



## What is the difference between 'Trace Enable' and 'TraceOn/TraceOff'?

2025-02-25 - Comments (0) - Tracing

In short: `TraceEnable` defines what trace data is generated, while `TraceON/TraceOFF` controls when the generation of trace data starts and stops.

If you only need a quick answer, you can stop reading here. But if you want a more detailed explanation, let's dive deeper.

### Understanding Trace Filtering

In many cases, generating trace data for the entire program and data flow is unnecessary. Instead, you may want to focus on specific instructions, data accesses, or a short program section. This is where trace filters—`TraceEnable`, `TraceON`, and `TraceOFF`—come into play. These filters configure the available comparators in the core trace logic so that trace data is only generated for the events of interest. The number of filters you can use and the level of detail in the trace data depend on the number of comparators and the trace protocol used.

### How TraceEnable Works

The command `Break.Set <program_event> /TraceEnable` configures the comparators to generate trace data only for the specified event. In this case, general settings for program and data flow visibility are ignored.

#### **Examples:**

Generates trace data when a write access to the variable 'flags[3]' occurs:

```
Var.Break.Set flags[3] /Write /TraceEnable
```

Generates trace data when the 'sieve' function is entered:

```
Break.Set sieve /Program /TraceEnable
```

When displaying trace data, TRACE32 always uses the `TRACE ENABLE` keyword when the specified event occurred:

record	run	address	cycle	data	symbol	ti.back
+0000058097		D:08002A0C	wr-long	00000068	..v7r_le\Global\_bss_end+0x110	824.435us
+0000058099		D:08002A10	wr-long	08002A50	..v7r_le\Global\_bss_end+0x114	<0.005us
+0000058110		TRACE ENABLE				
		R:08001B48	ptrace		..eve_ram_arm_v7r_le\sieve\sieve	0.500us
<pre> #define SIZE 18 char flags[SIZE+1];  static int sieve(void)                /* sieve of erathostenes */ {     push    {r4-r6,r11}     D:08002A04 wr-long 08002508 ..v7r_le\Global\_bss_end+0x108 &lt;0.005us     D:08002A08 wr-long 00000000 ..v7r_le\Global\_bss_end+0x10C 1.000us     D:08002A0C wr-long 00000068 ..v7r_le\Global\_bss_end+0x110 &lt;0.005us     D:08002A10 wr-long 08002A50 ..v7r_le\Global\_bss_end+0x114 &lt;0.005us </pre>						
+0000058137		TRACE ENABLE				
		R:08001B48	ptrace		..eve_ram_arm_v7r_le\sieve\sieve	824.935us
<pre> #define SIZE 18 char flags[SIZE+1];  static int sieve(void)                /* sieve of erathostenes */ {     push    {r4-r6,r11}     D:08002A04 wr-long 08002508 ..v7r_le\Global\_bss_end+0x108 &lt;0.005us     D:08002A08 wr-long 00000000 ..v7r_le\Global\_bss_end+0x10C 0.500us     D:08002A0C wr-long 00000068 ..v7r_le\Global\_bss_end+0x110 &lt;0.005us     D:08002A10 wr-long 08002A50 ..v7r_le\Global\_bss_end+0x114 &lt;0.005us </pre>						

## How TraceON and TraceOFF Work

The command `Break.Set <start_event> /TraceON` configures the comparators to start generating trace data when <start\_event> occurs. Similarly, the command `Break.Set <stop_event> /TraceOFF` stops trace data generation when <stop\_event> occurs.

### Example:

Start generating trace data when line 16 of the 'func2' function is executed and stop generating it when the 'sieve' function exits:

`Break.Set func2\16 /Program /TraceON`

`Break.Set symbol.EXIT(sieve) /Program /TraceOFF`

The TRACE ENABLE keyword marks where trace data generation stopped before restarting:

record	run	address	cycle	data	symbol	ti.back
+0000000843		cmp r4,#0x12 ; i,#18				
		b!e 0x8001B8C				
		R:08001BE4	ptrace		..am_arm_v7r_le\sieve\sieve+0x9C	<0.005us
		}				
757		return count;				
758		ldr r3,[r11,#-0x10]				
		}				
		cpy r0,r3				
		sub r13,r11,#0x0C ; r13,r11,#12				
		pop {r4-r6,r11}				
+0000000851		TRACE ENABLE				
		R:080002C8	ptrace		..am_arm_v7r_le\sieve\func2+0x78	9.000us
170		fstatic += mstatic1;				
		movw r3,#0x1F60				
		movt r3,#0x800				
		ldr r2,[r3]				
		movw r3,#0x1F90				
		movt r3,#0x800				
		ldr r3,[r3]				
		add r2,r2,r3				
		movw r3,#0x1F60				
		movt r3,#0x800				

Note

If TraceON <start\_event> occurs after TraceOFF <stop\_event> has taken effect, the generation of trace data is restarted.

### **Controlling What Type of Trace Data is Generated**

Now that we've discussed when tracing starts and stops, let's look at what kind of trace data is generated. For most core architectures, the default trace generates program flow trace data. However, additional details (e.g., data accesses) can be explicitly enabled. For the Cortex-R5 for example, in addition to program flow, you can also generate trace data for write accesses.

```
ETM.DataTrace Write
```

```
Break.Set func2\16 /Program /TraceON
```

```
Break.Set sYmbol.EXIT(sieve) /Program /TraceOFF
```

### **Combining TraceEnable with TraceON/TraceOFF**

Finally, let's look at a case where TraceEnable is combined with TraceON and TraceOFF—again for the Cortex-R5.

```
ETM.DataTracePrestore ON
```

```
Break.Set func2\16 /Program /TraceON
```

```
Break.Set sYmbol.EXIT(sieve) /Program /TraceOFF
```

```
Var.Break.Set flags /Write /TraceEnable
```

This configuration ensures that trace data for write accesses (i.e., what) to the variable 'flags' is generated, but only between the program points (i.e., when) at line 16 of 'func2' and the exit of 'sieve'.