Use on RT **-THREAD** **SYSTEMVIEW** analysis tool

RT-THREAD Document Center

WWW.RT-THREAD.ORG

Friday 28th September, 2018

!!! abstract "Abstract" This application note introduces the SystemView visualization analysis tool and how to use it in
It is used on RT-Thread to debug and analyze the system.

1 Purpose and structure of this paper

1.1 Purpose and Background of this Paper

As MCU performance continues to grow and embedded products become increasingly complex, new challenges are emerging for system debugging and analysis. Debugging a specific function or issue often requires significant effort. SystemView is a powerful tool for system debugging and analysis, significantly reducing development time and improving efficiency. This article aims to help users use SystemView to debug and analyze systems on RT-Thread.

1.2 Structure of this paper

This article first explains what the SystemView tool is, then introduces the SystemView software package on RT-Thread, along with specific methods for enabling and configuring it. It then uses a demo to demonstrate the detailed usage of the SystemView analysis tool. Finally, a parameter directory is provided for reference during configuration.

2 What is SystemView

SystemView is a tool for online debugging of embedded systems. It analyzes which interrupts and tasks have been executed, and the order in which they have been executed. It also displays the timing of the acquisition and release of kernel objects, such as semaphores, mutexes, events, and message queues. This is particularly useful when developing and manipulating complex systems with multiple threads and events.

SystemView consists of two parts:

- PC program, collects and displays data from the embedded terminal, and can also save this data locally for later use analyze.
- Embedded end program, collects the operating data of the embedded system and transmits them through the RTT module of J-Link For PC

Because SystemView's data transmission utilizes J-Link's RTT technology, SystemView can only be used when the development board is connected with J-Link.

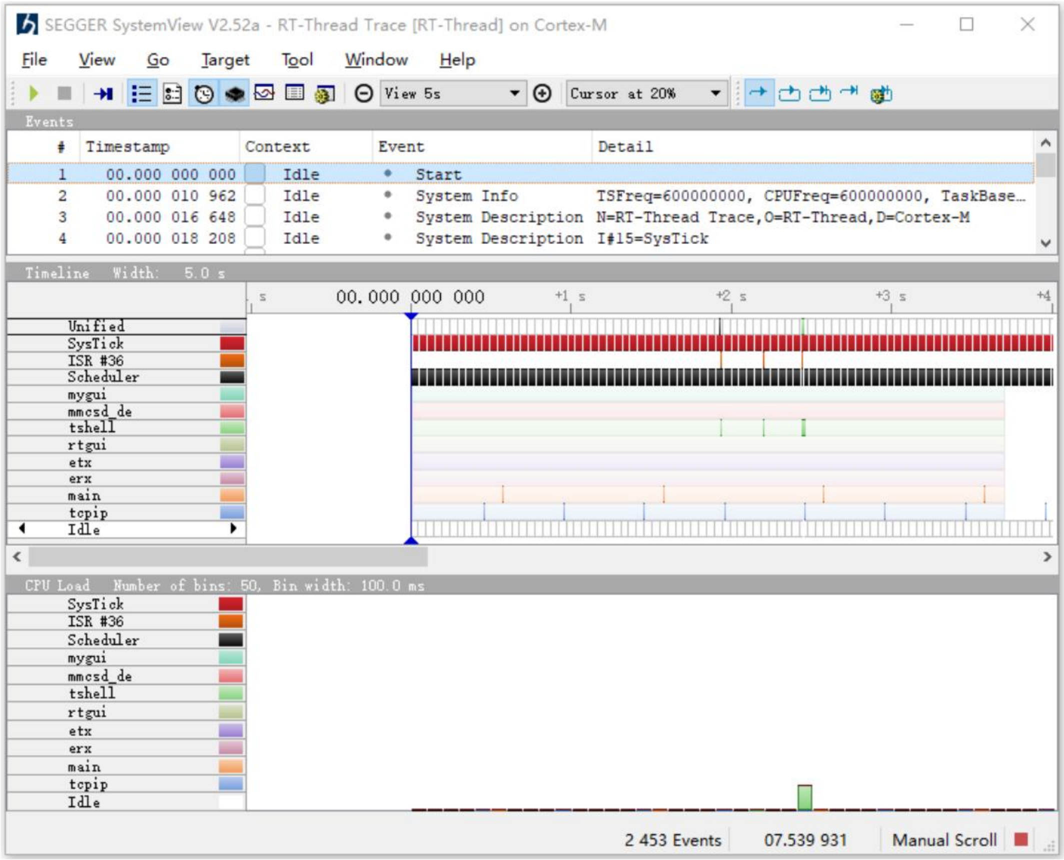

Figure 1: *SystemView* The main interface

3 SystemView Package on RT-Thread

3.1 Introduction

SystemView package on RT-Thread It is an embedded end program implementation of the SystemView tool. Its main functions include: configuring specific parameters of SYSTEMVIEW and RTT, collecting and formatting monitoring data, and sending data to the PC through J-Link. You only need to use the env tool launched by RT-Thread By enabling the SystemView software package and performing a simple configuration on it, you can complete the configuration of the SystemView embedded end program.

3.2 How to enable and configure options

Take the Zhengdian Atom RT1052 development board as an example

Step 1: Enter the menuconfig graphical configuration tool in the env tool

Open the env tool and use the command cd D:\rt-thread\bsp\imxrt1052-evk to switch to the RT-Thread source Code bsp imxrt1052-evk directory under the root directory, and then enter the command menuconfig to configure the project.

menuconfig is a graphical configuration tool that RT-Thread uses to configure and tailor the entire system.

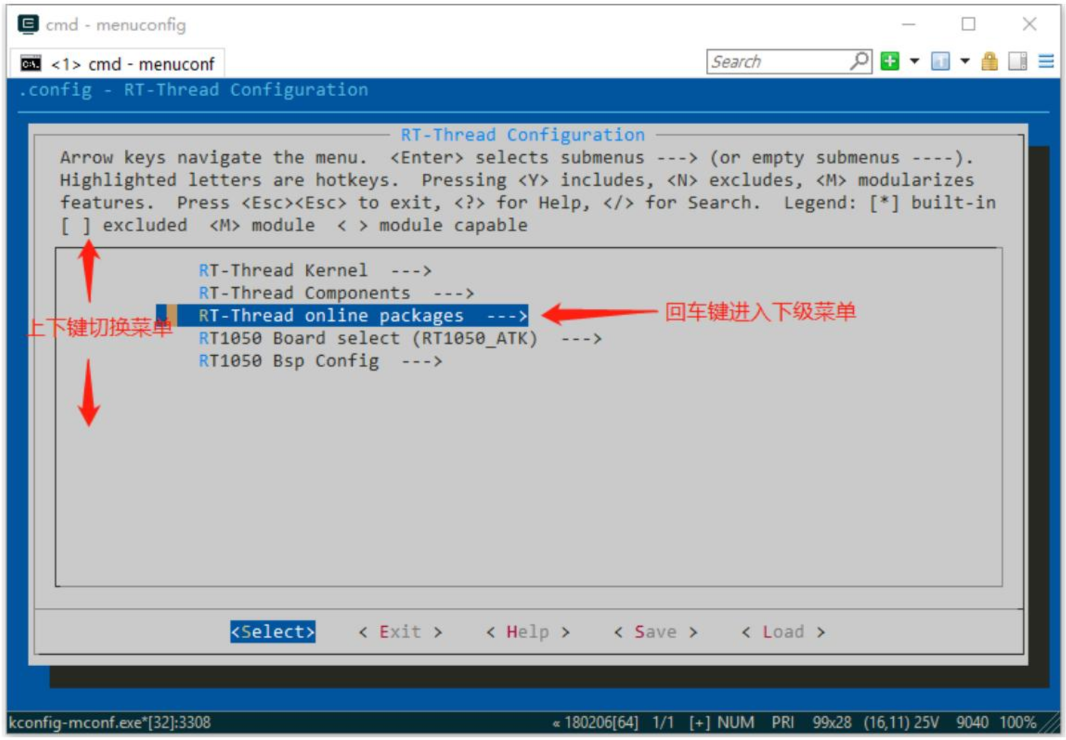

Figure 2: *menuconfig* Tool interface

Step 2: Enable the SystemView software package.

Use the up and down keys to select RT-Thread online packages, press Enter to enter the submenu, and then in tools

Open SystemView in packages.

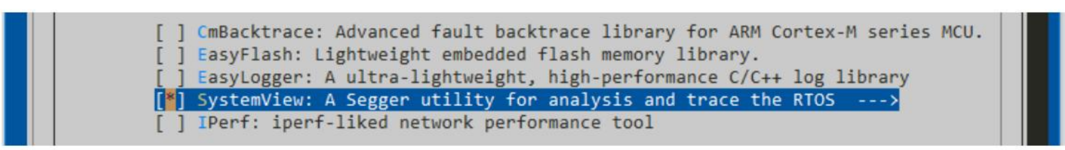

Figure 3: Enable *SystemView*

Step 3: Specific configuration.

Press the Enter key to enter the sub-menu and make specific configurations (enter? to display detailed information of the options).

The following are two common options. Other options can be configured as needed (refer to the last part of the

Number of directories):

1. Change the option circled in red in the figure below to the kernel corresponding to your development board, such as i.MX-RT1052 corresponding to M7 kernel.

nuclear

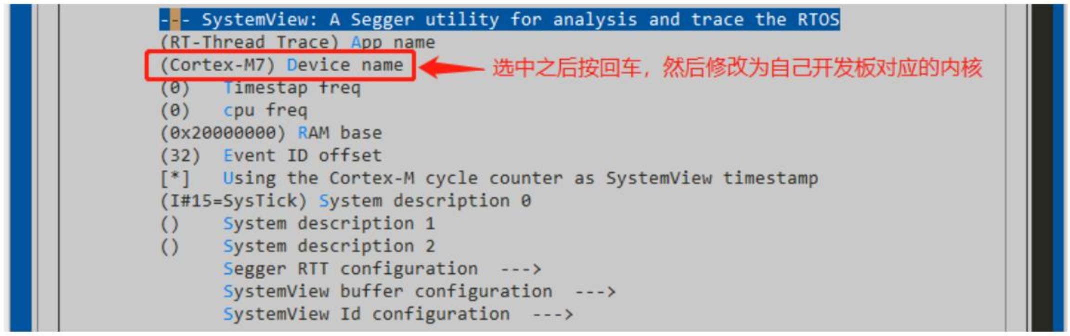

Figure 4: Specific configuration

2. Enter the SystemView buffer configuration submenu and turn off post-analysis mode

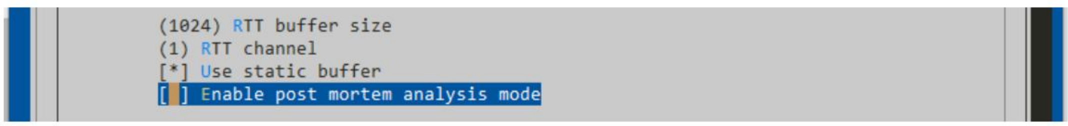

Figure 5: Turn off post-mortem mode

After enabling post-mortem analysis mode, call the SEGGER_SYSVIEW_Start() function in your project to start recording. System events are continuously recorded, overwriting older events when the buffer is full. Therefore, when reading from the buffer, only the most recent events are available.

Post-mortem analysis can be useful when a system has been running for a long time and suddenly crashes. In this case, the most recent events can be read from the target's buffer, and SystemView can show what happened just before the crash.

After closing the post-analysis mode, it will be in continuous recording mode, and you can control the start and end of recording on the PC. Continuously record the system operation. To analyze the operation of the following demo, you need to enable the continuous recording mode.

After configuring the options, press ESC to return, exit and save the configuration. This completes the enabling and related configuration of the SystemView software package.

4 Use of SystemView Tool

We use a specific demo to explain the use of the SystemView tool

4.1 Adding demo code

Add the following code to the main.c file, and then call the demo initialization function demo_init () in the main function to run the demo.

```
/*
* Program listing: systemview demonstration code
*
* In this example, a dynamic semaphore (initial value is 0) and two dynamic threads will be created.
        In thread
* Thread 2 will try to hold the semaphore by waiting forever, and send the running flag after holding it successfully. * Thread 1 will first send the running
flag, and then release the semaphore once. Because thread 2 has a higher priority, thread 1 will send the running flag first, and then release the semaphore once.
        Process 2 holds the semaphore and intercepts thread 1.
* Then after thread 2 sends the running flag, it cannot obtain the semaphore and is suspended, and thread 1 continues to run
*/

#define THREAD_PRIORITY 25
#define THREAD_STACK_SIZE 512
#define THREAD_TIMESLICE /* 5
Pointer to semaphore */
rt_sem_t sem_food; /*
Thread 1 entry*/
void thread1_entry(void* parameter) {

        while (1) {

                /* Thread 1 runs for the first
                time*/ rt_kprintf("thread1 is run!\n"); /* Release
                the semaphore once*/
                rt_sem_release(sem_food); /* Thread
                1 runs for the second time*/
                rt_kprintf("thread1 run again!\n"); /* Thread 1 delays
                for 1 second*/
                rt_thread_delay(RT_TICK_PER_SECOND);
        }

} /* Thread 2 entry*/
void thread2_entry(void* parameter) {

        while (1) {

                /* Try to hold the semaphore and wait forever until the semaphore is held*/
                rt_sem_take(sem_food, RT_WAITING_FOREVER); /* Thread 2
                is running*/
                rt_kprintf("thread2 is run!\n");
        }
}
/* DEMO initialization function*/
void demo_init(void) {
```

```
    /* Pointer to thread control block */
    rt_thread_t thread1_id, thread2_id; /* Create a
    semaphore, the initial value is 0 */
    sem_food = rt_sem_create("sem_food", 0, RT_IPC_FLAG_FIFO); if (sem_food ==
    RT_NULL) {

        rt_kprintf("sem created fail!\n"); return ;


    } /* Create thread 1 */
    thread1_id = rt_thread_create("thread1", thread1_entry,
                            RT_NULL,/* Thread entry is thread1_entry, parameter RT_NULL */

                            THREAD_STACK_SIZE, THREAD_PRIORITY, THREAD_TIMESLICE)
                                ;
    if (thread1_id != RT_NULL)
        rt_thread_startup(thread1_id); /* Create
    thread 2 */ thread2_id
    = rt_thread_create("thread2", thread2_entry, RT_NULL, /*
                            Thread entry is thread2_entry, parameter RT_NULL */

                            THREAD_STACK_SIZE, THREAD_PRIORITY - 1,
                                THREAD_TIMESLICE);
    if (thread2_id != RT_NULL)
        rt_thread_startup(thread2_id);
}
```

4.2 Configuration and Use of SystemView PC Program

Step 1: Download the SystemView analysis tool

Step 2: Add a system description file for RT-Thread

First, find the packages directory under the development board directory, and then find the segger_debug-xxx directory under the packages directory. There is a SystemView_Description folder in this directory, and the description file of the RT-Thread system is in it. The specific directory structure is as follows:

bsp\\your own development board\\packages\\segger_debug-xxx\\SystemView_Description\\
SYSVIEW_RT-Thread.txt

Copy this file to the Description directory under the SystemView tool installation directory, so that SystemView can identify the RT-Thread system.

Step 3: Configure device information and start recording

Double-click to open the SystemView PC program. Below is an introduction to the functions of some commonly used buttons on the main interface.

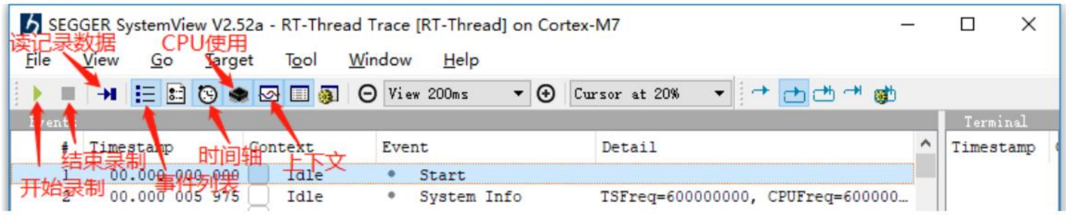

Figure 6: *PC* Terminal program button introduction

When in continuous recording mode, click the Start Recording button and a window will pop up to configure device information.

Figure 7: Window for configuring device information

The address of the RTT control block in the figure has been printed out through the serial port. Just open the terminal software and print the printed address.

Copy the address above.

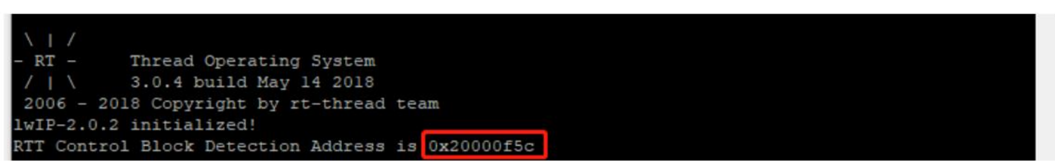

Figure 8: Get it from the terminal *RTT* Control block address

Click OK. Now SystemView starts recording system information in real time. Below is the real-time operation status of the system.

Machine Translated by Google

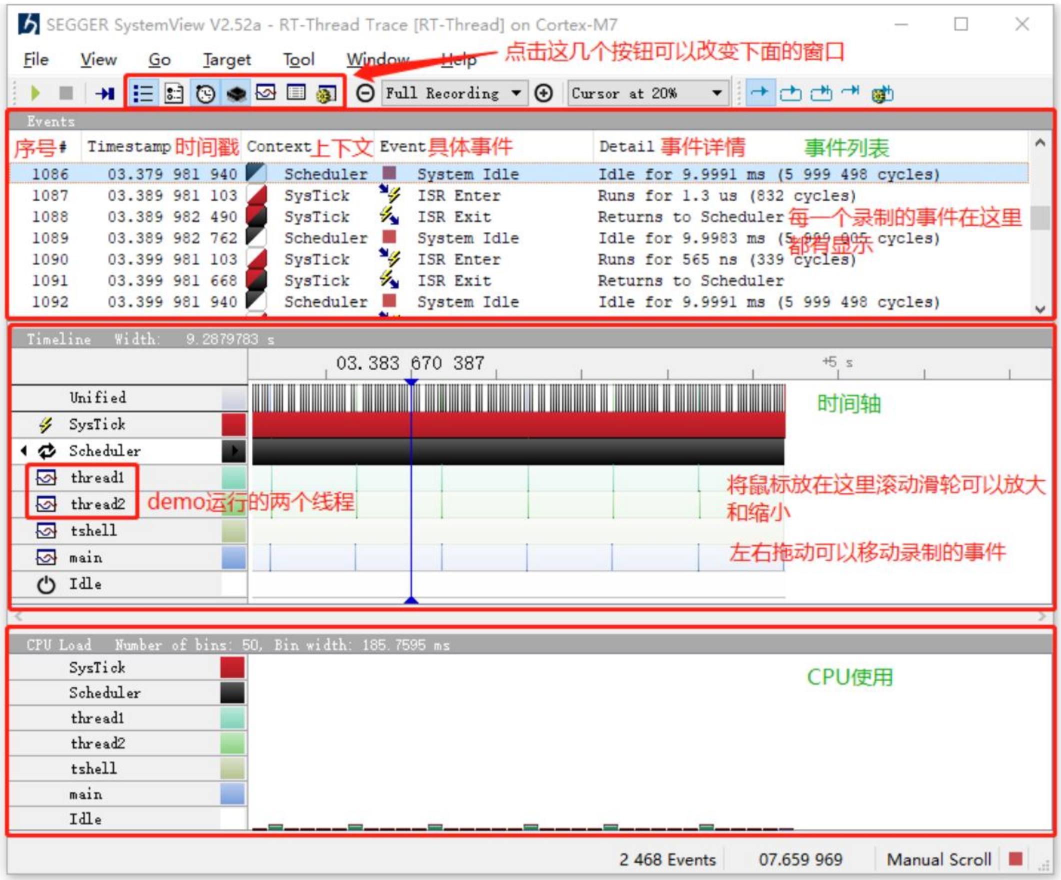

Figure 9: Real-time system operation status

Step 4: End recording and analyze the system

Click the End Recording button to end the recording. Place the mouse in the timeline window and use the scroll wheel to zoom in on the event to a suitable size.

Size of combined analysis

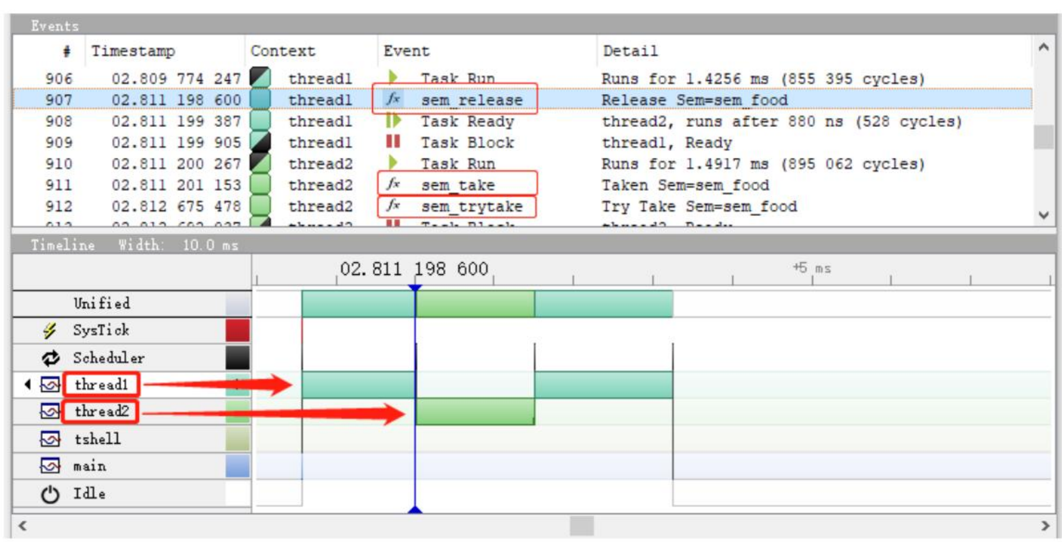

Figure 10: Analysis Thread 1、 2 run

Using the SystemView tool, we can see that the system is running as we expected. Thread 1 is running first. It starts running, and then is interrupted by thread 2 after releasing the semaphore. Then thread 2 runs, but cannot obtain the semaphore. From the above event list, we can also see the specific time when each thread acquires or releases the semaphore. time.

If the post-analysis mode is turned on, you cannot click the Start Recording button, but click Read Recorded Data button, and other operations are the same as those in real-time analysis mode.

After reading this document, I believe you have learned how to use the SystemView analysis tool on RT-Thread.

5 menuconfig parameter details

parameter	describe
App name	Application name
Device name	The kernel used by the device
Timestap freq	Timestamp frequency (0 means using the system default frequency)
cpu freq	CPU frequency (0 means using the system default frequency)
RAM base	RAM base address default value: 0x2000 0000
Event ID offset	Event ID offset default value: 32
Using the Cortex-M cycle …	Use system frequency as timestamp
System description 0-2	System descriptor "I#num=name, …" num is the interrupt number, name is Interrupt Name

parameter	describe
Max num of up buffer	Maximum number of RTT output buffers Default value: 2
Max num of downm buffers	Maximum number of RTT input buffers Default value: 2
buffer size up	The number of bytes used for the RTT endpoint output channel. Default value: 1024 bytes
buffer size down	Number of bytes used for RTT terminal input channel Default value: 16 words Festival
Segger RTT printf buffer size	The default buffer size when sending character blocks via RTT printf is Default value: 64
Mode for pre-initialized terminal channel	Pre-initial RTT terminal channel mode default value: NO_BLOCK_SKIP

parameter	describe
Max Interrupt priority	Maximum interrupt priority level
Use RTT ASM	Using the assembly version of RTT
memcpy uses byte-loop	Use a simple byte-loop instead of memcpy

parameter	describe
RTT buffer size	The number of bytes used by SystemView to log the buffer
RTT channel	RTT channel is used for SystemView event recording and communication, 0 table Automatic selection
Use static buffer	Using static buffers can save space
Enable post mortem analysis mode	Enable post-mortem analysis mode

parameter	describe
ID Base	The value subtracted from the ID recorded in the SystemView package. Default value: 0x1000 0000
ID Shift	The number of digits in the ID shift documented in the SystemView package. Default value: 2