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

2026-04-03 - Comments (0) - Tracing
In short:

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

Further Details:

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:

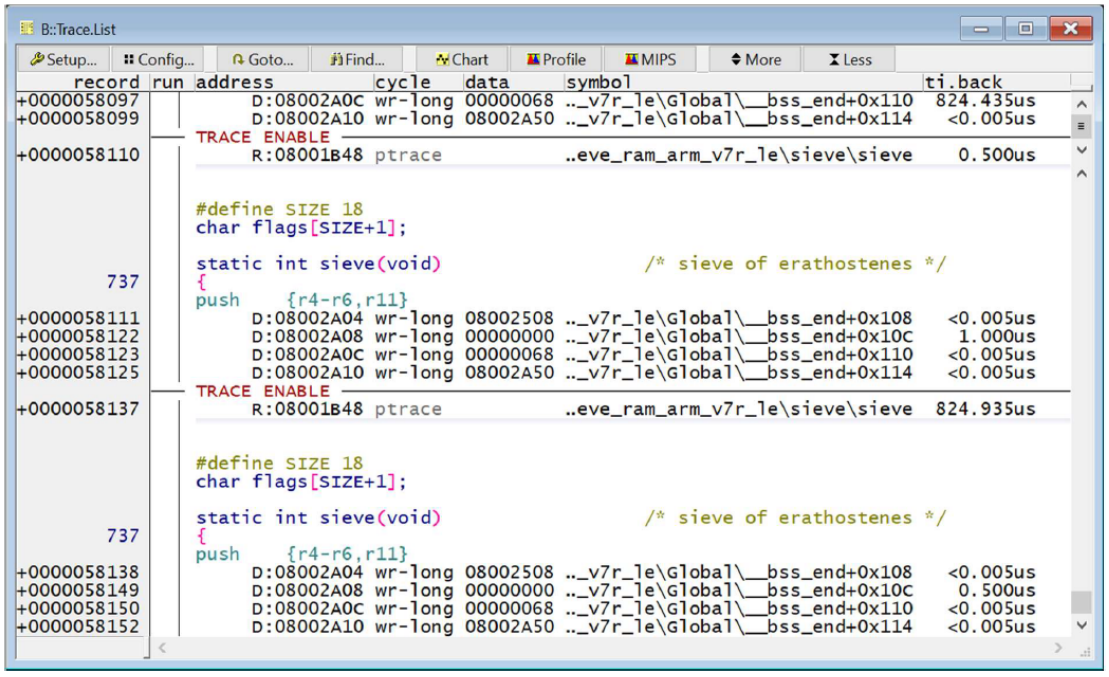
Generates trace data when a write access to the variable `flags[3]` occurs:

```
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:



How TraceON and TraceOFF Work

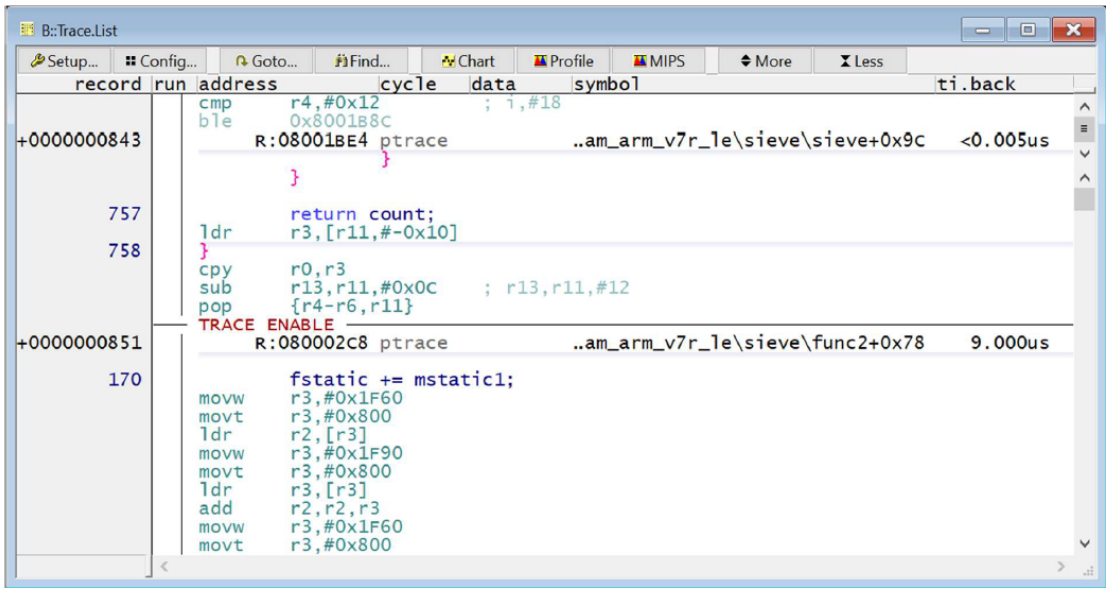
The command `Break.Set <start_event> /TraceON` configures the comparators to start generating trace data when `<start_event>` occurs. Similarly, the command `Break.Set <stop_event> /TraceOFF` stops trace data generation when `<stop_event>` occurs.

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:



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()`