

Execution Time for Report: timing.rvd Report has been upgraded, but not saved to disk

nanoseconds (ns) Find element

Root Summary

Function	Minimum	Average	High-Water	Maximum	
Name	Min-OverET (ns)	A-OverET (ns)	H-OverET (ns)	Max-OverET (ns)	#Tests
my_main	353,900	353,900	353,900	353,900	1

Function Summary

Overall Execution Time

Function	Minimum	Average	High-Water	Maximum	Difference
Name	Min-OverET (ns)	A-OverET (ns)	H-OverET (ns)	Max-OverET (ns)	Max - Av
self_test	338,400	338,400	338,400	338,400	
add_one	3,900	52,734	83,400	83,400	
add_pos			83,400	83,400	
add_neg			93,800	93,800	
negate				200	

test.dtx:58:1-139:100

Block Tests HWM Path Find text

```

211 82e0: e51b3014 ldr r3, [fp, #-2
212 82f0: e1823003 orr r3, r2, r3
213 82f4: e50b3010 str r3, [fp, #-2
214 82f8: e51b2018 ldr r2, [fp, #-2
215 82fc: e51b3010 ldr r3, [fp, #-2
216 8300: e1520003 cmp r2, r3
217 8304: 8affffd5 bhi 8260 <subtract_one>
218 8308: e51b3008 ldr r3, [fp, #-2
219 830c: e1a00000 nop ; (n
220 8310: e1a00003 mov r0, r3
221 8314: e24bd004 sub sp, fp, #4
222 8318: e8bd4800 pop {fp, lr}
223 831c: e12fff1e bx lr
224
225
226 00008320 <negate>:
227 8320: e52db004 push {fp}
228 8324: e28db000 add fp, sp, #0
229 8328: e24dd00c sub sp, sp, #12
230 832c: e50b0008 str r0, [fp, #-8
231 8330: e51b3008 ldr r3, [fp, #-8
232 8334: e2633000 rsb r3, r3, #0
233 8338: e1a00000 nop ; (n
234 833c: e1a00003 mov r0, r3
235 8340: e28bd000 add sp, fp, #0
236 8344: e8bd0800 ldmfd sp!, {fp, lr}
237 8348: e12fff1e bx lr
238
239 0000834c <absolute>:
240 834c: e92d4800 push {fp, lr}
241 8350: e28db004 add fp, sp, #4
242 8354: e24dd008 sub sp, sp, #8
243 8358: e50b0008 str r0, [fp, #-8
244 835c: e51b3008 ldr r3, [fp, #-8
245 8360: e3530000 cmp r3, #0
246 8364: aa000003 bge 8378 <absolute>
247 8368: e51b0008 ldr r0, [fp, #-8
248 836c: ebf00003 bl 8320 <negate>
249 8370: e1a03000 mov r3, r0
250 8374: ea000000 b 837c <absolute>
251 8378: e51b3008 ldr r3, [fp, #-8
252 837c: e1a00000 nop ; (n
253 8380: e1a00003 mov r0, r3
254 8384: e24bd004 sub sp, fp, #4
255 8388: e8bd4800 pop {fp, lr}
256 838c: e12fff1e bx lr

```



Safety through quality

PRODUCT BRIEF

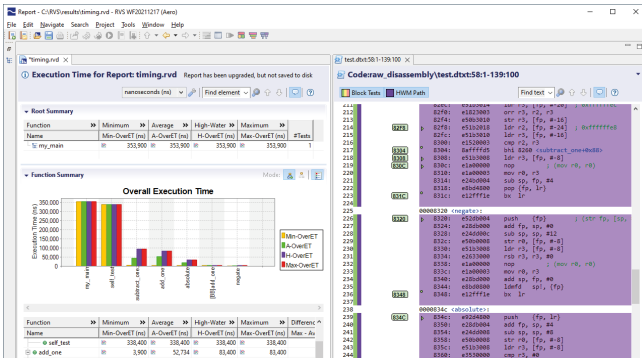
Zero-footprint execution time analysis with RapiTime<sup>Zero</sup>

# Product brief: RapiTime<sup>Zero</sup>

## RapiTime<sup>Zero</sup>

### How can RapiTime<sup>Zero</sup> help you?

RapiTime<sup>Zero</sup> lets you observe the execution time behavior (including worst-case execution time) of object code from critical software without needing to make any modifications to, or even have access to, your project's source code.



Timing results collected from RapiTime<sup>Zero</sup> alongside assembly code

### Benefits

Verify the execution time behavior of critical software without needing:

- Any instrumentation.
- Project source code.
- Any modification to your development environment.

### RapiTime<sup>Zero</sup> use cases

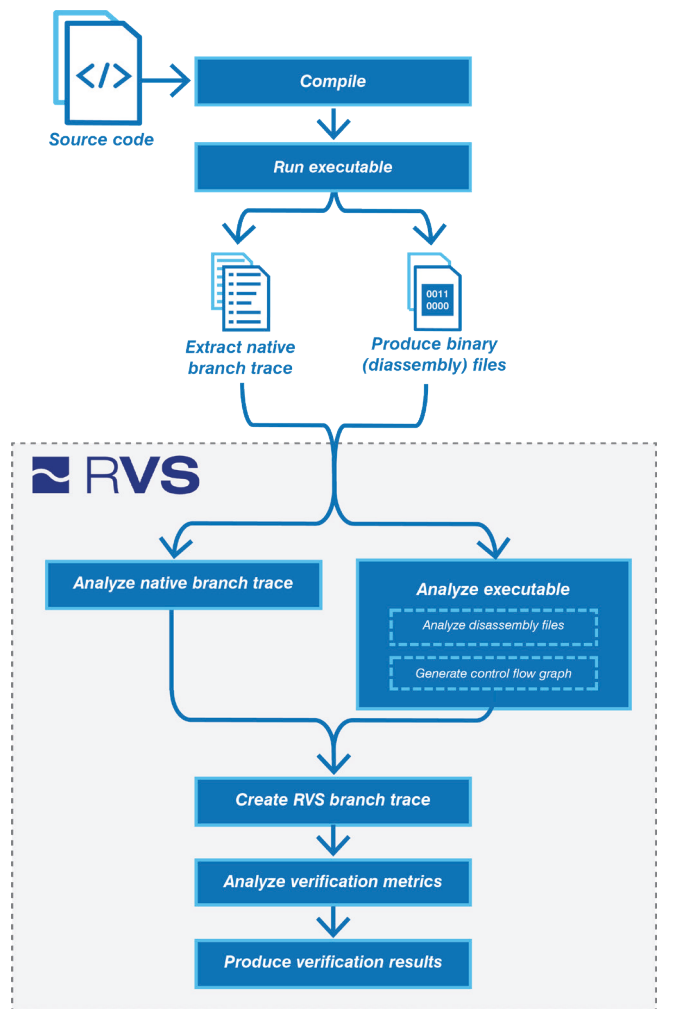
- Demonstrate that software executes within its time constraints.
- Conduct WCET/high water mark analysis.
- Execution time analysis for third-party libraries.
- Execution time analysis with no impact on the code base or development environment.
- Address avionics software guidelines: DO-178B/C, A(M)C 20-193, MIL-HDBK-512C, AA-22-01, AMACC ...
- Address space software standards: NASA NPR 7150.2d and ECSS-E-ST-40C.
- Address ISO 26262 requirements.

### How does RapiTime<sup>Zero</sup> work?

RapiTime<sup>Zero</sup> reconstructs information on software execution behavior by matching branch trace information collected from the hardware (which must support this) with a control flow graph produced from a disassembly of the software binary.

Having matched this data, a reconstructed branch trace is created, which can be used to analyze the execution time behavior of the executable code while it ran.

**The branch trace is a crucial component of the analysis process and this must be available in the existing development environment through the CPU and/or external hardware being used.**



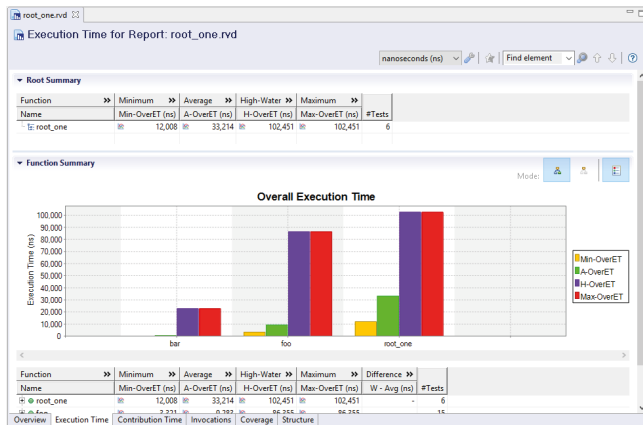
RapiTime<sup>Zero</sup> verification process

## Key features

### Execution time analysis

Calculation of detailed timing metrics for each function and sub-function:

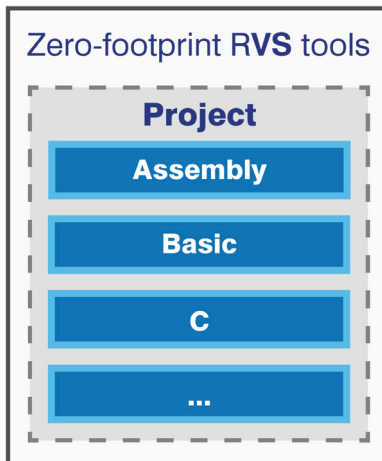
- Minimum, maximum and average execution time
- Execution time density
- Contribution to HWM path
- Merge results from different test runs, builds and strategies



Execution time results from RapiTimeZero displayed in the RVS user interface

### Language support

- Any language that targets machine code
- Mixed source languages



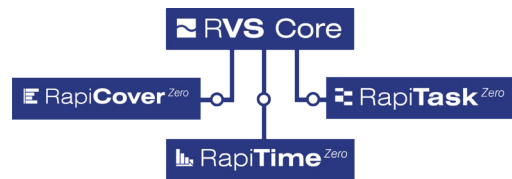
Zero footprint RVS tools can be used to analyze software in any language that targets machine code

### Supported platforms

- It must be possible to produce and collect branch trace information from the platform during program execution
- It must be possible to observe context switch information from executables on the platform
- Platform Support Package required to interface between RapiTimeZero and platform (see Platform Support Packages)
- To assess whether a Platform Support Package is available for your platform, see the [compatibility tab on our RapiTimeZero product page](#)
- We can develop additional Platform Support Packages to support RapiTimeZero analysis for compatible platforms

### Integration support

- Automatable testing environment
- Support for very large code bases
- No library/run-time dependencies or dynamic memory requirements
- Shared integration with zero-footprint RVS tools
- Continuous build servers e.g. Jenkins®, Atlassian Bamboo®
- Multicore support (depending on hardware support)



Shared integration with zero-footprint RVS tools

### Integrated testing environment

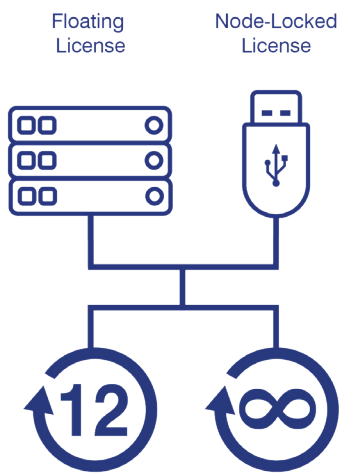
- Summary and detailed results views
- Invocation timeline, aggregate profile and treemap charts to help understand timing behavior at a glance
- Project and code base insights including code complexity, treemaps, call dependencies, and LOC
- Trace rewind feature to debug timing behavior
- Filter results by subprogram
- Code viewer:
  - View object code alongside source code, where available
  - Color-coded by high water mark path
  - View the mapping between object code and source code
- Aggregate timing metrics by directory, file and functions
- Multiple export formats e.g. text, XML, CSV
- Compare reports
- Database-like search function
- Multi-user testing environment

## Compatibility

- Runs on host operating systems
  - Windows® 10+ and Windows Server® 2016 R2+
  - Linux® distributions including Ubuntu® and Red Hat®
- Results can be collected from systems without supported operating systems and transferred to a supported system for analysis

## Licensing

- Enterprise license gives you access to new versions, support and maintenance
- One-year support and maintenance included in purchase price
- Single price for all features
- Licenses transferable across projects



RapiTime<sup>Zero</sup> has flexible licensing options, letting you get the most from the software depending on your needs

## Should I use RapiTime or RapiTime<sup>Zero</sup>?

RapiTime<sup>Zero</sup> offers many benefits, but in some cases RapiTime may be more appropriate for you. Consult Table 1 below to decide if RapiTime<sup>Zero</sup> or RapiTime is best for you. For more information, contact us at [info@rapitasystems.com](mailto:info@rapitasystems.com).

**Table 1.** Comparison of key RapiTime and RapiTime<sup>Zero</sup> features

Feature	RapiTime	RapiTime <sup>Zero</sup>
<i>Works without source code</i>	No	Yes
<i>Works without Instrumentation</i>	No	Yes
<i>Integration with development environment</i>	Integration needed	No integration needed
<i>Worst-case execution time analysis</i>	Yes	No
<i>Tool qualification support</i>	Yes	Not yet available
<i>Trace size and data processing time</i>	Depends on applied instrumentation	Typically larger trace and longer data processing times
<i>Supported platforms (target, data collection mechanism)</i>	Flexible, almost any platform supported	Requirements on platform (branch trace and context switch information must be available), PSP needed

## Platform Support Packages

To enable Rapi**Time**<sup>Zero</sup> analysis on a specific platform, Platform Support Packages (PSPs) are needed for Rapi**Time**<sup>Zero</sup> to interface with that platform in order to do the following:

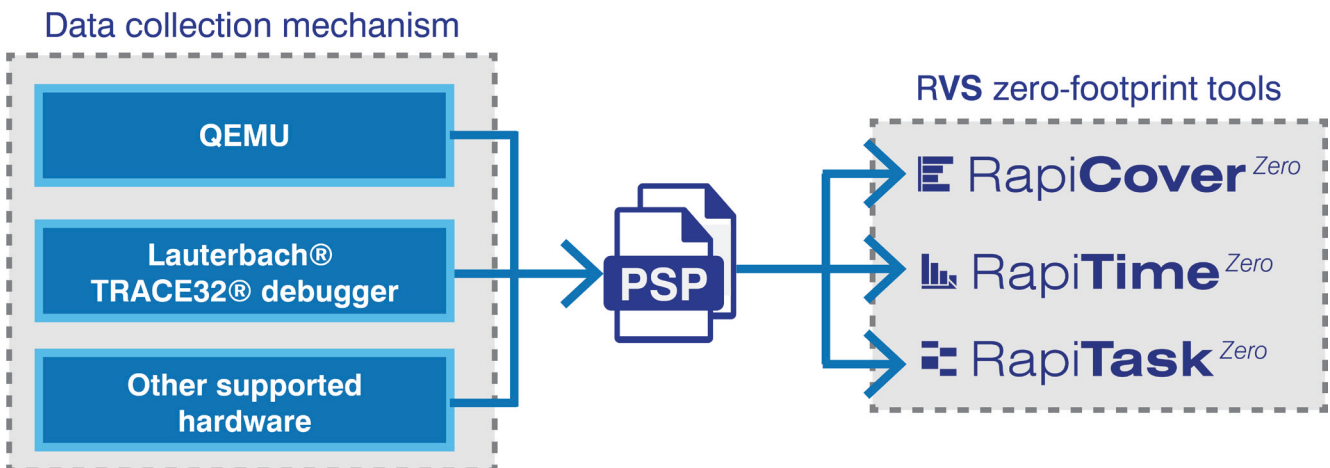
- Convert the specific format of native branch traces generated by the platform into a format that Rapi**Time**<sup>Zero</sup> understands and can use for subsequent analysis.
- Disassemble the object code to understand the structure and control flow of the code so this can be used for subsequent Rapi**Time**<sup>Zero</sup> analysis.

Each PSP is designed to support various components of a platform. These include:

- The compiler(s) used to generate executables to be analyzed by Rapi**Time**<sup>Zero</sup>
- The instruction set of object code to be analyzed by Rapi**Time**<sup>Zero</sup>
- The native branch trace format generated from the platform – this depends on the mechanism used to generate branch traces, which may be the target hardware (or simulator) or a third-party device e.g. debugger.
- The real-time operating system on which executables to be analyzed by Rapi**Time**<sup>Zero</sup> are to be run.

Different PSPs are needed to support analysis by Rapi**Time**<sup>Zero</sup> when any of the above items are different between two platforms. PSPs that support Rapi**Time**<sup>Zero</sup> analysis also support analysis by Rapi**Cover**<sup>Zero</sup> and Rapi**Task**<sup>Zero</sup>. For more information on how Zero-footprint PSPs support analysis by zero-footprint RVS tools including Rapi**Time**<sup>Zero</sup>, see our [Requirements for zero-footprint RVS analysis Technical note](#).

To see whether we have already developed PSPs compatible with the components on your platform, see the [compatibility tab on our Rapi\*\*Time\*\*<sup>Zero</sup> product page](#). If we have not yet developed PSPs compatible with one or more components of your platform, we may be able to develop them. For more information, contact us at [info@rapitasystems.com](mailto:info@rapitasystems.com).



*A Platform Support Package (PSP) is needed for RVS to interface with the development environment it is used in*



## About Rapita

Rapita Systems provides on-target software verification tools and services globally to the embedded aerospace and automotive electronics industries.

Our solutions help to increase software quality, deliver evidence to meet safety and certification objectives and reduce costs.

## Find out more

A range of free high-quality materials are available at:  
[rapitasystems.com/downloads](http://rapitasystems.com/downloads)

## SUPPORTING CUSTOMERS WITH:

### Tools

#### Rapita **Verification Suite:**

Rapi**Test**

Rapi**Cover**

Rapi**Time**

Rapi**Task**

### Engineering Services

#### V&V Services

Integration Services

Qualification

SW/HW Engineering

Compiler Verification

### Multicore verification

#### **MACH**<sup>178</sup>

Multicore Timing Solution

## Contact

### **Rapita Systems Ltd.**

Atlas House  
York, YO10 3JB  
UK

+44 (0)1904 413945

### **Rapita Systems, Inc.**

41131 Vincent Ct.  
Novi, Mi, 48375  
USA

+1 248-957-9801

### **Rapita Systems S.L.**

Parc UPC, Edificio K2M  
c/ Jordi Girona, 1-3  
Barcelona 08034  
Spain

+34 93 351 02 05



[rapitasystems.com](http://rapitasystems.com)



[linkedin.com/company/rapita-systems](https://www.linkedin.com/company/rapita-systems)



[info@rapitasystems.com](mailto:info@rapitasystems.com)