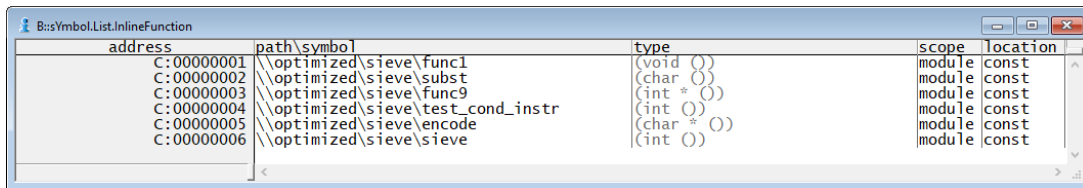


Can I debug and trace inline functions?

2023-10-25 - Comments (0) - TRACE32 PowerView

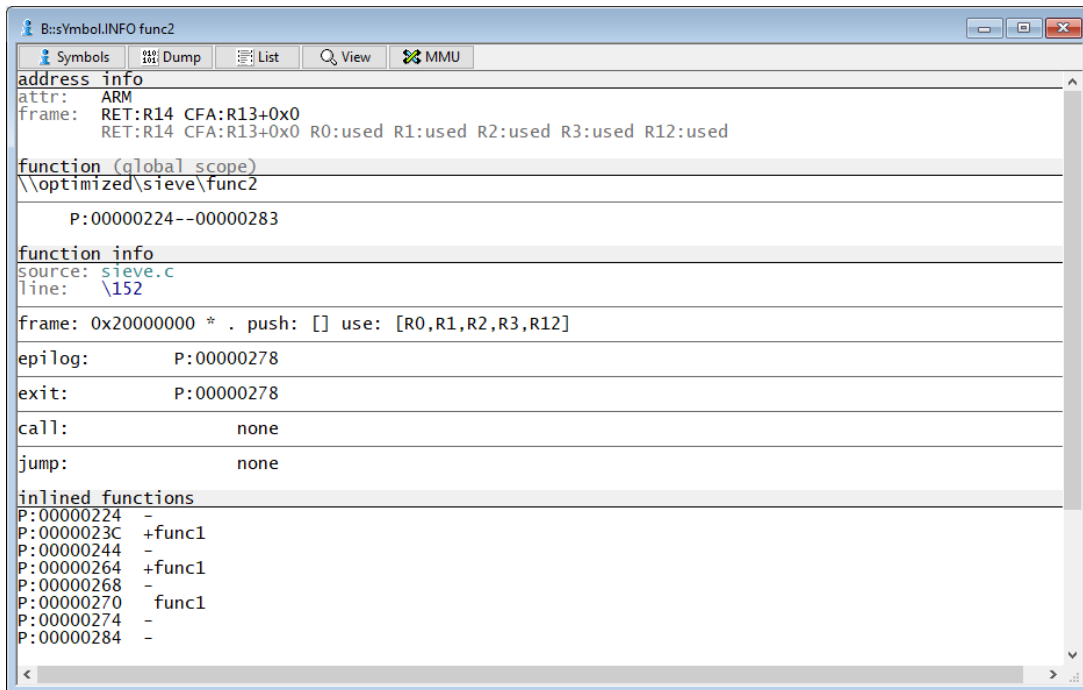
Inline functions are embedded inside the calling functions at each call site. They do not have a unique start address. Inline functions can be displayed in TRACE32 using the command **sYmbol.List.InlineFunction**



address	path\symbol	type	scope	location
C:00000001	\\optimized\sieve\func1	(void ())	module	const
C:00000002	\\optimized\sieve\subst	(char ())	module	const
C:00000003	\\optimized\sieve\func9	(int * ())	module	const
C:00000004	\\optimized\sieve\test_cond_instr	(int ())	module	const
C:00000005	\\optimized\sieve\encode	(char * ())	module	const
C:00000006	\\optimized\sieve\sieve	(int ())	module	const

The debugger can display some information about inline functions if the compiler provides information about inlining in the debug information.

The call sites of inline functions can be displayed in TRACE32 using the **sYmbol.INFO** command under "inlined functions". The following screenshot shows for instance that the function func1 has been inlined twice inside the function func2:



```

B::sYmbol.INFO func2
Symbols  Dump  List  View  MMU
address info
attr: ARM
frame: RET:R14 CFA:R13+0x0
      RET:R14 CFA:R13+0x0 R0:used R1:used R2:used R3:used R12:used

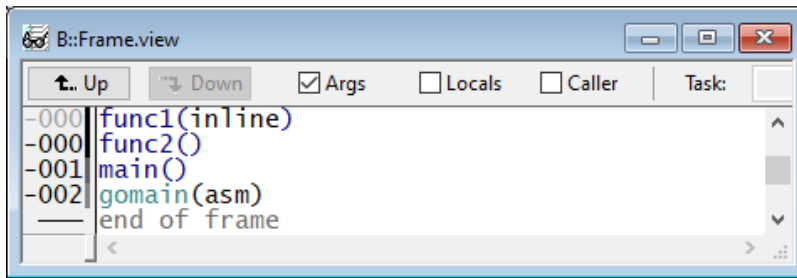
function (global scope)
\\optimized\sieve\func2
P:00000224--00000283

function info
source: sieve.c
line: \152

frame: 0x20000000 * . push: [] use: [R0,R1,R2,R3,R12]
epilog: P:00000278
exit: P:00000278
call: none
jump: none

inlined functions
P:00000224 -
P:0000023C +func1
P:00000244 -
P:00000264 +func1
P:00000268 -
P:00000270 func1
P:00000274 -
P:00000284 -
  
```

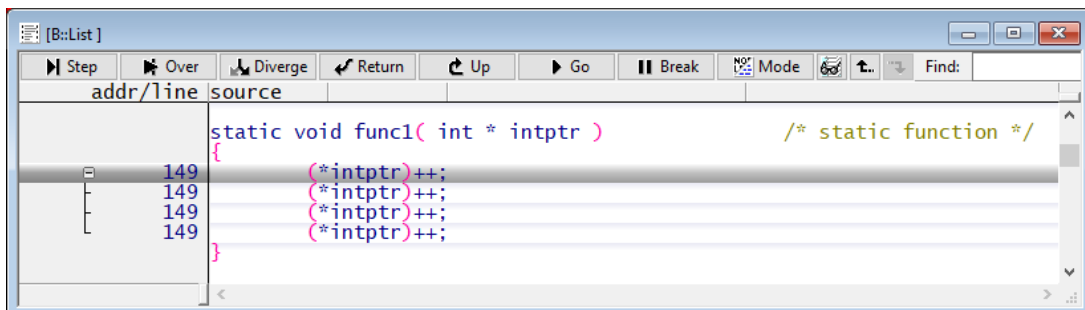
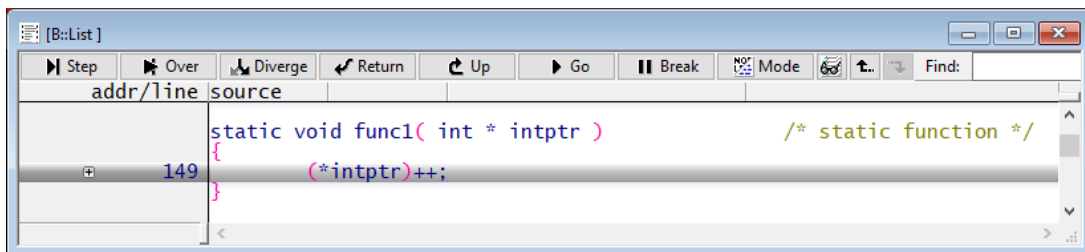
This information is used by the debugger in the function stack frame (**Frame.view**)



As the function does not have a unique start address, it is however not possible to set a breakpoint at the function start. TRACE32 will show the error message "symbol not found"



You can however set program breakpoints in single lines of inline function which have been translated into assembly code. If the function is inlined more than once, you will probably see a [+] sign which can be expanded to show the different calling points



TRACE32 additionally allows displaying runtime information about inline functions based on the trace results using the /INLINE option of the commands **Trace.STATistic.sYmbol**, **Trace.Chart.sYmbol** and **Trace.PROfileChart.sYmbol**. Refer for more information about these commands to https://www.lauterbach.com/pdf/general_ref_t.pdf

address	total	min	max	avr	count (first)	ratio%	1%	2%	5%	10%	20%
(other)	0.000us	0.000us	-	0.500us	0.	0.000%					
\\optimized\sieve\func1	11.202ms	0.000us	1.000us	0.500us	22405. (1/1)	0.643%	←				
\\optimized\sieve\func2	20.910ms	0.300us	1.100us	2.800us	7469. (1/0)	1.201%	←				
\\optimized\sieve\main	179.979ms	0.100us	9.200us	-	1. (1/0)	10.337%	←				
\\optimized\sieve\func2a	14.936ms	2.000us	2.000us	2.000us	7468.	0.857%	←				
\\optimized\sieve\func2b	14.936ms	2.000us	2.000us	2.000us	7468.	0.857%	←				

Refer for more information about debugging optimized code to the following video: <https://support.lauterbach.com/kb/articles/debugging-optimized-code-in-trace32>

