Debugger

PowerWriter 内置 Cortex-M Debugger 功能，支持主流的通用MCU，并保持不断的升级和维护，本节介绍调试器的具体使用方法。

准备工作

Power Writer ARM Debugger 基于 DAP 调试接口开发，支持Power Writer 适配的全品牌的芯片，集成 ITM Trace，最大时钟速度10Mhz，请留意以下信息：

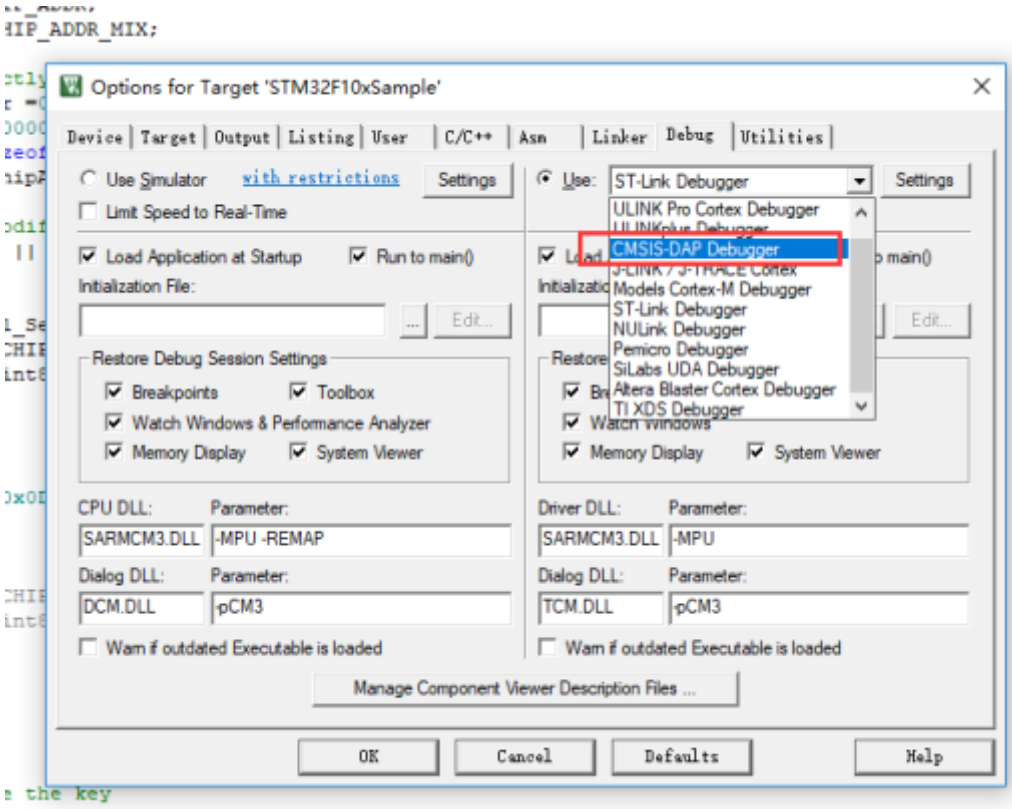
- Power Writer 集成的Debugger 为免驱 (HID设备)，但是在部分系统上，可能存在偶尔驱动异常的问题，在设备管理器中检查Power Writer 驱动是否正常，如驱动出现异常，请参考 驱动常见问题，进行排查。
- Power Writer 在Debugger 模式下，可以不断开PowerWriter 的连接，PowerWriter 会自动在Debugger 形态或者是Writer 形态下进行切换，在Debugger 模式下，PowerWriter 中额外提供的串口助手，此时可以当做日志打印，命令输入端口，或者使用Trace 功能，在MDK 中虚拟串口1中输入输出命令，关于这部分的功能请参考MDK 的官方文档：CMSIS-DAP Debugger User's Guide: About CMSIS-DAP (keil.com)。

PowerWriter Debugger For MDK

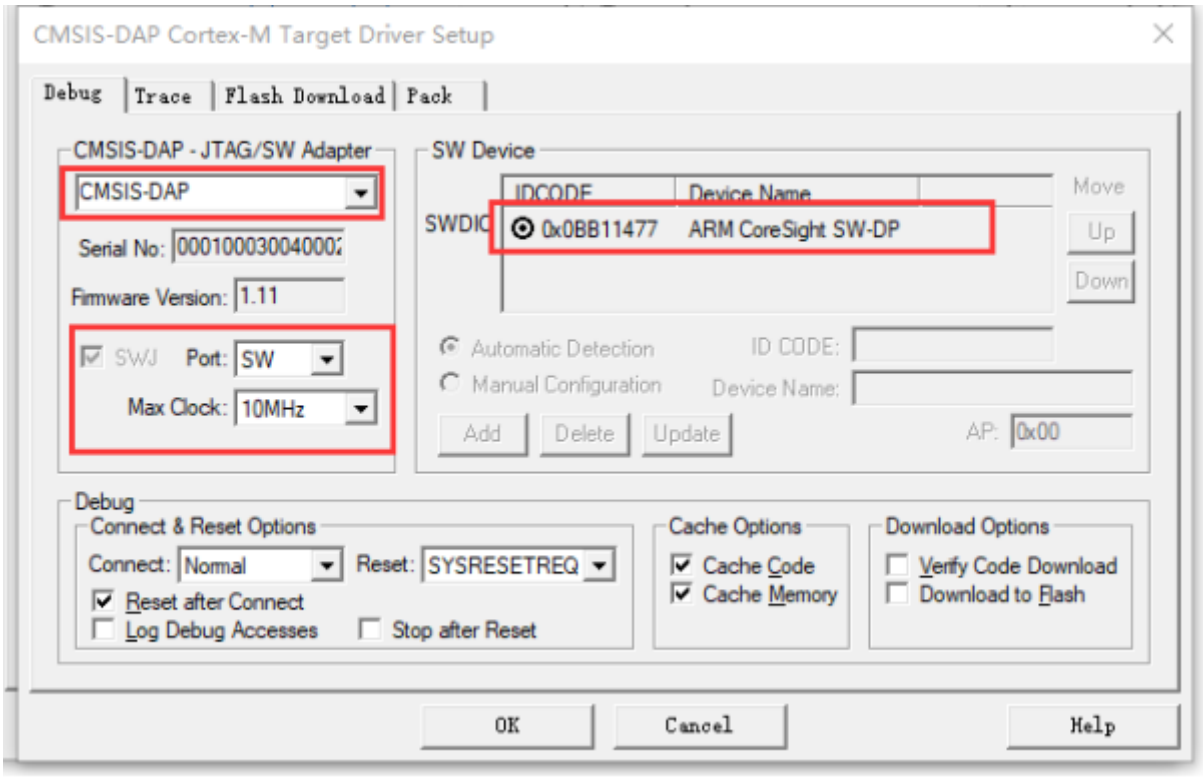
本节描述PowerWriter Debugger 在MDK 的应用方法。

PowerWriter Debugger For MDK 调试

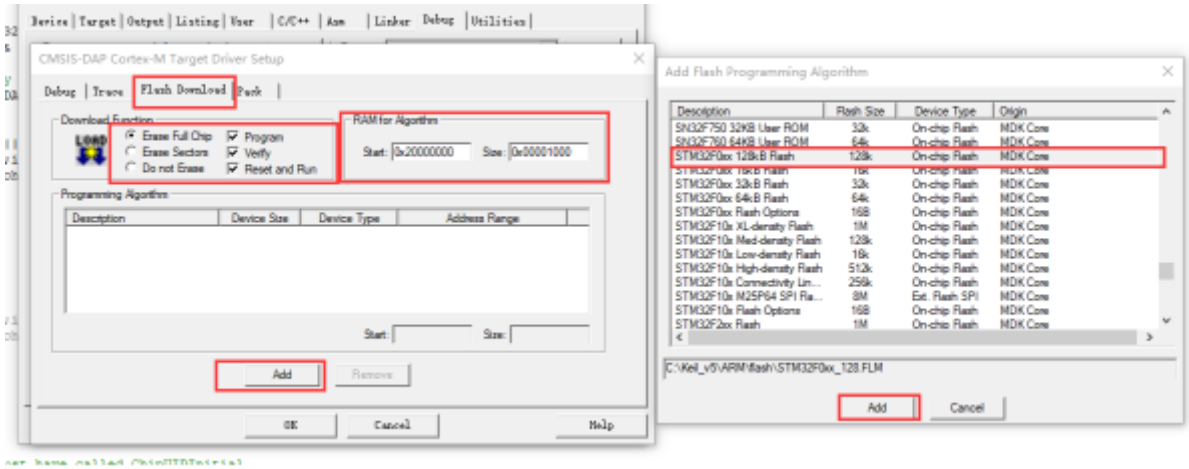
连接好目标板之后，在调试器选择页面选择 **CMSIS-DAP Debugger**作为调试器，如下图所示：



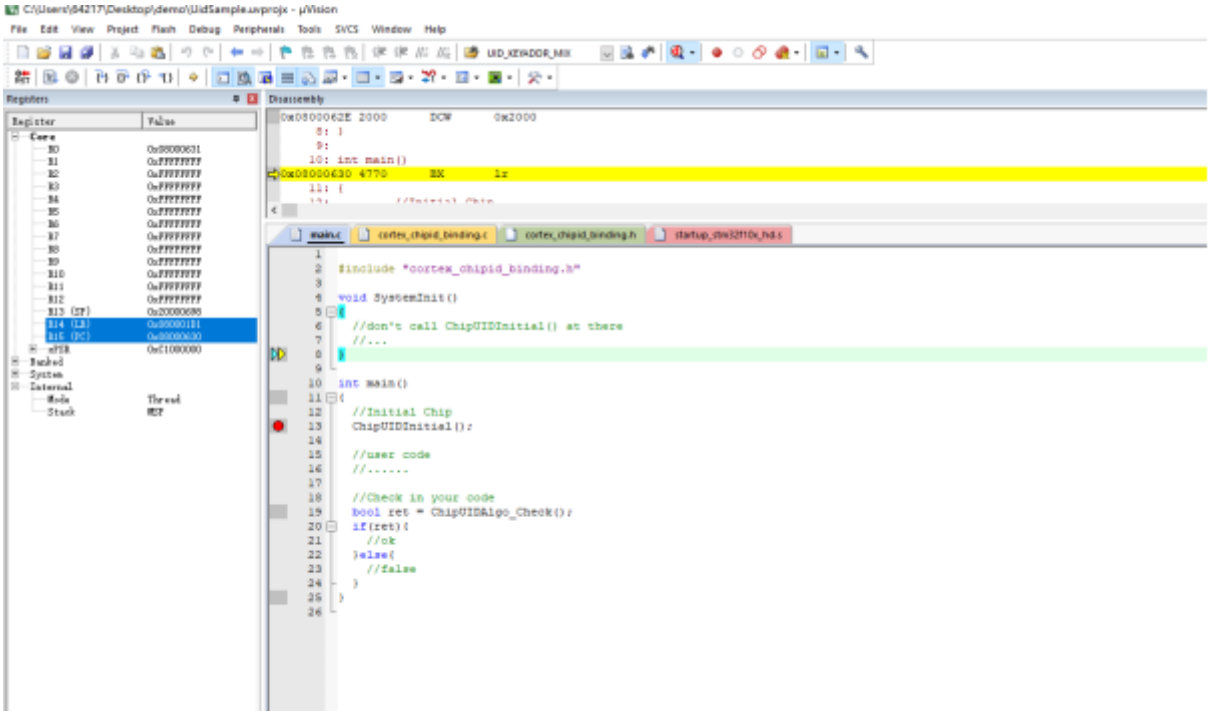
接下来点击设置页按钮，进入Debugger 的设置页，如图所示，从左侧列表选择已经连接好的CMSIS-DAP ，选择Port 为 **SW** 模式，选择 CLOCK，最高可以选10M，其余保留默认。



最后一步，需要添加 Debugger 所需的Flash Algorithm,点击Flash Download 标签页进入到下载配置页面，点击ADD 按钮从列表中选择对应芯片的Flash Algorithm，如图所示：



当添加好Flash Algorithm 之后，点击确定关闭窗口，回到MDK 主页面，点击工具栏的  按钮或者键盘的 **CTRL + F5** 即可进入Debug模式，如图所示，进入调试功能后，Power Writer 的 Debug 功能和 STLINK 、NULINK、GDLINK、J-LINK 功能一致。



> **注意**
>
> - 如调试器设置页面无法看到目标芯片ID，或者是提示错误，请参考 调试器常见问题 进行问题排查，确保目标芯片能正常被检测。
> - 其他IDE的应用方式和MDK 基本一致，比如CUBEIDE,IAR等IDE 工具。
> - 如果在Flash Algorithm找不到指定芯片的Algorithm 项，需要安装芯片对应的 **DFP** 包，下载方式包括：
>   - 使用MDK 自带的 pack installer： ，

  - 从MDK 设备适配包站点下载：MDK5 Software Packs (keil.com)
  - 从芯片厂家官方渠道获取 DFP 包
- 部分旧芯片型号 DFP 包可能已经停止更新和维护，可以从 MDK v4 Legacy Support (keil.com)处下载安装。

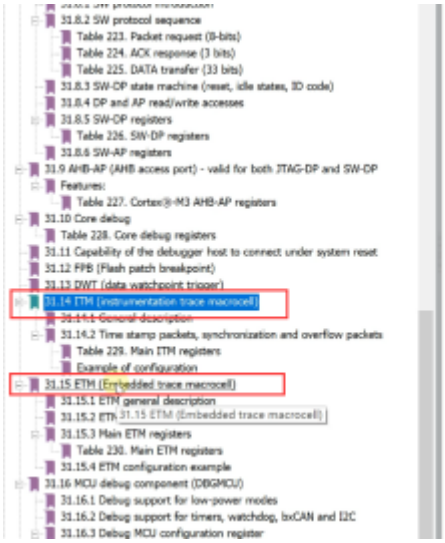
PowerWriter Debugger For MDK Trace

Power Writer 集成的Debugger 除了调试器之外，也集成了ITM Trace 功能，此功能可以代替串口等工具，跟MDK 配合使用将超过串口的实用性，在Power Writer的教学视频有完整的介绍如何使用Power Writer Debugger ITM Trace 功能，本节只介绍概要性的功能，支持ITM Trace 功能的芯片内核如表 4.1.1.3 所示：

内核版本	是否支持ITM
cm3、cm33	**支持ITM Trace**
cm35+	
cm4	
cm55	
cm7	
armv8mml	
armv81mml	
sc300	
Cm0	***不支持ITM Trace***
Cm0+	
Cm1	
Cm23	

内核版本	是否支持ITM
Armv7	
Armv8	
Armv8mbl	
Sc000	
...	其他内核请自行查阅官方文档手册

查询当前芯片是否支持ITM：可以通过当前用户手册进行查询，如下图所示：



The ITM is an application-driven trace source that supports *printf* style debugging to trace *Operating System* (OS) and application events, and emits diagnostic system information. The ITM emits trace information as packets which can be generated as:

- **Software trace.** Software can write directly to the ITM stimulus registers to emit packets.
- **Hardware trace.** The DWT generates these packets, and the ITM emits them.
- **Time stamping.** Timestamps are emitted relative to packets. The ITM contains a 21-bit counter to generate the timestamp. The Cortex®-M3 clock or the bit clock rate of the *Serial Wire Viewer* (SWV) output clocks the counter.

The packets emitted by the ITM are output to the TPIU (Trace Port Interface Unit). The formatter of the TPIU adds some extra packets (refer to TPIU) and then output the complete packets sequence to the debugger host.

The bit TRCEN of the Debug Exception and Monitor Control Register must be enabled before programming or using the ITM.

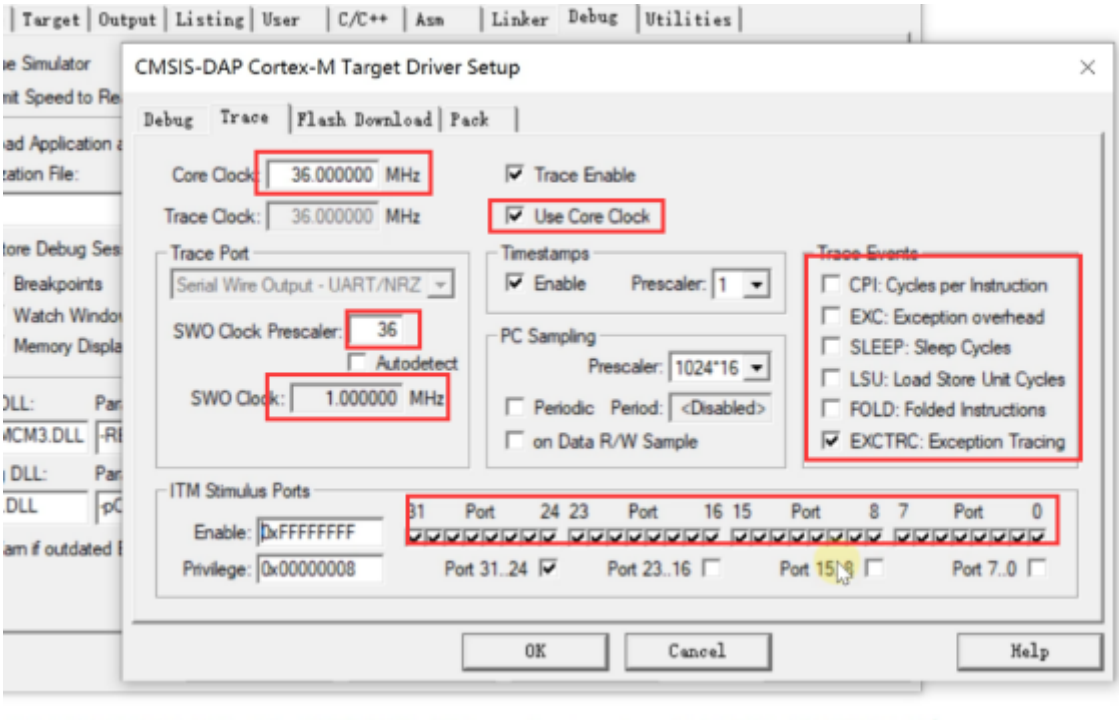
31.14.2 Time stamp packets, synchronization and overflow packets

Time stamp packets encode time stamp information, generic control and synchronization. It uses a 21-bit timestamp counter (with possible prescalers) which is reset at each time stamp packet emission. This counter can be either clocked by the CPU clock or the SWV clock.

A synchronization packet consists of 6 bytes equal to 0x80_00_00_00_00_00 which is emitted to the TPIU as 00 00 00 00 00 80 (LSB emitted first).

A synchronization packet is a timestamp packet control. It is emitted at each DWT trigger.

ITM 的开启方法：在调试器设置页Trace 设置页面，进行开启， 设置好相关参数，如下图所示：



Trace 功能设置页面：

- **Core Clock**：填写芯片的实际时钟。
- **Trace Enable**：开启Trace 功能。
- **Use Core Clock**：使用内核时钟。
- **Clock Prescaler预分频**：推介设置为和内核时钟一致，过高的速度，可能会导致缓冲区溢出(PowerWriter内置Cache为4K）。
- **端口部分**：按照实际开启：具体查阅芯片手册。
- **Trace Events**：可按照实际的需求，开启时事件追踪。

Trace 功能支持输入输出，需要在项目中，代理fputc 和fgetc 接口，最基本的标准输入输出重载代码，如下所示：

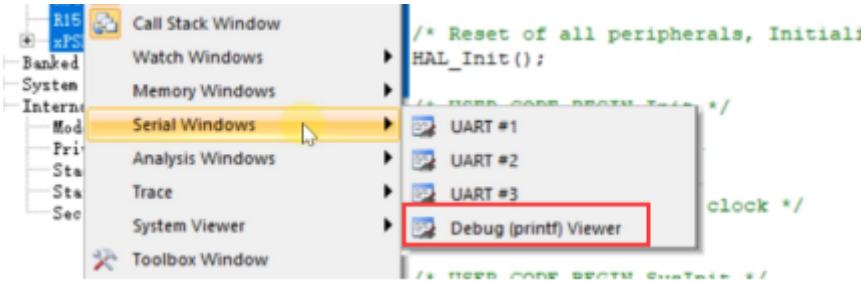
```
int fputc(int ch,FILE *f){
    return ITM_SendChar(ch);
}

volatile int32_t ITM_RxBuffer = ITM_RXBUFFER_EMPTY;

int fgetc(FILE * f){
    while(ITM_CheckChar()!=1) __NOP();
    return (ITM_ReceiveChar());
}

int ferror(FILE *f){
    return EOF;
```

```
}

void _ttywrch(int c){
    return fputc(c,&__stdout);
}

int __backspace(){
    return 0;
}

void _sys_exit(int code){
    label:
    goto label;
}
```

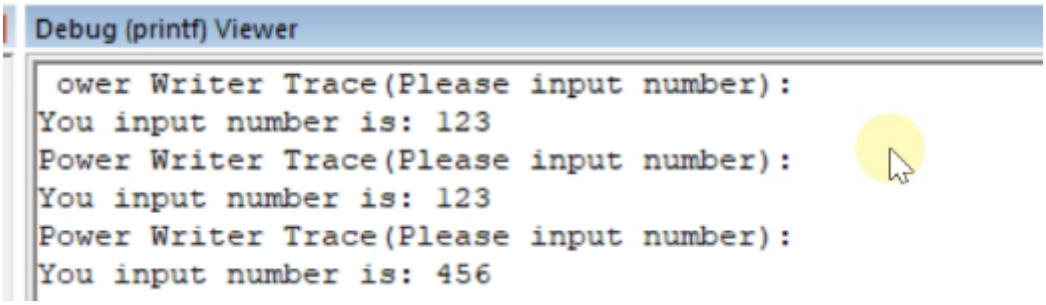
添加终端托管代码之后，即可正常执行打印日志，和输入信息，如下所示：

```
while(1){
    printf("PowerWriter Trace (please input number): ");
    scanf("%d",&input);
    printf("\r\nYou input number is: %d",input);
}
```

MDK Trace 菜单入口如图所示：

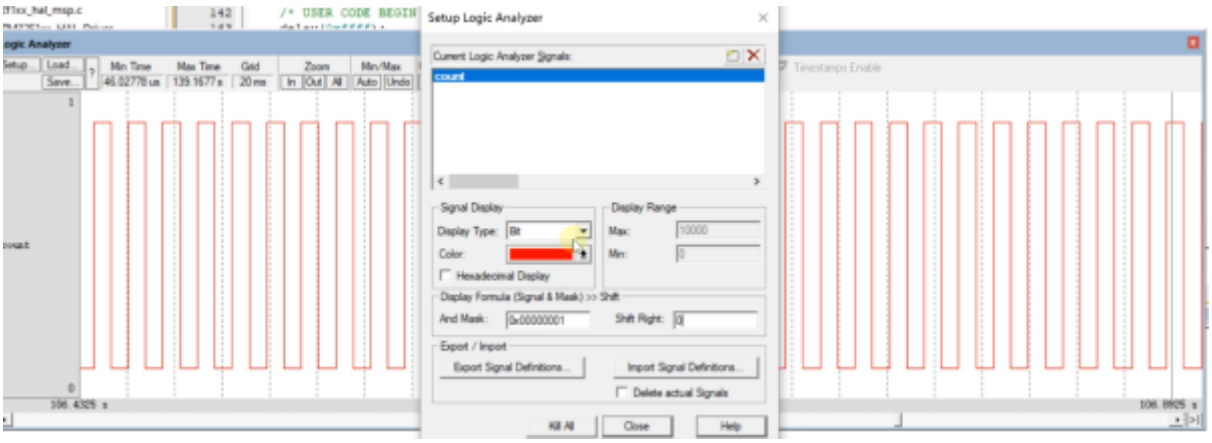


MDK Trace 演示效果如图所示,可代替串口使用：

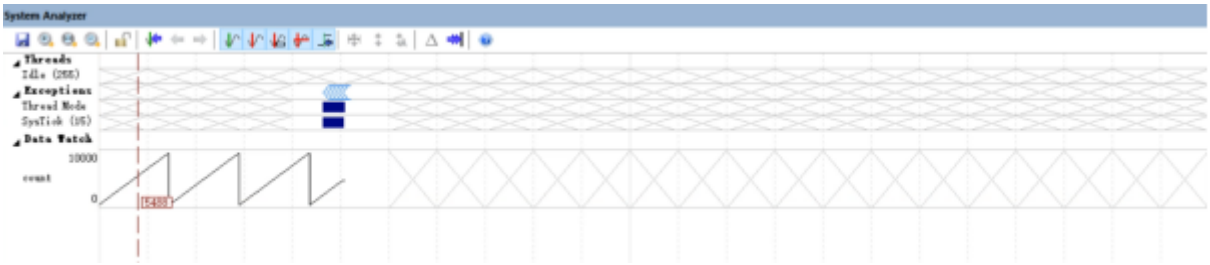


Trace 功能除了能作为输入输出使用之后，还可以使用MDK 自带的虚拟示波器，事件追踪等功能，具体请查阅视频教程，以及相关demo 资料，以下列举出Trace 功能比较常见的高级应用：

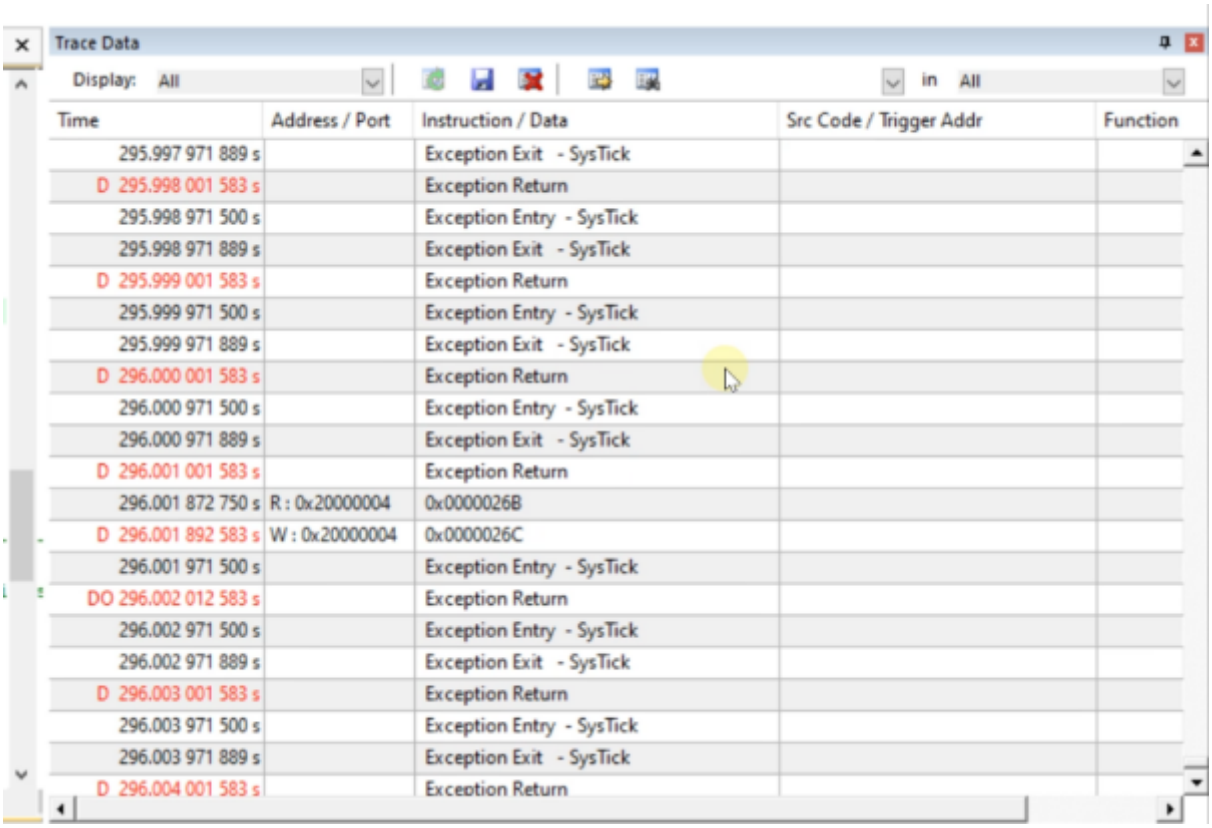
Power Writer Trace 逻辑分析仪：



Power Writer Trace 系统分析：

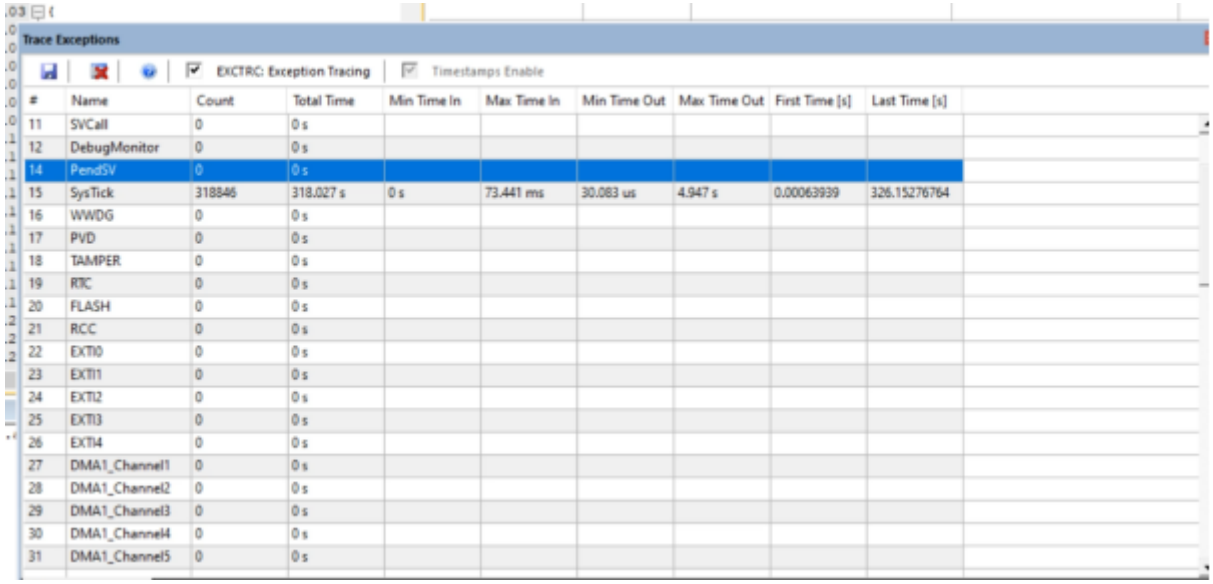


Power Writer Trace 系统Trace 记录：



Time	Address / Port	Instruction / Data	Src Code / Trigger Addr	Function
295.997 971 889 s		Exception Exit - SysTick		
D 295.998 001 583 s		Exception Return		
295.998 971 500 s		Exception Entry - SysTick		
295.998 971 889 s		Exception Exit - SysTick		
D 295.999 001 583 s		Exception Return		
295.999 971 500 s		Exception Entry - SysTick		
295.999 971 889 s		Exception Exit - SysTick		
D 296.000 001 583 s		Exception Return		
296.000 971 500 s		Exception Entry - SysTick		
296.000 971 889 s		Exception Exit - SysTick		
D 296.001 001 583 s		Exception Return		
296.001 872 750 s	R : 0x20000004	0x0000026B		
D 296.001 892 583 s	W : 0x20000004	0x0000026C		
296.001 971 500 s		Exception Entry - SysTick		
DO 296.002 012 583 s		Exception Return		
296.002 971 500 s		Exception Entry - SysTick		
296.002 971 889 s		Exception Exit - SysTick		
D 296.003 001 583 s		Exception Return		
296.003 971 500 s		Exception Entry - SysTick		
296.003 971 889 s		Exception Exit - SysTick		
D 296.004 001 583 s		Exception Return		

Power Writer Trace 系统Trace 模块追踪：



#	Name	Count	Total Time	Min Time In	Max Time In	Min Time Out	Max Time Out	First Time [s]	Last Time [s]
11	SVCall	0	0 s						
12	DebugMonitor	0	0 s						
14	PendSV	0	0 s						
15	SysTick	318846	318.027 s	0 s	73.441 ms	30.083 us	4.947 s	0.00063939	326.15276764
16	WWDG	0	0 s						
17	PVD	0	0 s						
18	TAMPER	0	0 s						
19	RTC	0	0 s						
20	FLASH	0	0 s						
21	RCC	0	0 s						
22	EXTI0	0	0 s						
23	EXTI1	0	0 s						
24	EXTI2	0	0 s						
25	EXTI3	0	0 s						
26	EXTI4	0	0 s						
27	DMA1_Channel1	0	0 s						
28	DMA1_Channel2	0	0 s						
29	DMA1_Channel3	0	0 s						
30	DMA1_Channel4	0	0 s						
31	DMA1_Channel5	0	0 s						

注意

- ITM (SWO)，需要芯片有ITM 模块，比较常见的CM3，具体请看Trance 功能中的列举。
- IAR For ARM 中的Trance 功能为**ETM**，PowerWriter 接口引脚不够暂未支持，请关注后续产品。
- ITM 暂时只支持MDK使用。
- LINK 类型的产品没有ITM Trace 功能，请使用串口代替。

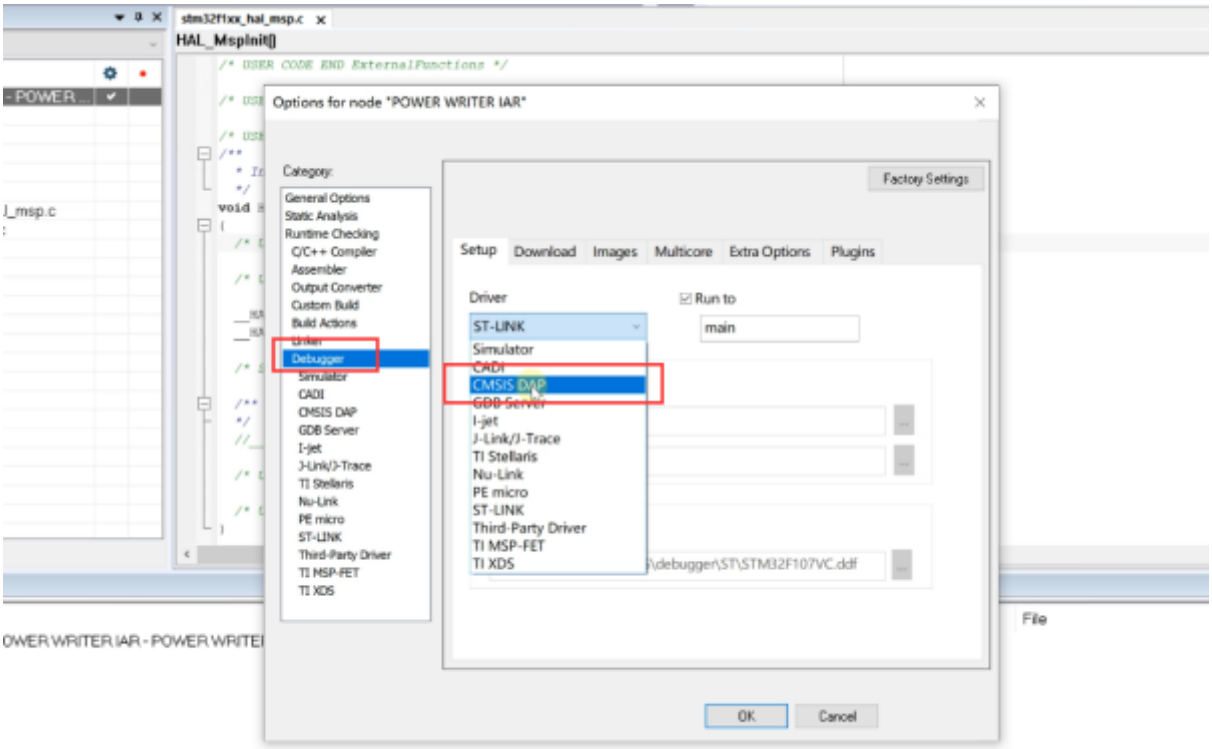
提示

参考视频：MDK中怎样使用Powerwriter的SWO（ITM Trace)

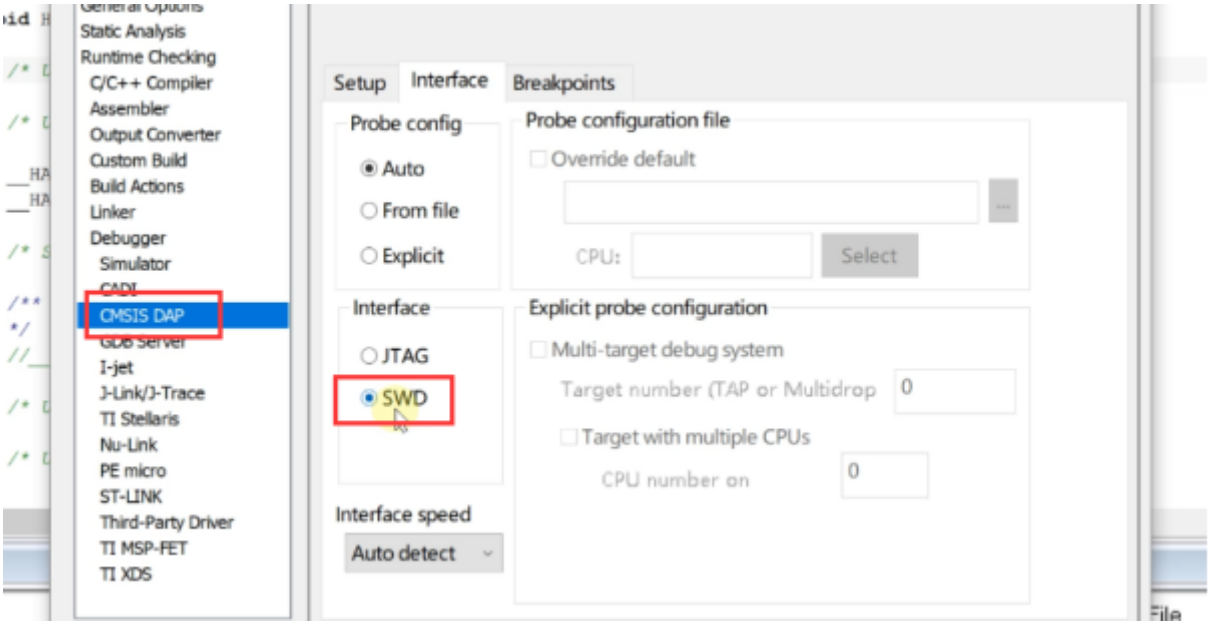
MDK中怎样使用ITM的其他高级功能

PowerWriter Debugger For IAR

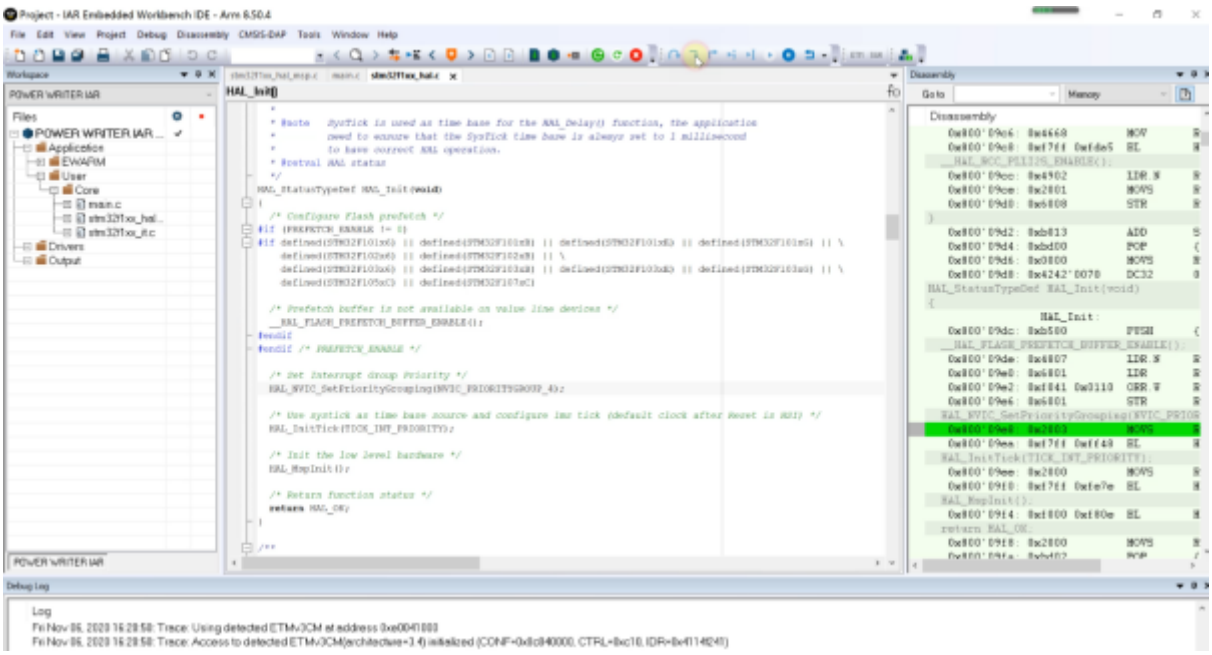
Power Writer 集成的Debugger 除了在MDK 中使用之外，也可以在IAR中使用，首先在Debugger 页面选择CMSIS-DAP 调试器，如下图所示：



然后在CMSIS DAP 中的Interface 接口选择 SWD 接口，如图所示：



设置完成后，执行启动调试操作后即可进入调试页面，如下图所示：



> **注意**
>
> - PowerWriter For IAR 调试时，没有Trace 功能，IAR中的Trace 功能使用的是ETM，如有必要请使用PowerWriter串口代替。
> - PowerWriter 中Debugger和 串口可以同时使用，两者为异步。
> - PowerWriter 提供的调试协议为SWJ协议，原因在于PowerWriter 需要兼顾Debugger,Writer,以及生产控制，串口功能。

PowerWriter Debugger For Other

Power Writer 的调试器兼容DAPLINK，其他DAPLINK 调试器方法，同样可以在PowerWriter 上使用，如配合OpenOCD + GDB 使用，由于此类配置方法比较灵活，请自行查阅相关资料来配置对应的调试环境，Openocd 官方站点：

Open On-Chip Debugger (openocd.org)

> **提示**
>
> 参考视频：MDK中怎样使用Powerwriter集成的Debugger
>
> 在IAR中使用Powerwriter的Debugger
>
> 如何在其他IDE中使用Powerwriter的Debugger

> **提示**

> **下载本页PDF文件**

编辑本页