1. 介绍

IAR 支持 SWO Trace 调试这一强大的调试功能。SWO Trace 利用 SWD 接口中的 SWO 串行线输出调试过程中的跟踪信息，可以在调试过程中输出 PC 指针采样、中断记录和数据记录等信息，帮助我们更好的分析程序运行的情况。

使用 SWO Trace 的硬件要求:

(1). 使用的仿真器支持 SWO 串行通信。

(2). 芯片支持 SWD 调试接口，且芯片 SWO 引脚有和硬件调试接口连接。

使用 SWO Trace 的软件设置:

1、 调试器接口应该选择使用 SWD 接口进行连接。

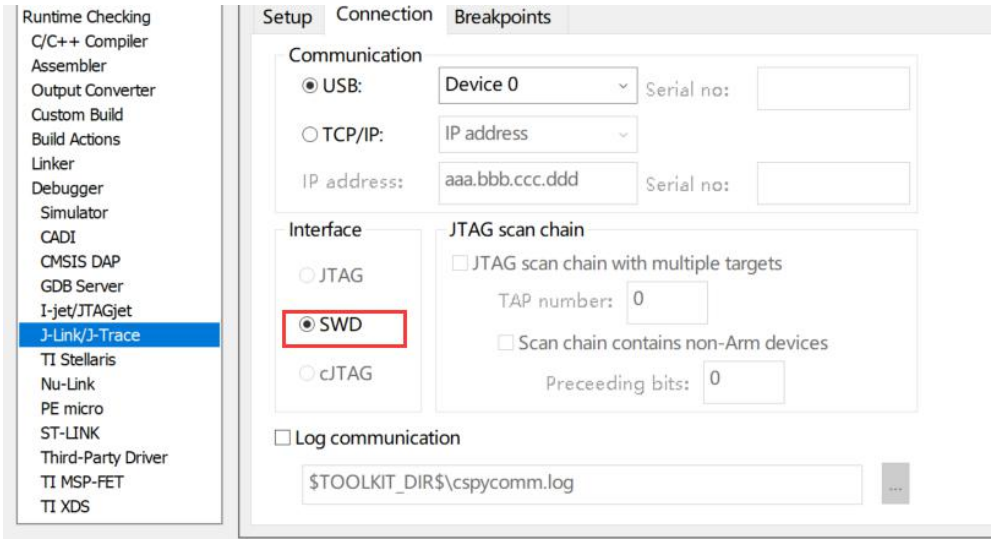

图 1-1

IAR 支持的 SWO Trace 调试的具体功能，主要有如下几个:

a.Data logs　　数据记录功能，记录静态/全局变量的值变化

b.Interrupt logs　　记录中断相关信息，如中断进入和退出时间，中断执行时间等

c.ITM logs　　通过 ITM 端口实时输出信息

d.PC sampling　　PC 指针采样，包括函数覆盖率分析和函数执行时间占比分析等

e.Timeline　　通过图形化的方式显示 Data logs、Interrupt logs 和 ITM logs 的信息

下面基于 stm32f401 开发板的例程来具体介绍这些调试功能。

1.1 Data logs 功能

1.1.1 变量数据变化和图形化显示

(1) 打开 EWARM Training project\SWO_Trace-data_logs\ SWO_Trace-data_logs.eww。

要使用 Data logs 功能，首先要将全局/静态变量，或者内存区域添加为一个 Data log 断点。这里我们添加全局变量 TestPoint 为 Data log 断点，添加断点后，还需要在 breakpoint 断点窗口进行使能。

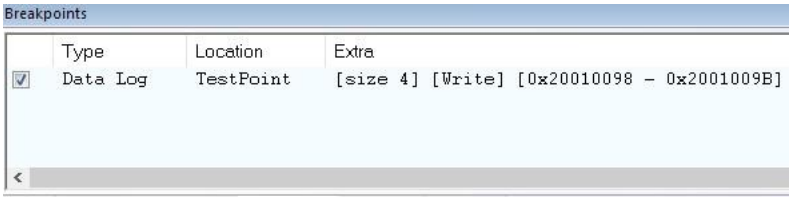

图 1-2

(2)选中上图中的断点，按右键， 弹出断点编辑窗口，对 Testpoint 进行编辑，将访问类型设置为“Write”，当 TestPoint 变量的值被执行写操作的时候就会触发记录。

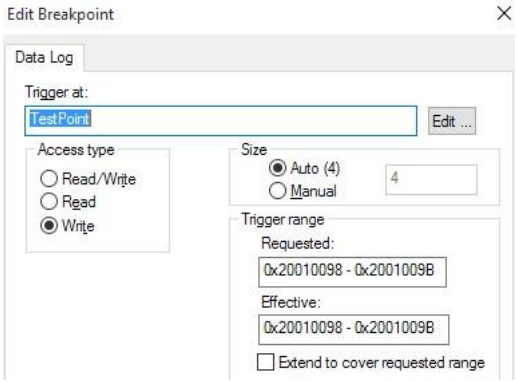

图 1-3

(3). 从对应的仿真器菜单下打开 Data log 窗口，此例中我们使用的是 ST-Link，单击右键“Enable”启用数据跟踪，然后运行程序，在窗口中会看到显示的变量 TestPoint 的跟踪记录。

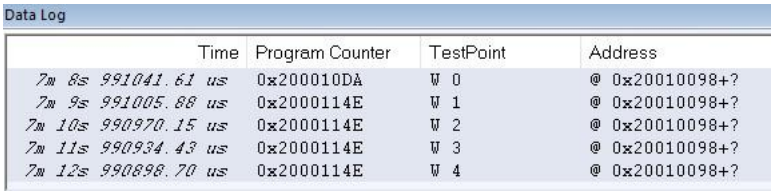

图 1-4

(4).从 ST-Link 菜单栏下打开 Timeline 窗口，在 Data log 栏右键选择“Enable”使能图形化显示。选中下图中的区域，按右键，选择 Style->Levels 设置图形化显示的方式。

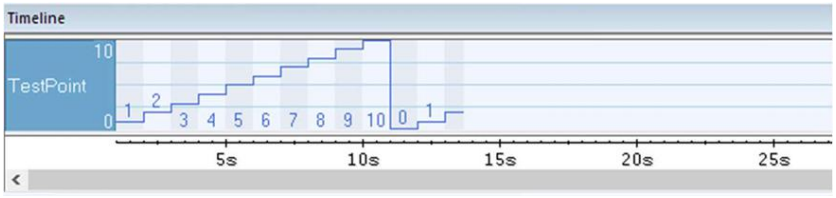

图 1-5

1.1.2 计算函数执行时间

(1). 打开 EWARM Training project\SWO_Trace-data_logs_TimeStamp\ SWO_Tracedata_logs_TimeStamp.eww。

例程定义了一个 TimeStamp 静态变量，分别在数组排序函数前后对其赋值两次，并且将 TimeStamp 设置为 Data log 断点，写触发。运行程序记录跟踪结果，将用于测量 NumberSorting()函数的执行时间。

```
TimeStamp = 0x05;
/*Sort an array*/
NumberSorting(arry,  IndexLength);
TimeStamp = 0x0A
```

在 main.c 的 137-140 行

Data Log

Time	Program Counter	TimeStamp	Address
1s 20992.01 us	0x200010FC	W 0x00000005	@ 0x2001009C+?
1s 21576.23 us	0x2000110A	W 0x0000000A	@ 0x2001009C+?
2s 20956.29 us	0x200010FC	W 0x00000005	@ 0x2001009C+?
2s 21540.50 us	0x2000110A	W 0x0000000A	@ 0x2001009C+?

图 1-6

如上图所示，数据跟踪记录了 TimeStamp 这个变量值的变化情况，该变量两次赋值语句中间调用了排序函数，我们根据该变量值的变化即可计算出函数的执行时间。

根据跟踪的结果计算，TimeStamp 从 0x05 变化到 0x0A 所用的时间：

t = 21576.23 - 20992.01 = 584.22us

由此可以知道排序函数 NumberSorting()执行所用的时间为 584.22us。

1.2 Interrupt logs

(1)打开 EWARM Training project\SWO_Trace-interrupt_logs\ SWO_Trace-interrupt_logs.eww。

从 ST-Link 菜单下打开 Interrupt log 窗口，然后单击右键选择“Enable”使能，运行程序，系统发生的中断都会被记录。我们可以看到中断进入和退出的时间，以及中断服务函数执行的时间。

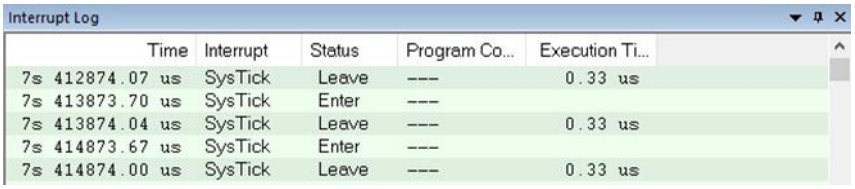

图 1-7

(2).从 ST-Link 菜单下打开 Interrupt log summary 窗口，窗口从左往右依次记录了中断发生的次数 Count，第一次中断发生的时间 First Time，中断服务函数总消耗的时间和及所占程序行时间的百分比等。

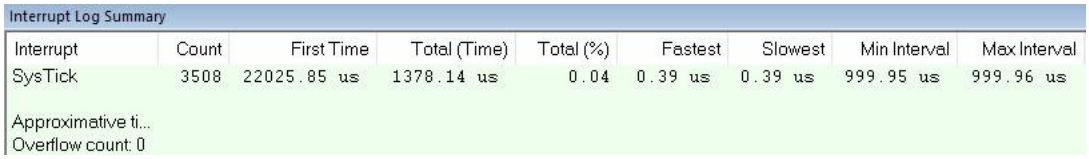

图 1-8

(3). 打开 Timeline 可视化调试，使能中断跟踪显示，在时间轴上可以显示中断的发生情况，如下图中所示中断的执行时间为 0.33us，然后按键盘的“+”“-”可以放大和缩小时间轴的分辨率。

此处我们只有一个 SysTick 中断，当存在多个中断时，我们可以通过该时间轴方便的分析中断执行的情况以及对中断嵌套进行分析。

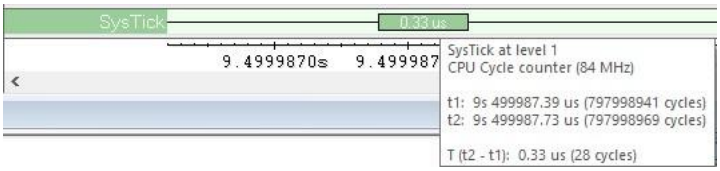

图 1-9

1.3 ITM 调试信息输出

打开 EWARM Training project\SWO_Trace-ITM_event_logs\ SWO_Trace-ITM_event_logs.eww。

1.3.1 使用 Terminal IO 打印调试信息

(1). IAR 提供的 Terminal IO 功能可以将函数内 printf 语句打印的信息在 Terminal IO 窗口中显示出来，printf 信息的输出有两种方式，半主机模式和 SWO 模式，半主机模式时信息的输出速率较慢，当我们使用 SWO 模式进行信息输出时，速度很快，适用于有大量信息需要输出的时候。在 General Options ->Library Configuration 选项中勾选使用 SWO 模式，Terminal IO 窗口可在进入调试后通过 View 菜单打开。

图 1-10

(2). SWO 模式输出 printf 信息使用的是 ITM 的 32 通道中的通道 0，我们需要在 SWO Configuration 选项中使能该通道。

1.3.2 ITM 输出信息和图形化显示

(1). C-SPY 调试器中的 Event log 和 Timeline 功能可以使用 ITM Port #1~#4 共四个通道，其他的通道可以将信息保存成 log 文件使用。实验代码 main 函数中有下列所示的两行代码，第一行是使用 ITM Port1 和 Port2 输出 32 位的 TestPoint 的值。

```
__ITM_EVENT32(1,  TestPoint);
__ITM_EVENT32(2,  TestPoint);
```

在 main.c 的 129-130 行

首先在 SWO Configuration 里面启用 ITM Port#0 ~ Port#3，再从 ST-Link 菜单打开 Event log 窗口，并单击右键，选择使能 Event log 功能。运行程序，在窗口内记录了 ITM1 和 ITM2 的事件，我们可以从 Event log 窗口中看到 ITM 端口输出的信息。

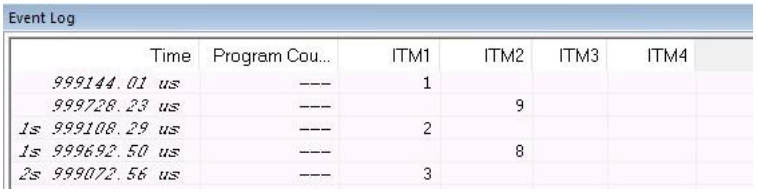

图 1-11

(2). 打开 Timeline，并使能 ITM 选项，在 ITM 显示栏上单击右键，从弹出的二级菜单里选择“Style”> Levels，让图形以水平的方式显示。然后运行程序，下图中以水平的方式显示了 ITM 端口输出的信息。

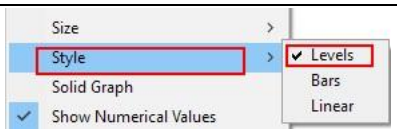

图 1-12

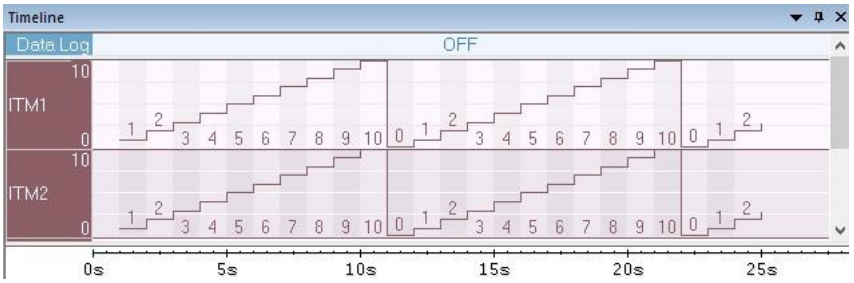

图 1-13

1.3.3 ITM 计算函数执行时间

(1).可以在一段代码的前后分别通过 ITM 输出一个值,通过两次 ITM 事件之间的时间差值就可以计算出这之间代码的运行时间。例程在 GenerateRandomNumbe()前后通过 ITM3 输出两个数据:

```
__ITM_EVENT32(3， 0x05);
/*Generate random Numbers*/
GenerateRandomNumber(arry， IndexLength);
__ITM_EVENT32(3， 0x0A);
```

在 main.c 的 132-135 行

通过 Event log 的记录，ITM3 输出 0x05 至 0x0A 输出的时间间隔

t = 20853.33 – 20278.73 = 574.6us

所以 GenerateRandomNumber()函数的执行时间为 574.6us。

1.4 PC sampling

(1). 打开 EWARM Training project\SWO_Trace-PC_sampling\ SWO_Trace-PC_sampling.eww。

进入调试模式后， 从 ST-Link->SWO Configuration 设置 PC 的采样率和系统时钟频率，同时关掉不使用的窗口和 ITM 端口，只留下 port 0，这些设置将减少跟踪产生的事件数据量，因为 stm32f401 开发板板载的 ST-LINK 传输速率有限，过多的事件来不及传输，会引起溢出。

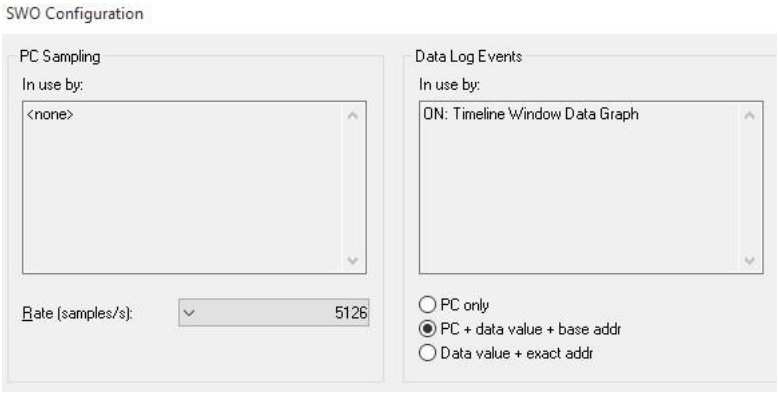

图 1-14

(2). 从 ST-Link 打开 Function Profiler 窗口，然后右键选择 Enable 使能，运行程序。让程序跑一会后，再手动暂停运行，当程序停止运行后，Function Profiler 窗口会显示程序运行这段时间各函数的分析结果。该分析结果是基于对 PC 指针采样得出的，受限于采样频率，部分执行次数较少的函数可能未被采样到，在分析结果中显示的数据为 0，但这并不是代表该函数未被执行，而是未被采样到。

该分析结果体现了各函数执行时间占比的情况，我们可以依据该分析结果对执行时间较长的程序进行针对性的优化，提高整个系统的运行效率。

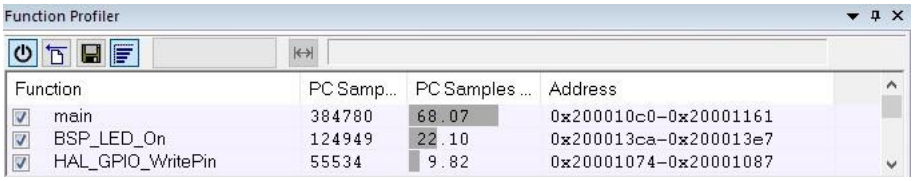

图 1-15