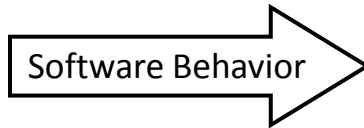
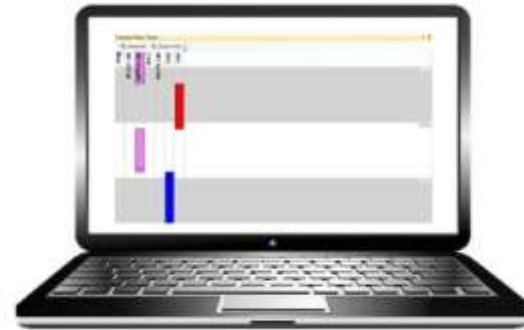


Tracing shows the real-time software behavior

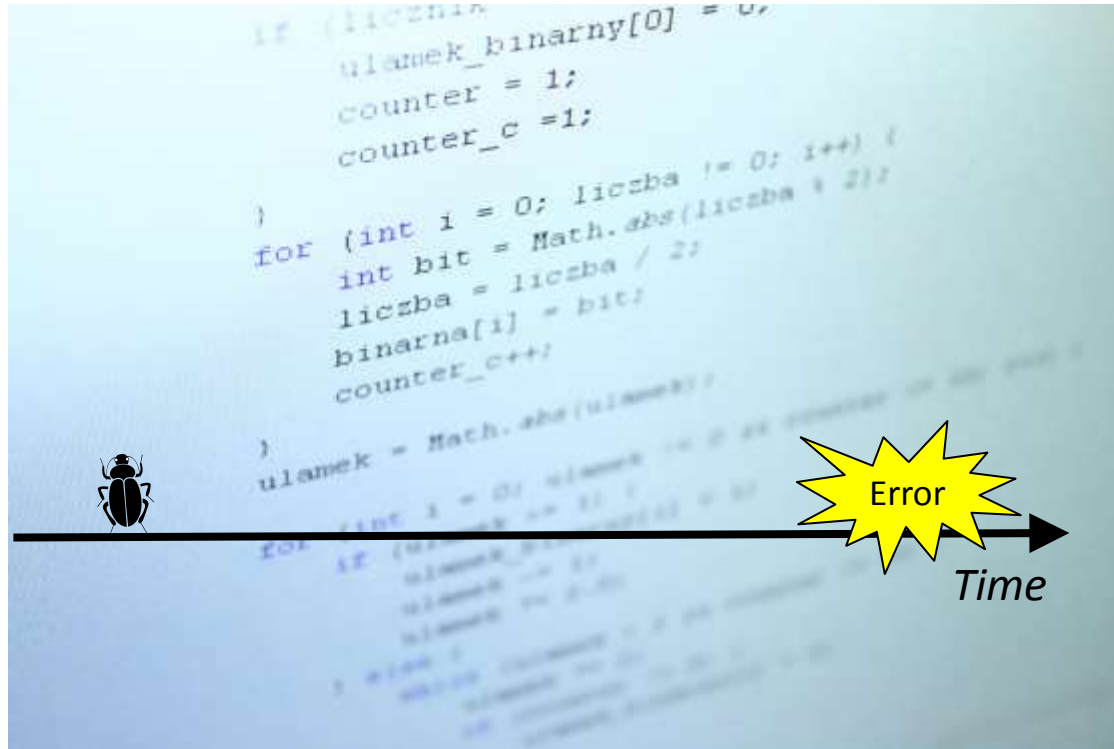
Record



Visualize

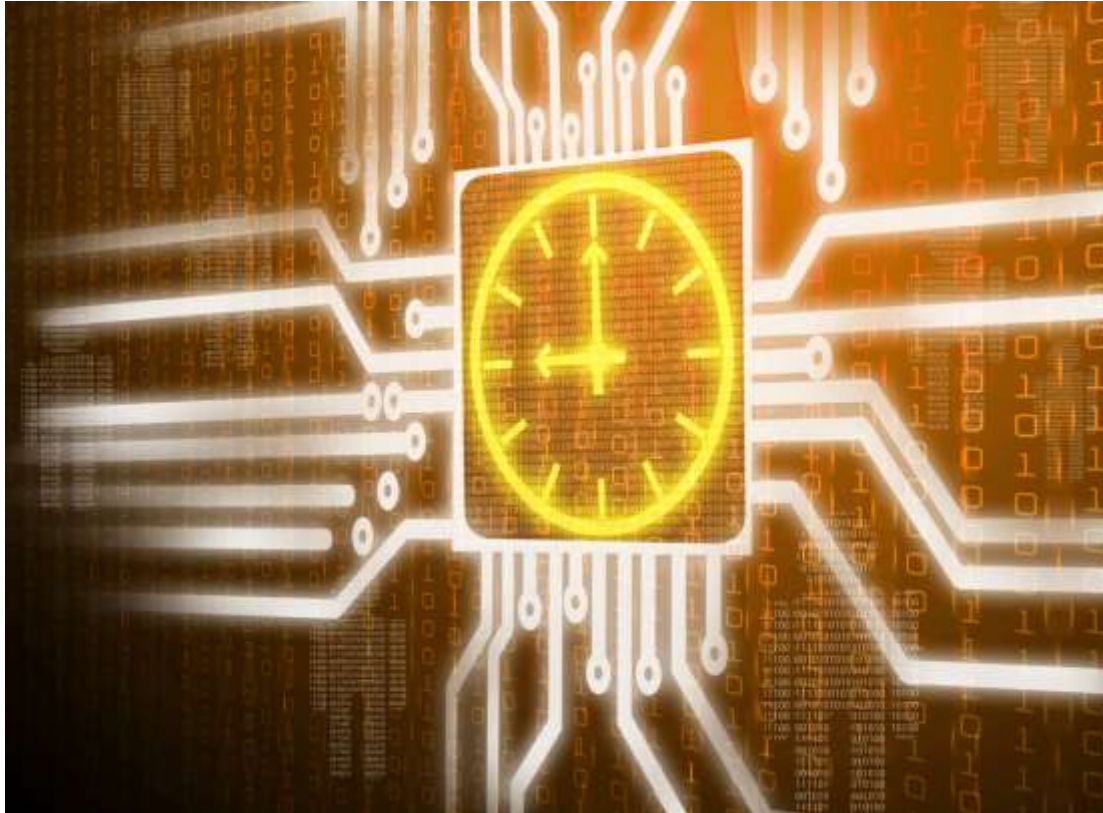


When to use Tracing?



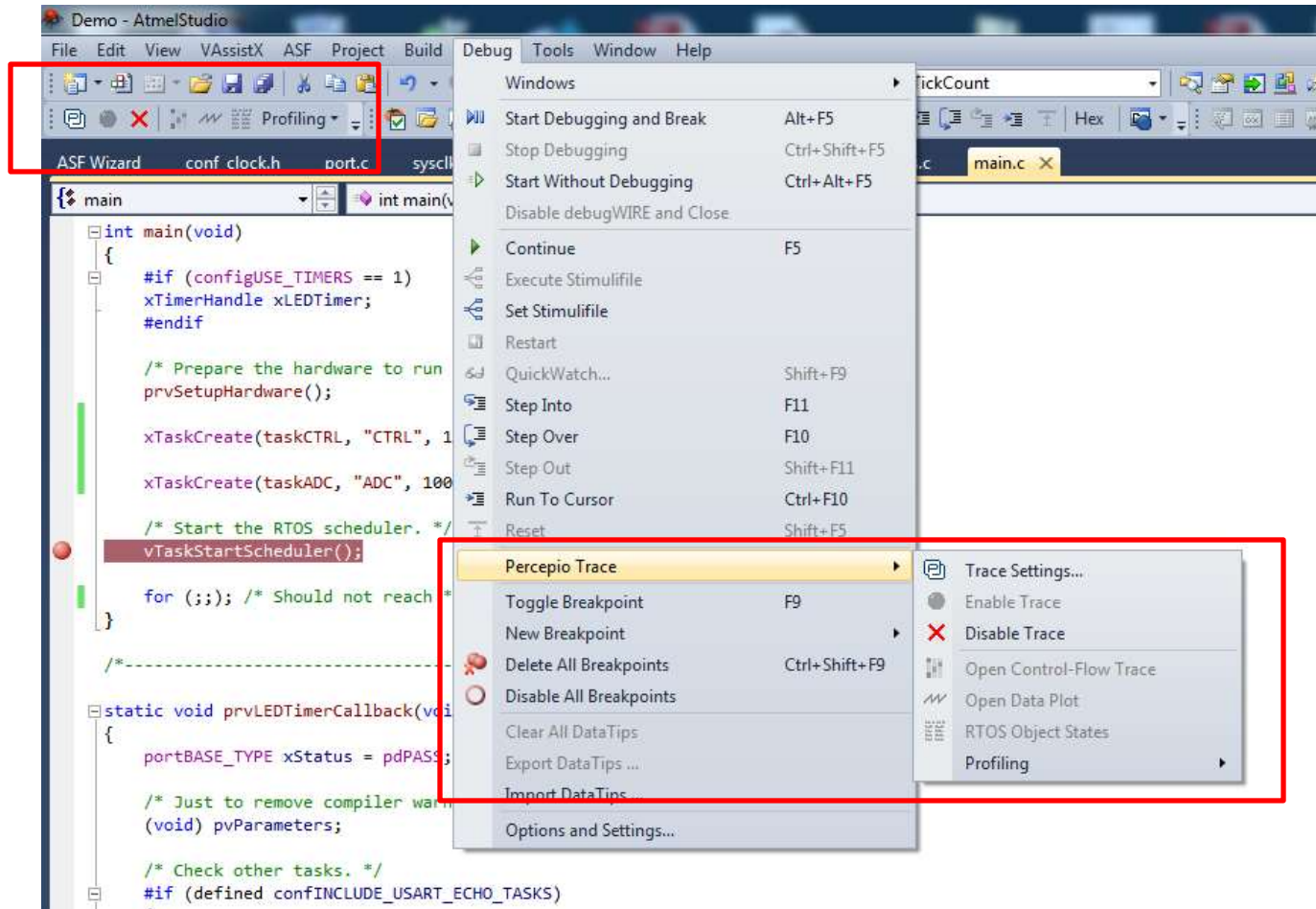
Complex problems – what events caused the error?

When to use Tracing?

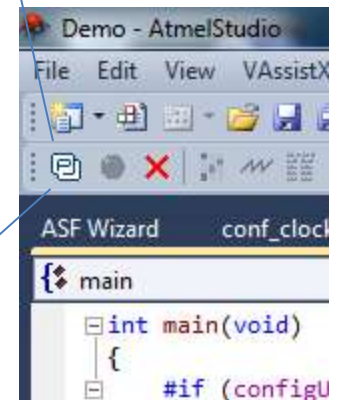
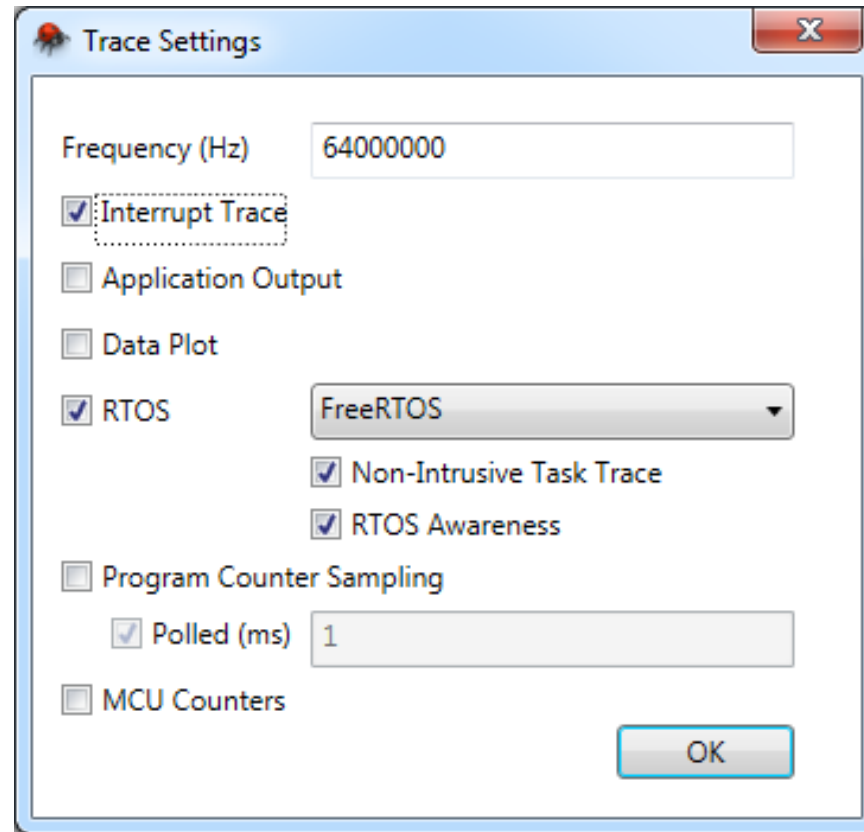


Timing issues and performance optimization

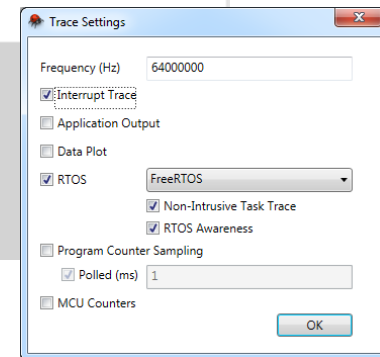
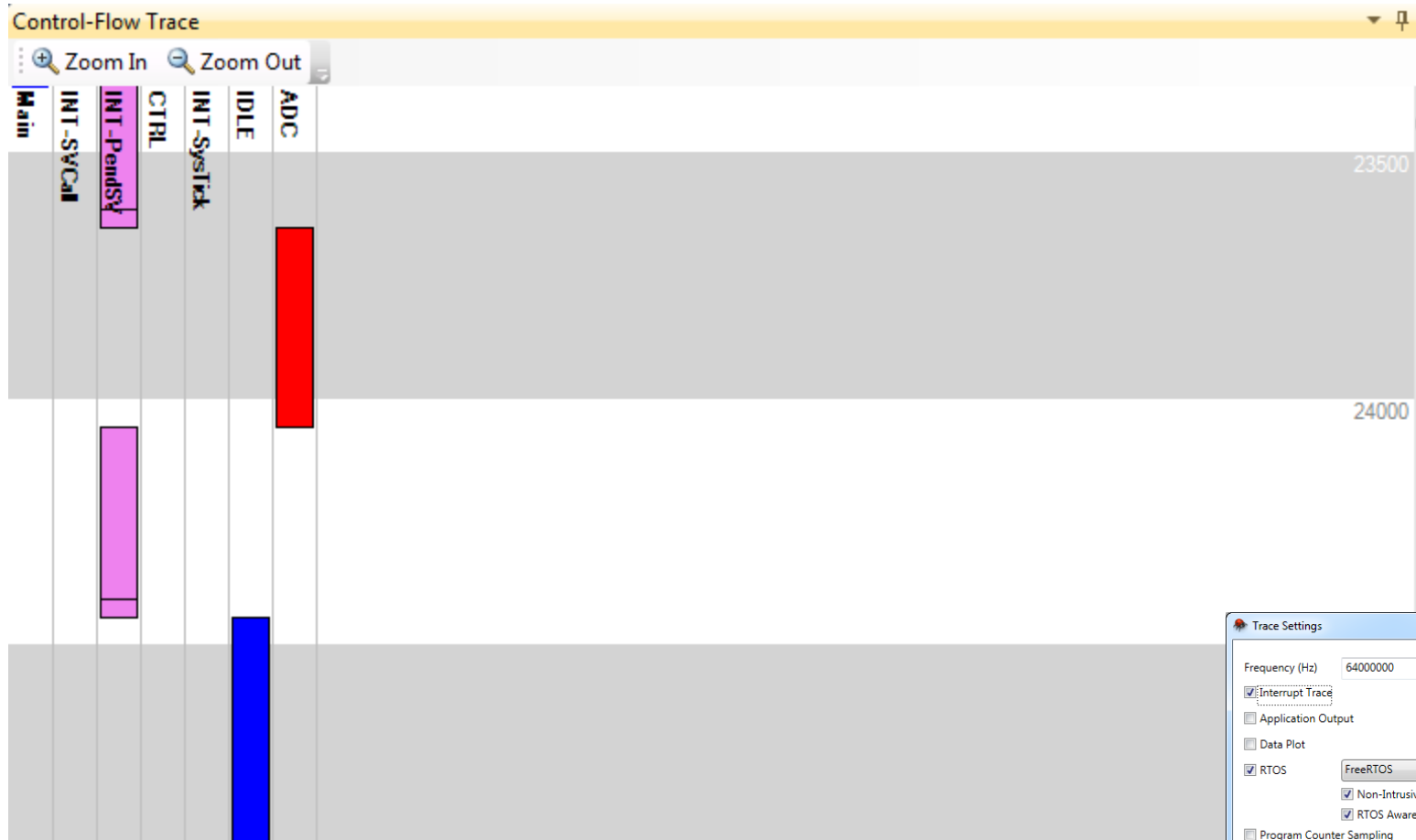
Perceprio Trace™



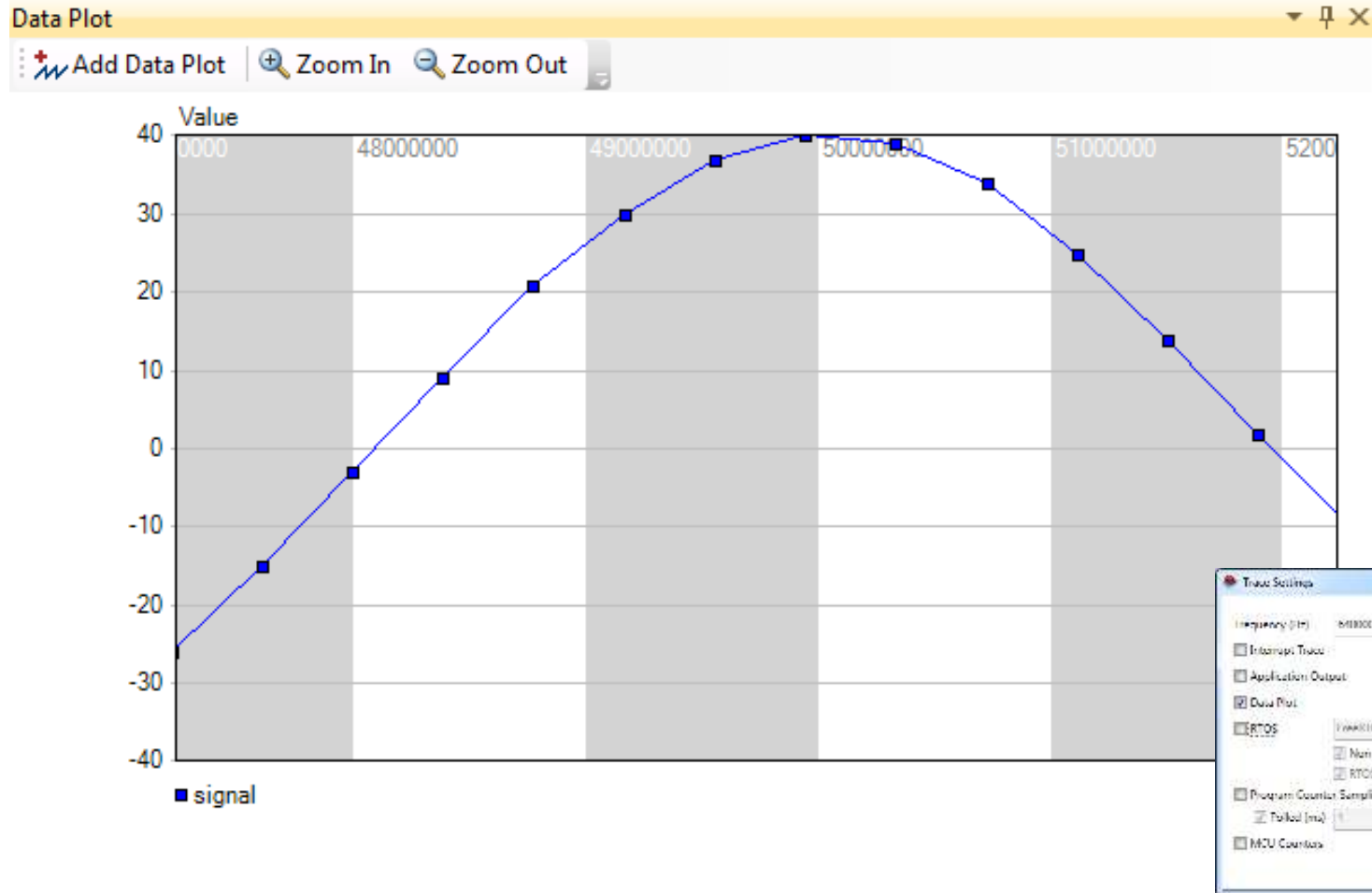
Percepio Trace Settings



Control-Flow Trace



Data Trace



Profiling

The image displays a development environment with the following components:

- Code Editor:** Shows the `SysTick_Handler` function. Key lines include:

```
void SysTick_Handler( void )  
{  
    /* If using preemption, also force a context switch. */  
    #if configUSE_PREEMPTION == 1  
        portNVIC_INT_CTRL_REG = portNVIC_INTENSET_INT11_Msk;  
    #endif  
  
    /* Only reset the SysTick load register if configUSE_TICKLESS_IDLE is set to 1. If it is set to 0 tickless idle is not being used. If it is set to a value other than 0 or 1 then a timer other than the SysTick is being used to generate the tick interrupt. */  
    #if configUSE_TICKLESS_IDLE == 1  
        portNVIC_SYSTICK_LOAD_REG = ulTimerReloadValueForOneTick;  
    #endif  
  
    ( void ) portSET_INTERRUPT_MASK_FROM_ISR();  
  
    vTaskIncrementTick();  
  
    portCLEAR_INTERRUPT_MASK_FROM_ISR( 0 );  
  
    #if configUSE_TICKLESS_IDLE == 1  
        __attribute__((weak)) void vPortSuppressTicksAndSleep( portTickType xExpectedIdleTime )  
        {  
            unsigned long ulReloadValue, ulCompleteTickPeriods, ulCompletedSysTickIncrements;  
  
            /* Make sure the SysTick reload value does not overflow the counter. */  
            if( xExpectedIdleTime > ulMaximumPossibleSuppressedTicks )  
            {  
                xExpectedIdleTime = ulMaximumPossibleSuppressedTicks;  
            }  
  
            /* Calculate the reload value required to wait xExpectedIdleTime tick periods. -1 is used because this code will execute part way through one of the tick periods, and the fraction of a tick period is accounted for later. */  
            ulReloadValue = ( ulTimerReloadValueForOneTick * ( xExpectedIdleTime - 1UL ) );  
            if( ulReloadValue > ulStoppedTimerCompensation )  
            {  
                ulReloadValue -= ulStoppedTimerCompensation;  
            }  
  
            /* Stop the SysTick momentarily. The time the SysTick is stopped for
```
- PC Sampling Timeline:** A horizontal bar chart showing the execution of various tasks over time. The tasks include `prvCheckTasksWaitingTermination`, `prvIdleTask`, `Calculate`, `memcmp`, `vTaskIncrementTick`, and `vTaskDelay`.
- PC Sampling Summary:** A table summarizing the sampling data:

Name	Sample...	Samples	Graphical
prvCheckTasksWaitingTermination	43.0	2026	
prvIdleTask	25.1	1183	
Calculate	19.5	898	
memcmp	0.8	110	
vTaskIncrementTick	0.6	122	
FeedSV_Handler	0.5	71	
SysTick_Handler	0.3	85	
vTaskSwitchContext	0.3	60	
get_run_time_counter_value	0.07	32	
vPortClearInterruptMask	0.03	15	
os_get_time	0.03	14	
xTaskGetTickCountFromISR	0.03	14	
sysclk_get_main_hz	0.03	13	
sysclk_get_cpu_hz	0.03	12	
uPortSetInterruptMask	0.02	11	
uPortCountLeadingZeros	0.02	9	
xTaskResumeAll	0.01	6	
vTaskDelay	0.01	4	
uPortReset	0.01	4	
- Trace Settings Dialog:** A configuration window for the trace tool. It shows a frequency of 6400000 Hz and several checked options: `Interrupt Trace`, `Application Output`, `Data Plot`, `Non-Intrusive Task Traces`, `RTOS Awareness`, `Program Counter Sampling`, and `MCU Counters`.

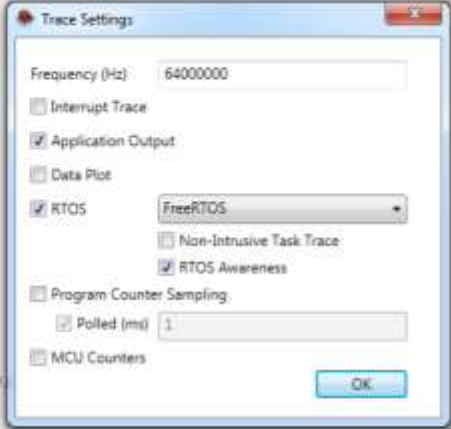
Debug prints (SWO)

```
while(1)
{
    switch (AppData.task_state)
    {
        case 0:
            SWO_print_string(0, "State 0");
            AppData.task_state = 1;
            ReadInput(AppData.input_buffer, 20);
            break;

        case 1:
            SWO_print_string(0, "State 1");
            AppData.task_state = 2;
            Calculate(AppData.input_buffer, AppData.output_buffer);
            break;

        case 2:
            SWO_print_string(0, "State 2");
            AppData.task_state = 0;
            WriteOutput(AppData.output_buffer);
            break;

        default:
            SWO_print_string(0, "Error - Unknown state value in taskCTRL.");
            for(;;); // Error, unexpected value of task_state.
    }
}
```



The Trace Settings dialog box is open, showing the following configuration:

- Frequency (Hz): 64000000
- Interrupt Trace
- Application Output
- Data Plot
- RTOS (FreeRTOS)
- Non-Intrusive Task Trace
- RTOS Awareness
- Program Counter Sampling
- Polled (ms): 1
- MCU Counters

OK

Output

Show output from: PercepioTrace

```
[119578116 - 119579719] (0) "State 2"  
[122706213 - 122707832] (0) "State 0"  
[125834322 - 125835924] (0) "State 1"  
[129801551 - 129803163] (0) "State 2"  
[132929654 - 132931255] (0) "State 0"  
[136857763 - 136859368] (0) "State 1"  
[140024993 - 140026607] (0) "State 2"  
[143153095 - 143154699] (0) "State 0"  
[146261205 - 146262812] (0) "State 1"  
[150248433 - 150250030] (0) "State 2"  
[153376536 - 153378143] (0) "State 0"  
[156588814 - 156513970] (0) "Error! Unknown value for AppData.task_state"
```

SWO_print_string (SAM3/4)

```
#define ITM_TER *(volatile int*)0xE0000E00 // Trace Enable Register
#define ITM_TCR *(volatile int*)0xE0000E80 // Trace Control Register
#define ITM_STIM_8(n) *(volatile uint8_t*)(0xE0000000 + ((n) * 4)) // 8 Bit N < 32

void SWO_print_char(int port, char c);
void SWO_print_string(int port, const char *s);

void SWO_print_char(int port, char c)
{
    while ((ITM_STIM_8(port) & 1) == 0);
    ITM_STIM_8(port) = c;
}

void SWO_print_string(int port, const char *s)
{
    if ((ITM_TCR & 1) == 0) return;
    if ((ITM_TER & 1) == 0) return;
    while (*s) SWO_print_char(port, *s++);
    SWO_print_char(port, 0);
}
```

Demo

- SAM3S-EK2 board + SAM-ICE
- Basic FreeRTOS project with three issues.
 - Data corruption
 - Task scheduling issue
 - Performance bottleneck

