

Systemview 工具抓点

工具路径

====10.21.10.6====Software====Setup_SystemView_V252a(pro).zip （老版本，和谐版本）

====10.21.10.6====Software====SystemView_Windows_V358_x64.exe （[2024-10-09]发布版本）

操作方法

1. 用 Menuconfig 打开 systevew

```

[Top] → Third party packages
Sifli Configuration
[ ] Enable Nano Jpeg Decoder (NEW)
[ ] Enable libhelix mp3 decoder(code size 30K) (NEW)
[ ] Enable libmad mp3 decoder(code size 75K) (NEW)
[*] LittlevGL2RTT: The LittlevGL gui lib adapter RT-Thread
    LVGL configuration --->
[ ] LZ4 compression lib (NEW) ----
[ ] Seleted MBedTLS functions for boot (NEW)
[ ] Enable Micropython (NEW)
[ ] MicroPython: A lean and efficient Python implementation for microcontrollers and cons
[ ] mpu6xxx: Universal 6-axis sensor driver library (NEW) ----
[ ] Enable Nanopb (NEW)
[ ] nmea_lib: GPS nmea parse (NEW) ----
[ ] Enable Open H.264 (NEW)
[ ] libopus (NEW)
[ ] Enable Pixman (NEW)
[ ] Enable Quick JS (NEW)
[ ] Enable RLOTTIE (NEW)
[ ] rpmsg-lite: rpmsg-lite for RT-Thread (NEW) ----
*- Segger RTT package (NEW) --->
[*] SystemView: A Segger utility for analysis and trace the RTOS --->
[ ] Enable sensor libs (NEW)
[ ] Enable UI Loader (NEW) ----
[ ] Enable VG-Lite (NEW) ----

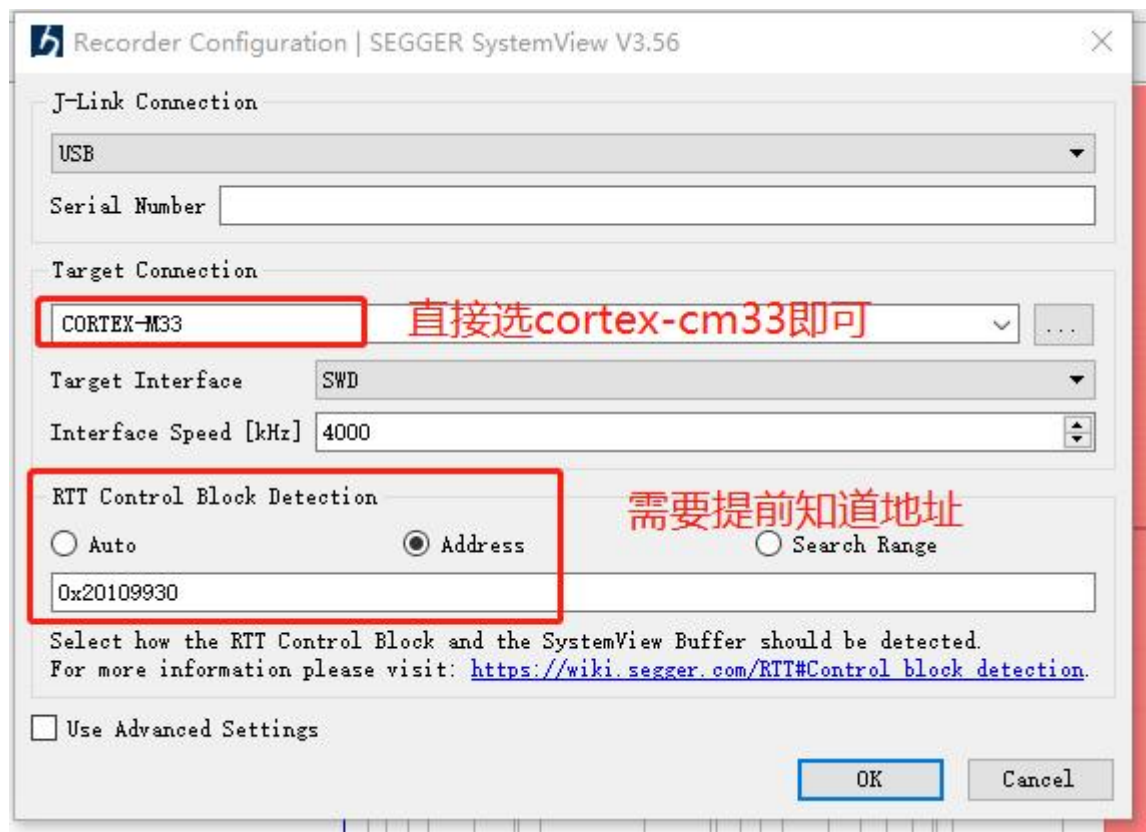
```

2. 调整 Systemview log 的循环 buffer 大小，如果内存比较充裕建议设置 32KB，这样不容易丢包

```
(Top) → Third party packages→ SystemView: A Segger utility for analysis and trace the RTOS→ SystemView buffer configuration
          ↑↑↑↑↑↑↑
          Sifli Configuration
(32768) RTT buffer size
(1) RTT channel (NEW)
[*] Use static buffer (NEW)
[ ] Enable post mortem analysis mode (NEW)
```

3. 编译烧录板子

4. 打开 Systemview, 设置 RTT Control Block 的地址:

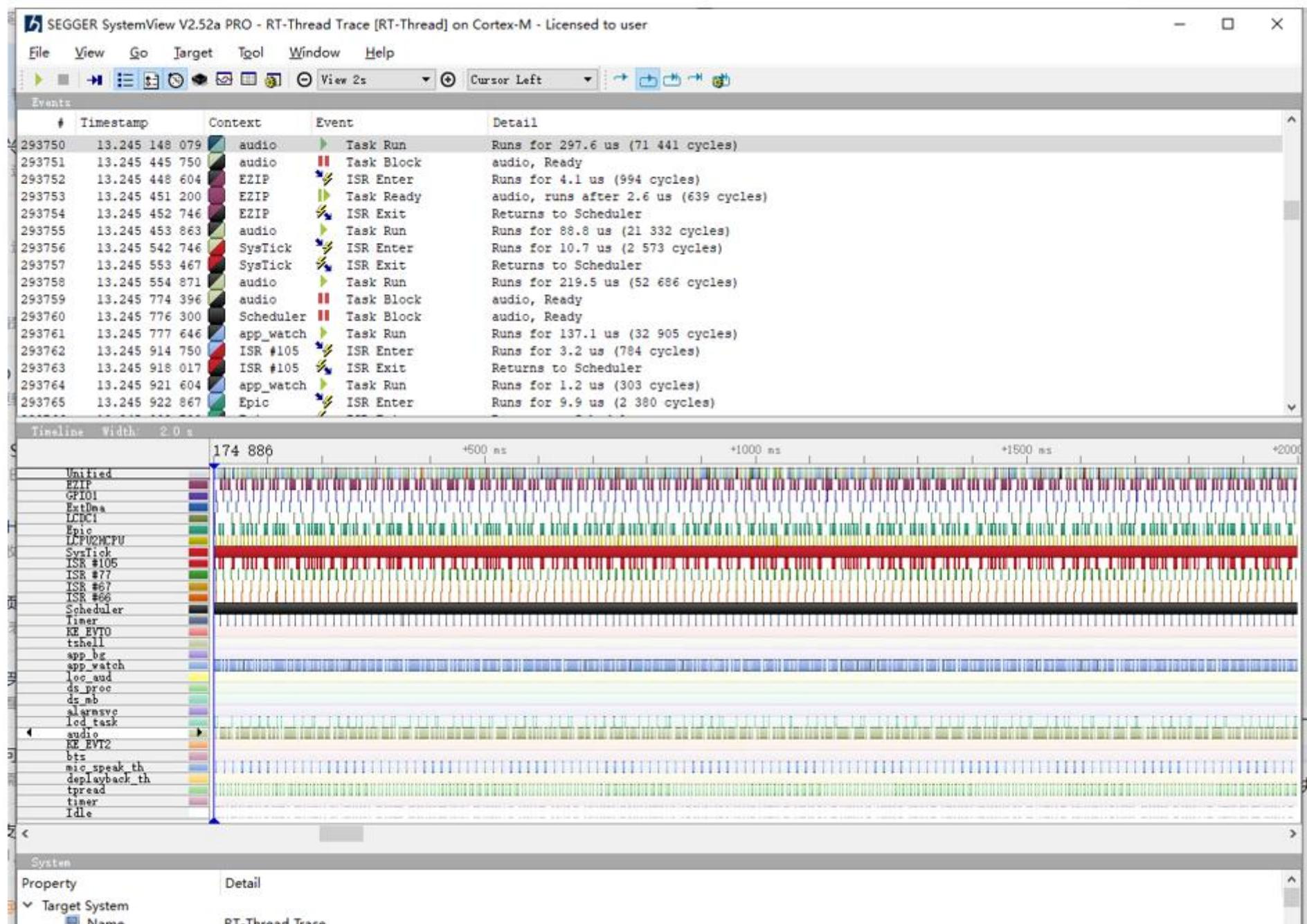


RTT Control Block 的地址是动态的，每次编译可能有变化，可以通过 2 种方式获取：

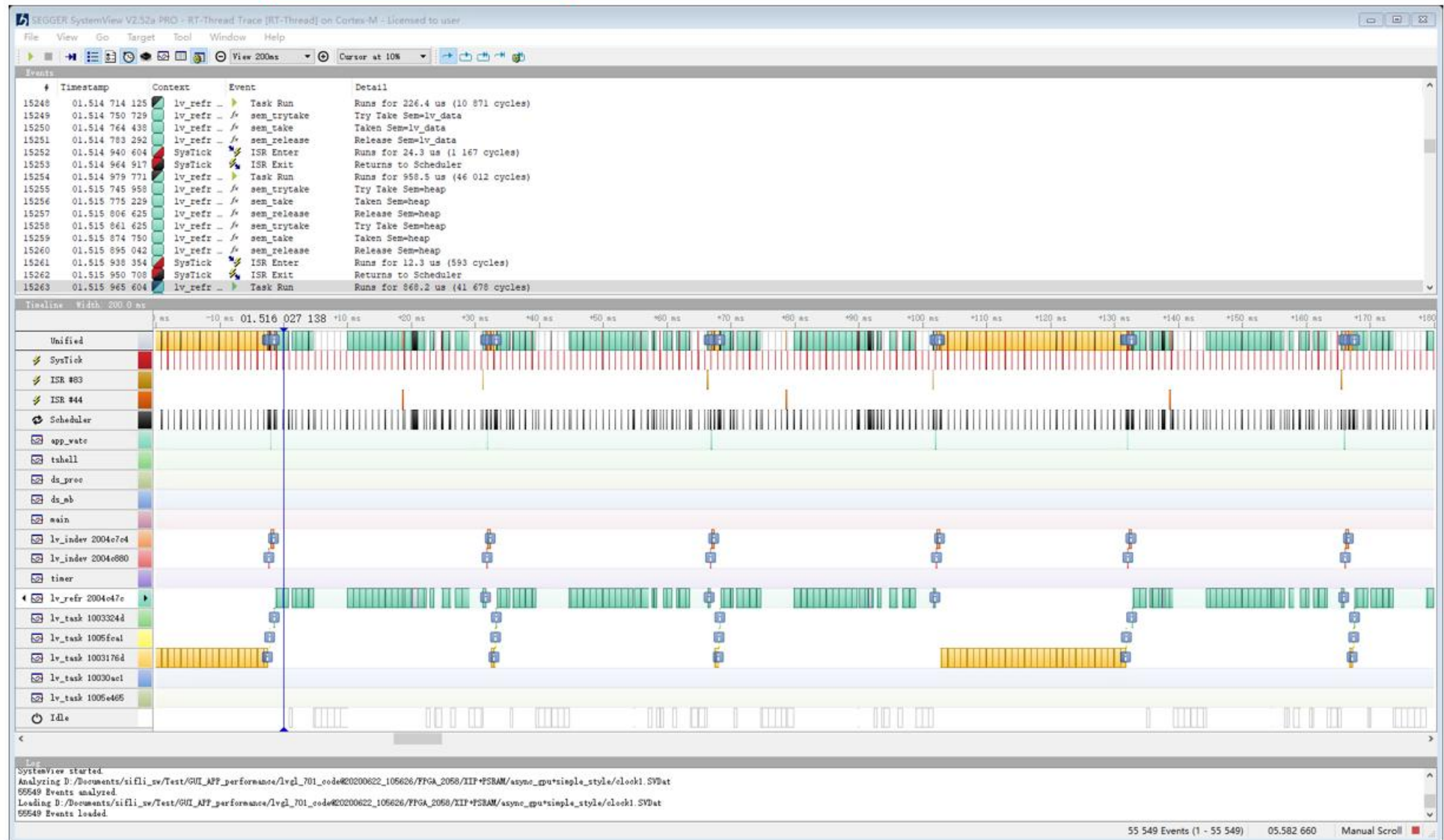
- 通过 console 命令：rtt_show_address
- 直接查找 map 表中 _SEGGER_RTT 的地址

g_pll_vco_tab	0x2003afb0	Data	24	drv_audcodec.o(.data.g_pll_vco_tab)
g_rf_ful_ver	0x2003afc8	Data	4	bt_rf_fulcal.o(.data.g_rf_ful_ver)
g_voice_eqValue	0x2003afcc	Data	200	drv_audprc.o(.data.g_voice_eqValue)
pin_irq_hdr_tab	0x2003b114	Data	384	drv_gpio.o(.data.pin_irq_hdr_tab)
working_directory	0x2003b559	Data	256	dfs.o(.data.working_directory)
SEGGER_RTT	0x2003b65c	Data	168	SEGGER_RTT.o(Jlink_RTT)
__stdin	0x2003c3b8	Data	84	stdio_streams.o(.bss)
__stdout	0x2003c40c	Data	84	stdio_streams.o(.bss)
__stderr	0x2003c460	Data	84	stdio_streams.o(.bss)
_random_number_data	0x2003c4b4	Data	228	rand.o(.bss)
libgcc_start	0x2003c500	Data	84	libgcc.o(.bss)

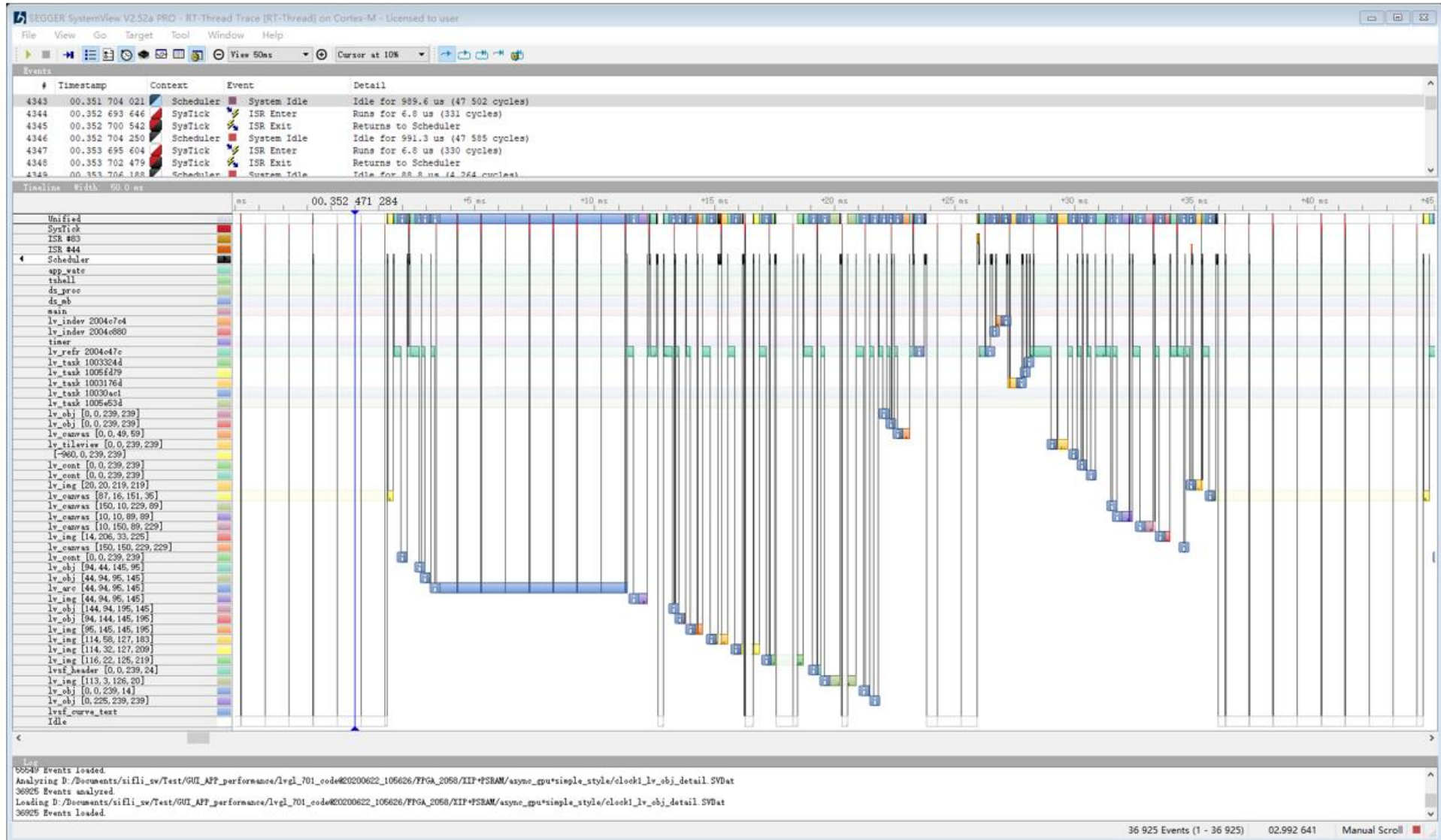
5. 连接上 system view 后，默认将只显示整个 RTOS 线程、中断的运行情况



6. 通过 `finsh` 命令 `hook_rtt_obj 1` 可以打开 `rt-thread` 信号量、`mutex`、`mailbox` 等 `rt_object` 的 `systemview` log.
7. 通过 `finsh` 命令 `en_lv_perf_debug 1` 可以显示启动 `lvgl` task 以及其他 `lvgl` 相关的 mark



8. 通过 `finsh` 命令 `lv refr cfg sysview 1` 可以将 `lv_obj` 在 `lv_refr_task` 内刷新的时间显示在 `systemview` 上, 还有 VDB 送屏的 log ↵



9. 常见问题

a. 连接不上 Systemview

Jlink 要保持通畅、板子不能进入睡眠，

RTT Control Block 的地址是否设置正确

b. 连上后录一小段后卡住

可能是工具版权问题，可以尝试这个比较老的版本：([10.21.10.6 Software Setup_SystemView_V252a\(pro\).zip](#))

c. 中间丢数据

i. SWD 的频率可以尝试更高

ii. Systemview 的循环 buffer 增加到更大，比如 64KB

iii. 去掉一些不需要的 log 点，比如不关心 rtthread 的信号量、mailbox 等 IPC 消息时，可以通过 finsh 命令来关掉：

```
hook_rtt_obj 0
```

d. 连上 Systemview 死机

需要把 Systemview 跟上位机交互的 Control Memory 放到 SRAM, 比如修改工程对应的 Scatter file, 将这两个放到 sram 里

```
*o (Jlink_RTT, +first)
```

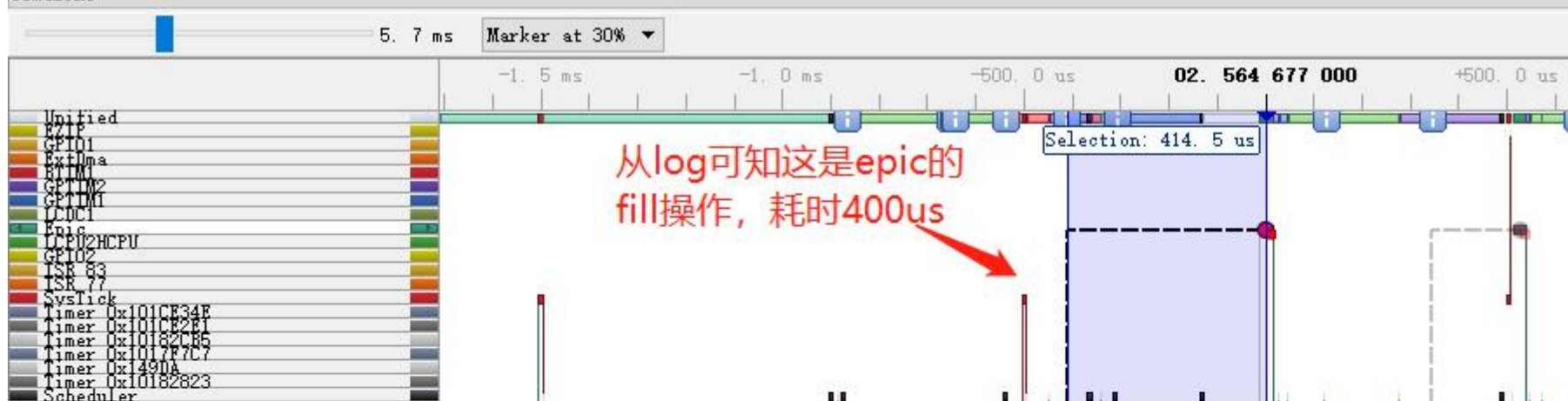
```
SEGGER_SYSVIEW.o (+RW+ZI)
```

我们自己加的一些 **user marker**:

1. 0xAAAAAAAA 代表 EPIC 的渲染的开始和结束
2. 0xBBBBBBBBBB 代表刷屏操作的启动和结束
3. Framebuffer 的地址 代表在图形库层的 VDB 刷屏的启动和结束 *这个跟实际的屏幕刷屏时间可能不太匹配，是个纯软件的标记。见 *LV-DEBUG-VDB-START-FLUSH/LV-DEBUG-VDB-STOP-FLUSH* 的宏实现，在 *lv_debug.c* (lvgl v7) 或 *lvgl_perf.c* (lvgl v8,v9)
4. 0xC094FABF 代表 CopyFramebuffer 的启动和结束，SRAM>PSRAM 的 copy。 *这个在新版本 *drv_lcd_fb.c* 里面才开始使用，有些老版本没有这个文件，那就没有这个 *marker*。

#	Timestamp	Context	Event	Detail
68658	02. 564 033 683	lv_task 100bb575	• Stop Marker 0x000000C4	Ran for 3. 479 us, pass #56
68663	02. 564 138 492	obj [0,0,409,493]	• Log	mask[0,0,409,169]
68670	02. 564 262 438	thread	• Start Marker 0xAAAAAAAA	Runs for 414. 563 us, pass #1045
68671	02. 564 265 717	thread	• Log	EPIC Fill opa=255, color=0x0 w,h=410,170
68681	02. 564 366 992	image [40,151,129,240]	• Log	mask[0,0,409,169]
68686	02. 564 677 000	Epic	• Stop Marker 0xAAAAAAAA	Ran for 414. 563 us, pass #1045
68703	02. 564 804 775	image [3,67,92,156]	• Log	mask[0,0,409,169]
68707	02. 565 020 379	thread	• Start Marker 0xAAAAAAAA	Runs for 184. 654 us, pass #1046
68708	02. 565 025 775	thread	• Log	EPIC IMG_ROT area=39,150,129,169 angle=0, scale=1
68718	02. 565 205 033	Epic	• Stop Marker 0xAAAAAAAA	Ran for 184. 654 us, pass #1046
68734	02. 565 324 450	image [93,62,182,151]	• Log	mask[0,0,409,169]
68738	02. 565 535 079	thread	• Start Marker 0xAAAAAAAA	Runs for 252. 479 us, pass #1047
68739	02. 565 540 288	thread	• Log	EPIC IMG_ROT area=12,76,82,146 angle=0, scale=129
68749	02. 565 762 592	image [148,151,237,240]	• Log	mask[0,0,409,169]
68754	02. 565 787 558	EZIP	• Stop Marker 0xAAAAAAAA	Ran for 252. 479 us, pass #1047
68767	02. 566 010 983	thread	• Start Marker 0xAAAAAAAA	Runs for 244. 138 us, pass #1048
68768	02. 566 016 017	thread	• Log	EPIC IMG_ROT area=96,65,178,147 angle=0, scale=1
68781	02. 566 255 121	Epic	• Stop Marker 0xAAAAAAAA	Ran for 244. 138 us, pass #1048
68795	02. 566 334 163	obj [0,0,409,493]	• Log	mask[0,0,409,169]
68802	02. 566 381 208	obj [0,0,409,493]	• Log	mask[0,0,409,169]

Timeline



如何加一些自己的 Log 点：

比如 EPIC 的 0xAAAAAAAA 点是如何打印出来的：

1. 在启动 epic 的地方加入：

```
SEGGER_SYSVIEW_OnUserStart(0xAAAAAAAA);
```

```
SEGGER_SYSVIEW_Print ("This is an epic filling operation xxxxx");
```

2. 在 EPIC 结束的中断里面加入：

```
SEGGER_SYSVIEW_OnUserStop(0xAAAAAAAA);
```

2025.02.12 使用笔记（liguiyong）

1. 安装 systemview



2. 编译前修改 menuconfig 打开配置

```
[Top) → SiFl1 SDK configuration → Third party packages
↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑ Sifli Configuration
] Enable Micropython
] MicroPython: A lean and efficient Python implementation for microcont
] mpu6xxx: Universal 6-axis sensor driver library ----
*] Enable Nanopb
] nmea_lib: GPS nmea parse ----
] Enable Open H.264
] libopus
] Enable Pixman
] Enable Quick JS
] Enable RLOTTIE
*- Segger RTT package --->
*] SystemView: A Segger utility for analysis and trace the RTOS --->
] Enable sensor libs
```

```
→Third party packages →SystemView: A Segger utility for analysis and trace the RTOS →SystemView buffer configuration  
Sifli Configuration  
(32768) RTT buffer size  
(1) RTT channel  
[*] Use static buffer  
[ ] Enable post mortem analysis mode
```



3. 修改.set, 确保 segger 相关的部分放在 sram 的最开始。(这样做的目的是: 在 SEGGER SystemView 中配置 Recorder Configuration 中 RTT Control

Block Detection 为固定地址 0x20000000

Edit	Navigate	Blocks	Whitespaces	Diff	View
solution/framework/_template_/config/hcpu/linker_script/link_flash_523.sct: f4089507			solution/framework/_template_/config/hcpu/linker_script/link_flash_523.sct: Wk		
<pre>2 #else 3 ScatterAssert((ImageLength(RW_PSRAM_NON_RET) + ImageLength(RW_PSRAM_NAND) + ImageLength(RW_PSRAM_RET) + ImageLength(RW_PSRAM_FLASH)) < 0x100000000); 4 #endif 5 6 #endif 7 8 RW_IRAM0_HPSYS_RAM0_BASE UNINIT { ; ZI data, not retained 9 *.o (Jlink_RTT, +First) 10 ;SEGGER_SYSVIEW.o (+RW +ZI) 11 } 12 13 //The previous 128KB is the retention area 14 audio_server.o (.bss.audio_server_stack) 15 audio_server.o (.bss.bt_downvoice_stack) 16 media_player_server.o (.bss.ffmpeg_thread_stack) 17 *.o (.l1_non_ret_bss_uhog_be_ram_buf) 18 #if defined (SOLUTION_RES_USING_NAND) 19 drv_spi_nand.o (.l2_ret_bss_nand_buf) 20 nand.o (.bss.dhara_meta_cache) 21 #endif 22 bts2_app_av_snk.o (.bss.deplayback_thread_stack) 23 *.o (STACK) ; ISR stack 24 *.o (.l1_non_ret_bss_drv_lcd_stack) 25 *.o (.bss.sensor_thread_stack) 26 *.o (.bss.idle_thread_stack) 27 *.o (.l1_non_ret_bss_app_bg_thread_stack) 28 *.o (.l1_non_ret_bss_timer_thread_stack) 29 *.o (.l1_non_ret_bss_touch_thread_stack) 30 *.o (.l1_non_ret_bss_watch_thread_stack) 31 *.o (.l1_non_ret_bss_frambuf) 32 *.o (.l1_non_ret_bss_app_sram_non_ret_cache) 33 *.o (.l2_non_ret_bss_ft_render_pool) 34 #ifndef BSP_USING_PSRAM 35 *.o (.l2_non_ret_data_*) 36 *.o (.l2_non_ret_bss_*) 37 *.o (.l2_cache_non_ret_data_*) 38 *.o (.l2_cache_non_ret_bss_*) 39 #endif 40 efuse.o (+ZI) 41 42 *.o (.l1_non_ret_bss_ring) 43 } 44 45 RW_IRAM_RET +0 { 46 *.o (Jlink_RTT) 47 SEGGER_SYSVIEW.o (+RW +ZI) 48 *.o (.l1_ret_text_*) 49 *.o (.l1_ret_rodata_*) 50 }</pre>			<pre>312 #else 313 ScatterAssert((ImageLength(RW_PSRAM_NON_RET) + ImageLength(RW_PSRAM_NAND) + ImageLength(RW_PSRAM_RET) + ImageLength(RW_PSRAM_FLASH)) < 0x100000000); 314 #endif 315 316 #endif 317 318 RW_IRAM0_HPSYS_RAM0_BASE UNINIT { ; ZI data, not retained 319 *.o (Jlink_RTT, +First) 320 SEGGER_RTT.d(+RW+ZI) 321 SEGGER_SYSVIEW.o(+RW+ZI) 322 } 323 324 //The previous 128KB is the retention area 325 audio_server.o (.bss.audio_server_stack) 326 audio_server.o (.bss.bt_downvoice_stack) 327 media_player_server.o (.bss.ffmpeg_thread_stack) 328 *.o (.l1_non_ret_bss_uhog_be_ram_buf) 329 #if defined (SOLUTION_RES_USING_NAND) 330 drv_spi_nand.o (.l2_ret_bss_nand_buf) 331 nand.o (.bss.dhara_meta_cache) 332 #endif 333 bts2_app_av_snk.o (.bss.deplayback_thread_stack) 334 *.o (STACK) ; ISR stack 335 *.o (.l1_non_ret_bss_drv_lcd_stack) 336 *.o (.bss.sensor_thread_stack) 337 *.o (.bss.idle_thread_stack) 338 *.o (.l1_non_ret_bss_app_bg_thread_stack) 339 *.o (.l1_non_ret_bss_timer_thread_stack) 340 *.o (.l1_non_ret_bss_touch_thread_stack) 341 *.o (.l1_non_ret_bss_watch_thread_stack) 342 *.o (.l1_non_ret_bss_frambuf) 343 *.o (.l1_non_ret_bss_app_sram_non_ret_cache) 344 *.o (.l2_non_ret_bss_ft_render_pool) 345 #ifndef BSP_USING_PSRAM 346 *.o (.l2_non_ret_data_*) 347 *.o (.l2_non_ret_bss_*) 348 *.o (.l2_cache_non_ret_data_*) 349 *.o (.l2_cache_non_ret_bss_*) 350 #endif 351 efuse.o (+ZI) 352 353 *.o (.l1_non_ret_bss_ring) 354 } 355 356 RW_IRAM_RET +0 { 357 *.o (Jlink_RTT) 358 SEGGER_RTT.d(+RW+ZI) 359 SEGGER_SYSVIEW.o(+RW+ZI) 360 *.o (.l1_ret_text_*) 361 *.o (.l1_ret_rodata_*) 362 }</pre>		

4. 修改 lvsf_perf.c

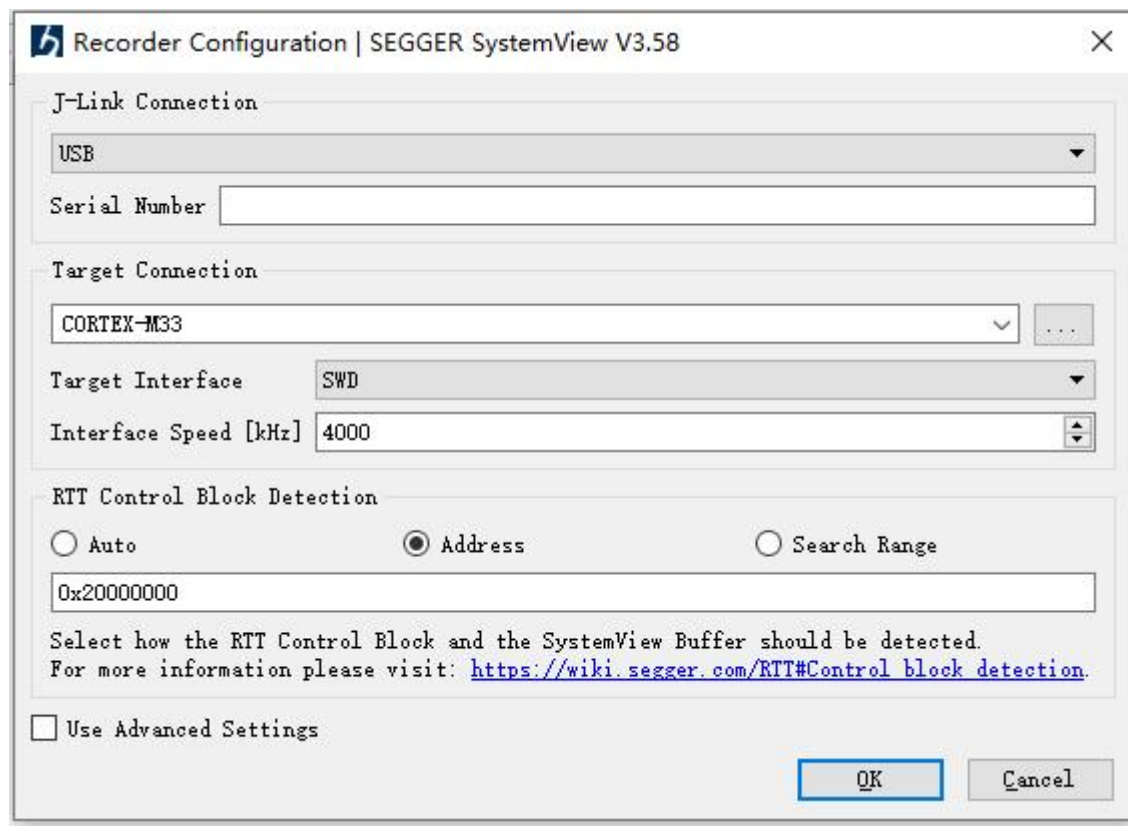
middleware/lvgl/lvsf_perf.c: d1854d0c	middleware/lvgl/lv
<pre>1 2 #include "littlevgl2rtt.h" 3 #include "lvsf.h" 4 5 static uint8_t obj_detail = 0; 6</pre>	<pre>1 2 #include "littlevgl2rtt.h" 3 #include "lvsf.h" 4 5 static uint8_t obj_detail = 1; 6</pre>

middleware/lvgl/lvtf_perf.c: d1854d0c	middleware/lvgl/lvtf_perf.c: Working Tree
<pre>189 #ifdef PKG_USING_SYSTEMVIEW 190 #include "SEGGER_SYSVIEW.h" 191 #include "lv_gc.h" 192 #include "lv_refr.h" 193 #include "lv_img.h" 194 #include "lv_indev.h" 195 #include "rthw.h" 196 extern void lv_enter_func(U32 id, const char *task_name, U32 param1, U32 param2); 197 extern void lv_exit_func(U32 id); 198 static bool systemview lv debug en = false; 199 200</pre>	<pre>189 #ifdef PKG_USING_SYSTEMVIEW 190 #include "SEGGER_SYSVIEW.h" 191 #include "lv_gc.h" 192 #include "lv_refr.h" 193 #include "lv_img.h" 194 #include "lv_indev.h" 195 #include "rthw.h" 196 extern void lv_enter_func(U32 id, const char *task_name, U32 param1, U32 param2); 197 extern void lv_exit_func(U32 id); 198 static bool systemview lv debug en = true; 199 200</pre>

5. 编译烧写机器后, 进入 setting 页面, 找到 develop 模式, 打开 Switch to Jlink
6. 板上跳线 (两个) 到 SWD



7. 打开 systemview, 配置 SEGGER SystemView 中配置 Recorder Configuration 中 RTT Control Block Detection 为固定地址 0x20000000



8. Start Recorder, 选择 jlink, 开始抓数。