SystemView 简易使用说明

0. 需要的工具

Jlink 仿真器 (接口 Jtag 或 SWD 均可),

能够正常运行的 Cortex M 单片机板及工程文件,

SystemView 软件包,目前最新的版本是 2.30。

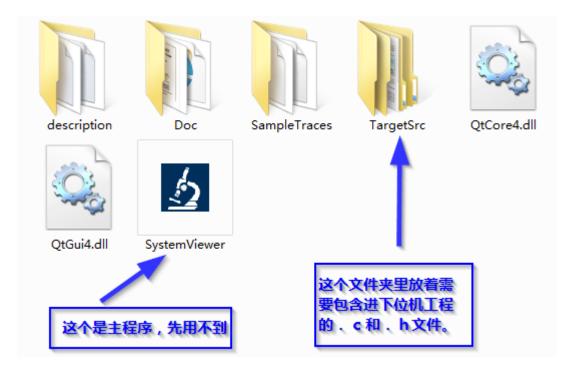
1. 去 Segger 官网下载软件包

网址: https://www.segger.com/downloads/free_tools#SystemView



本次使用了免安装绿色版。另外, SystemView 需要使用到 RTT 功能, Jlink 驱动版本太旧的话可能不支持,建议更新驱动,关于更新驱动的方法以及注意事项,网上资料很多,大家就自己查找了。

2. 软件包下载完成后解压缩,里面是这个样子:



需要添加到工程中的具体文件:

\TargetSrc\Config 目录下的文件。

\TargetSrc\SEGGER 目录下的文件。

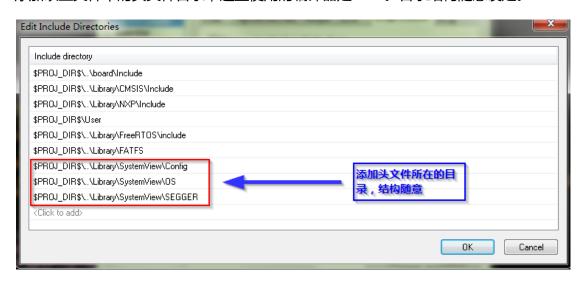
与下位机使用的操作系统相关的文件。本人使用系统的是 FreeRTOS 8.2.2,用到的是这几个文件:

\TargetSrc\Sample\Config 目录下的 SEGGER_SYSVIEW_Config_FreeRTOS.c 文件。

\TargetSrc\Sample\OS 目录下的 SEGGER_SYSVIEW_FreeRTOS.c 和 SEGGER_SYSVIEW_FreeRTOS.h 文件。

SystemView 使用的 ROM 和 RAM 不多,差不多只要下位机工程的存储空间之前没有塞满就能包含成功。

添加以上文件中的头文件目录,这里使用的编译器是 IAR。目录结构随意设定。



3. 修改下位机工程代码

3-1. 修改下位机工程主函数所在的文件中的代码。

在下位机系统运行之间初始化 SystemView,具体来说就是在系统运行之前增加以下语句: SEGGER_SYSVIEW_Conf();

同时还需要在文件头包含头文件 SEGGER_SYSVIEW.h。具体如下图(图片来自 SystemView 的说明文档,具体位置 \Doc\ UM08027_SystemView.pdf),图片中使用 的系统应该是 Segger 自己的 embOS,不过其他系统也大同小异。

```
#include "SEGGER_SYSVIEW.h"
        main()
* Function description
                                          在下位机工程中主函数所在的文
件加入左方红框中所示的代码
   Application entry point
int main(void) {
 OS IncDI();
                                        Initially disable interrupts
 OS_InitKern();
                                        Initialize OS
 OS InitHW();
                                      /* Initialize Hardware for OS
 BSP Init();
                                      /* Initialize BSP module
 SEGGER SYSVIEW Conf();
                                     /* Configure and initialize SystemView
  /* You need to create at least one task before calling OS Start() */
 OS_CREATETASK(&TCB0, "MainTask", MainTask, 100, Stack0);
                                     /* Start multitasking
 OS Start();
 return 0:
```

3-2. 修改 FreeRTOS 相关文件

在 FreeRTOS.h 文件的头部包含 SEGGER_SYSVIEW_FreeRTOS.h 文件,如下图中的红框:

(根据 UM08027_SystemView.pdf 5.4.2 描述,将 SEGGER_SYSVIEW_FreeRTOS.h 包含到 FreeRTOSConfig.h 文件末尾也可以,不过该方法本人在使用的时候编译器报错,于是就改用了上述方法。原因可能是软件版本的问题,比如 Segger 测试时使用的是FreeRTOS8.2.3,本人使用的是 8.2.2 版。)

在 FreeRTOSConfig.h 中添加两个宏 INCLUDE_xTaskGetIdleTaskHandle 和 INCLUDE_pxTaskGetStackStart ,根据 Segger 的说明。这一步是可选的。如下图中的红框所示:

下位机工程所需要的修改基本完成了。不过 SystemView 目前对 FreeRTOS 的支持不是很完善,默认情况下无法看到中断函数的运行情况。若需要查看中断函数的运行情况,还需要在中断函数中增加一些代码,如下图中的红框所示。(其中 SEGGER_SYSVIEW_Print 函数不是必需的,它的作用是向上位机发送一些提示,便于查看。)

根据 SystemView 的说明文档,如果希望 SystemView 查看到更多内容,需要对 FreeRTOS 的源代码做一些修改,修改方法保存在

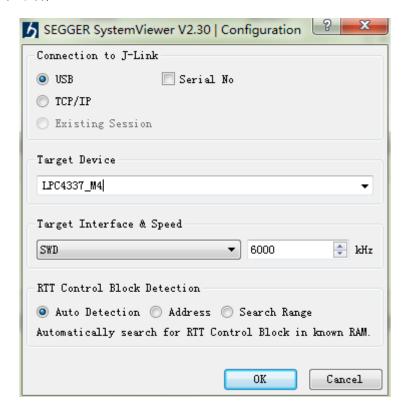
\TargetSrc\Sample\Patch\FreeRTOSV8.2.3_Core.patch 文件中。本人时间有限没有做相应修改,用兴趣的朋友可以试试。

4. 编译、下载代码到下位机并运行。

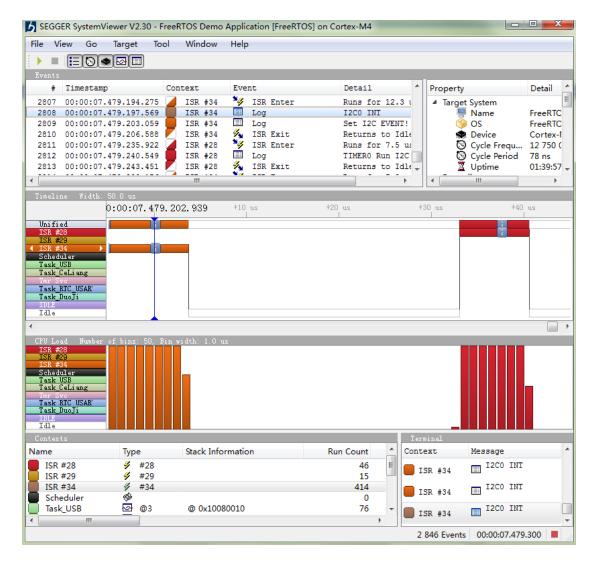
现在用的到主程序 SystemViewer.exe 了。双击运行,选择 Target -> Start Recording:



然后出现这个对话框:



检查一下 Jlink 的接口是否正确,其他选项一般不用修改。然后点击 OK。 然后就能看到 SystemView 跑起来了:



任务直接以名称显示出来,中断以 ISR # + 中断异常号的形式显示。在 SystemView 中可以很直观地看到事件发生的时间,顺序,还能像使用示波器一样对时间轴进行放大,非常方便。

除了任务和中断, SystemView 似乎还支持普通函数运行情况的查看, 不过本人还弄清怎么做, 望高手指点了。