到每一个线程，每一个中断，占用 CPU 的时间，特别适合找到占用 CPU 资源的地方。如分析中断、图像显示帧率等

2、如何打开 SystemView？

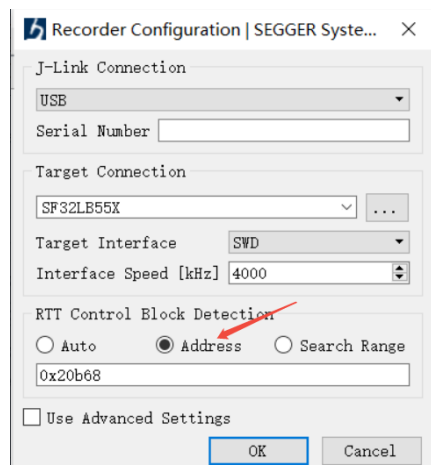
- 1) 配置：打开 hcpl 的 menuconfig，选择 SiFla SDK configuration -> Third party packages → SystemView: A Segger utility for analysis and trace the RTOS 其他都采用默认配置。
- 2) 打开 finish 指令，Hcpu 的串口 console 输入命令：rtt_show_address，返回 RTT Control Block 的地址，如：

```
msh />
msh />rtt_show_address
RTT Control Block Detection Address is 0x20b68
msh />
```

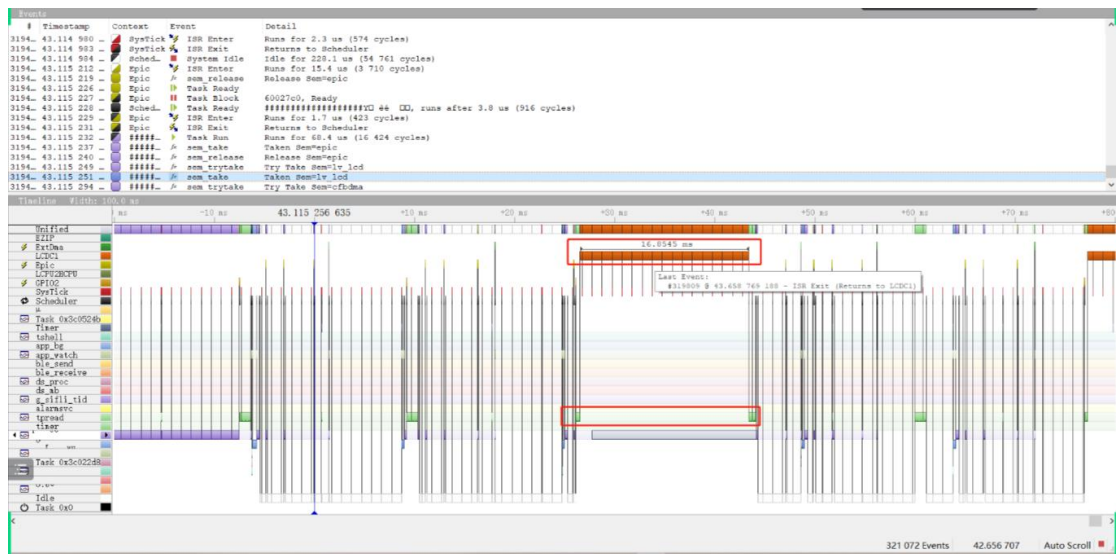
3) 打开 SystemView.exe 软件，菜单->Target->Start Recording



4) 弹出的对话框中的 address 中填写 RTT Control Block 的地址



5) Start Recording 记录后，就能看到如下的窗口



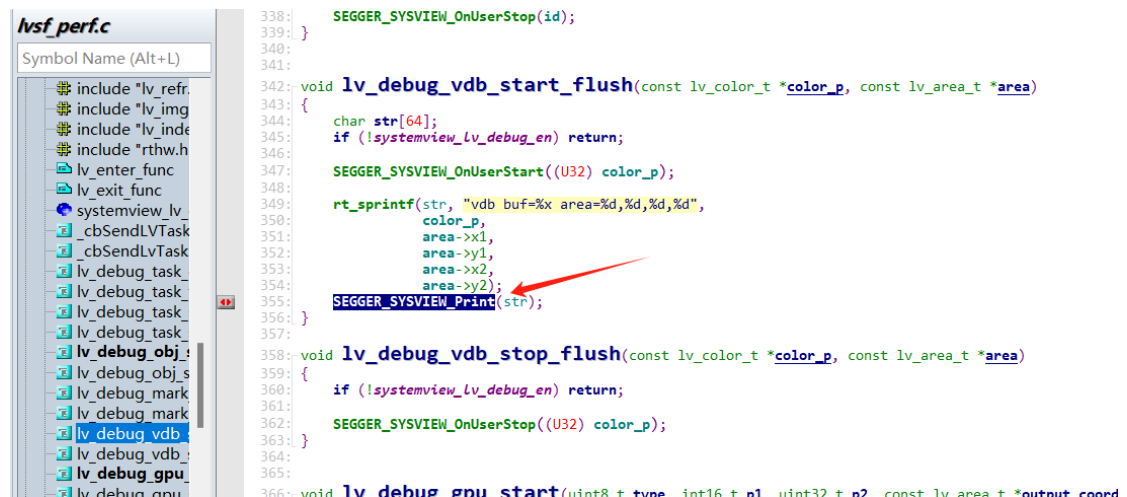
6) 等抓取完成之后，点击保存即可。

PS: 添加 Segger 打印到 SystemView 的串口中，可以在 SDK 的代码中添加如下代码：

```
extern void SEGGER_SYSVIEW_Print(const char* s);
```

```
SEGGER_SYSVIEW_Print("A");
```

如：



3、52X 系列怎么抓取 SystemView 数据？

因为 52X 系列默认没有使用 J-link 的管脚，而使用 SifliUsartServer.exe 工具转的 jlink 抓取的数据很慢，不能正常分析使用，所以需要将 52X 的 PA18 以及 PA19 默认使用的 UART_DGB 功能改为 SWD 的功能定义。

- 1) bsp_pinmux.c 文件 BSP_PIN_Init 函数中, PA18,PA19 配置为 jlink 接口

```
#if 0

// UART1

HAL_PIN_Set(PAD_PA19, USART1_TXD, PIN_PULLUP, 1);

HAL_PIN_Set(PAD_PA18, USART1_RXD, PIN_PULLUP, 1);

#else

//SWD

HAL_PIN_Set(PAD_PA18, SWDIO, PIN_PULLDOWN, 1);

HAL_PIN_Set(PAD_PA19, SWCLK, PIN_PULLDOWN, 1);

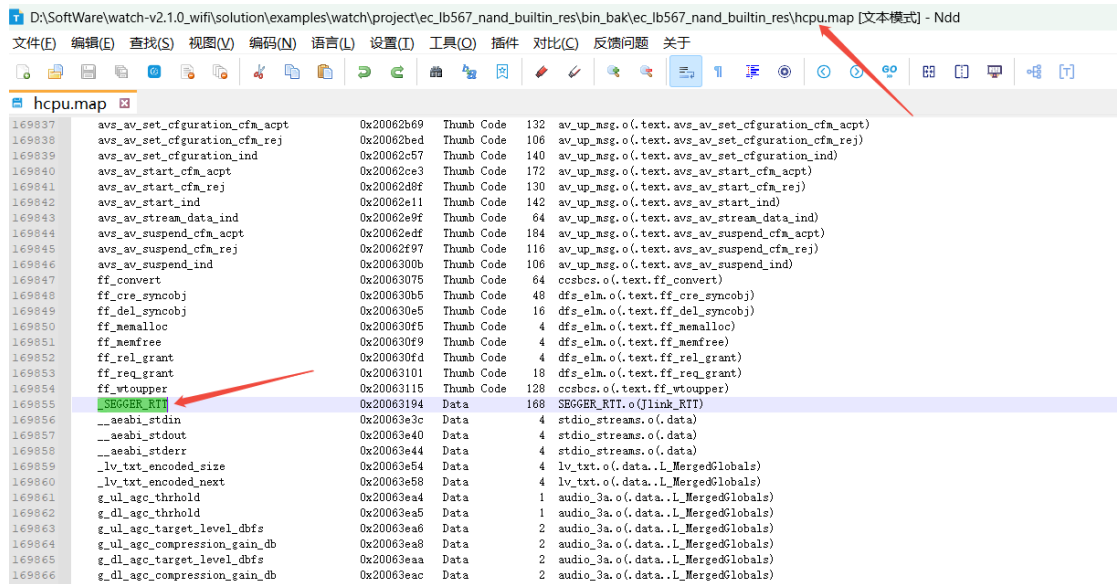
HAL_PIN_SetMode(PAD_PA18, 1, PIN_DIGITAL_IO_PULLDOWN);

HAL_PIN_SetMode(PAD_PA19, 1, PIN_DIGITAL_IO_PULLDOWN);

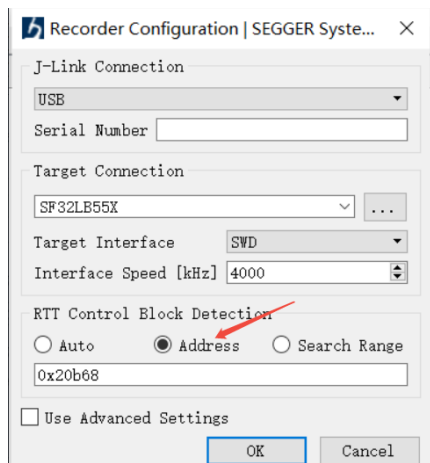
#endif
```

- 2) 由于 PA18 和 PA19 修改为 SWD 口, 就不能打印 log 了, hcpu 的 log 打印也改成 jlink 的 segger 打印

- 3) 通过 hcpu 编译出来的 map 文件查询_SEGGER_RTT 的指定地址



- 4) 打开 SystemView 工具, 将上面查询到的_SEGGER_RTT 地址填到下面的 address 中, 即可开始抓取



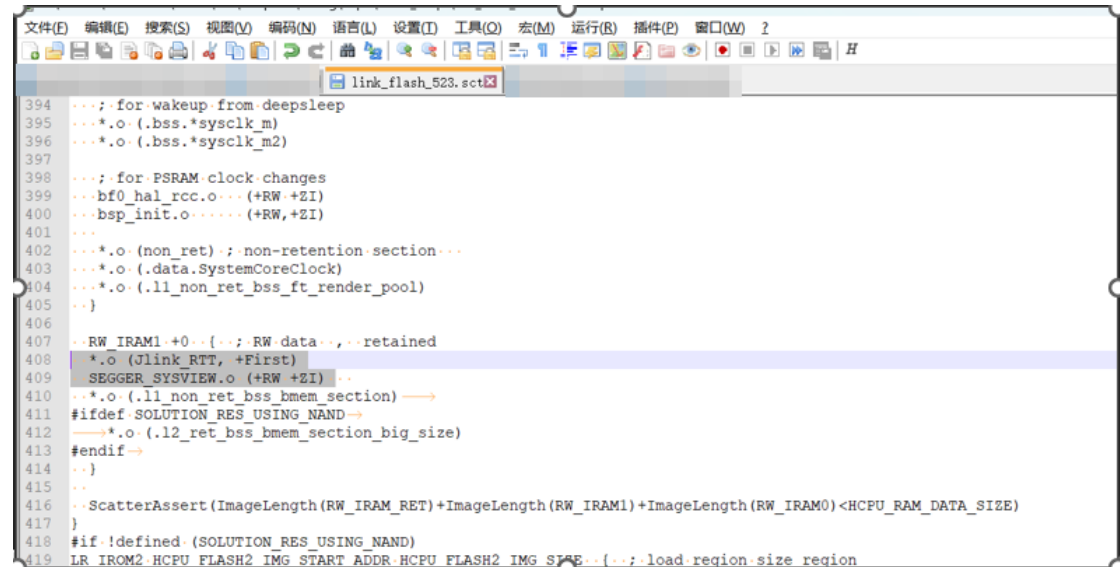
5) 遇到的可能问题问题

a、52X 连接 SystemView 死机，原先放在 PSRAM 上面，跟上位机交互可能会出现 psram cache 异常，导致死机，需要把如下两个放到 sram 里

*.o (Jlink_RTT, +First)

SEGGER_SYSVIEW.o (+RW +ZI) 把这两个放到 sram 里。

如:



```
394 ...; for wakeup from deepsleep
395 ...*.o (.bss.*sysclk_m)
396 ...*.o (.bss.*sysclk_m2)
397
398 ...; for PSRAM clock changes
399 ...bf0_hal_rcc.o... (+RW +ZI)
400 ...bsp_init.o... (+RW +ZI)
401 ...
402 ...*.o (non_ret); non-retention section...
403 ...*.o (.data.SystemCoreClock)
404 ...*.o (.l1_non_ret_bss_ft_render_pool)
405 ...}
406
407 ..RW_IRAM1+0..{...; RW data...; retained
408     *.o (Jlink_RTT, +First)
409     SEGGER_SYSVIEW.o (+RW +ZI) ...
410 ...*.o (.l1_non_ret_bss_bmem_section) ->
411 #ifdef SOLUTION_RES_USING_NAND ->
412     ->*.o (.l2_ret_bss_bmem_section_big_size)
413 #endif ->
414 ...}
415 ...
416 ...ScatterAssert(ImageLength(RW_IRAM_RET)+ImageLength(RW_IRAM1)+ImageLength(RW_IRAM0)<HCPU_RAM_DATA_SIZE)
417 }
418 #if !defined (SOLUTION_RES_USING_NAND)
419 LR IROM2 HCPU FLASH2 IMG START ADDR HCPU FLASH2 IMG SIZE...{...; load region size region
```