Ozone-Sim

High-speed simulator

Manual

Document: UM20008
Software Version: 2.10.0
Revision: 0
Date: July 1, 2026

A product of SEGGER Microcontroller GmbH

www.segger.com

Disclaimer

The information written in this document is assumed to be accurate without guarantee. The information in this manual is subject to change for functional or performance improvements without notice. SEGGER Microcontroller GmbH (SEGGER) assumes no responsibility for any errors or omissions in this document. SEGGER disclaims any warranties or conditions, express, implied or statutory for the fitness of the product for a particular purpose. It is your sole responsibility to evaluate the fitness of the product for any specific use.

Copyright notice

Trademarks

Names mentioned in this manual may be trademarks of their respective companies.

Brand and product names are trademarks or registered trademarks of their respective holders.

Contact address

SEGGER Microcontroller GmbH

Ecolab-Allee 5
D-40789 Monheim am Rhein

Germany

Tel. +49 2173-99312-0
Fax. +49 2173-99312-28
E-mail: support@segger.com*
Internet: www.segger.com

*By sending us an email your (personal) data will automatically be processed. For further information please refer to our privacy policy which is available at https://www.segger.com/legal/privacy-policy/.

Manual versions

This manual describes the current software version. If you find an error in the manual or a problem in the software, please inform us and we will try to assist you as soon as possible. Contact us for further information on topics or functions that are not yet documented.

Print date: July 1, 2026

Software	Date	By	Description
2.10.0	2026-07-01	MHA	Initial version

About this document

Assumptions

This document assumes that you already have a solid knowledge of the following:

- The software tools used for building your application (compiler, linker, Integrated Development Environment).
- The C programming language.
- The target processor.

Typographic conventions for syntax

This manual uses the following typographic conventions:

Style	Used for
Body	Body text.
`Keyword`	Text that you enter at the command prompt or that appears on the display (that is system functions, file- or pathnames).
`Parameter`	Parameters in API functions.
`Sample`	Sample code in program examples.
`Sample comment`	Comments in program examples.
Reference	Reference to chapters, sections, tables and figures or other documents.
GUIElement	Buttons, dialog boxes, menu names, menu commands.
Emphasis	Very important sections.

Table of contents

Chapter 1

Introduction

Ozone-Sim is a fast simulator for ARM32 and RISC-V based MCUs. It can be used in two different modes:

GDB server

In GDB server mode, the simulator acts as GDB server and can be used with debuggers such as Ozone, Embedded Studio or even the GDB command line client.

Standalone mode

In standalone mode, Ozone-Sim is provided with an Elf file containing the application to be executed in the simulator. This mode is useful for regression tests inside a continuous integration setup. The simulation will run until the application performs a semihosting `SYS_EXIT` operation or until an unhandled exception occurs. Endless loops are also detected by the simulator and will also cause it to stop the simulation. The stop reason is signalled by the simulator using an exit code.

Chapter 2

Command line arguments

This chapter explains how Ozone-Sim can be configured manually using command line arguments.

Options	
--help	Print a list of supported command line arguments.
--version	Print simulator version and exit.
--verbose	Enable verbose output.

2.1 License management

Ozone-Sim requires its own license. Without a valid license, it can be evaluated for 30 days. Ozone-Sim can be activated using a license string by using the following command:

```
ozone-sim --add-license
```

Ozone-Sim will prompt for the license string and store the license.

Options	
--add-license	Prompts for the license string and stores the license on the system.
--list-licenses	Lists the installed licenses, their expiration dates and length of SUA.

2.2 Running in debug server mode

In debug server mode, the server listens for a connection from a GDB client and provides an additional telnet connection as a semihosting console.

2.2.1 Configuration of network settings

Options	
--port <port-number>	Select the port to listen for GDB clients, default: 2331.

--TelnetPort <port-number>	Select the port to listen for clients for telnet output, default: 2333.
--localhostonly	Listen on localhost only (default behavior on Windows).
--no-localhostonly	Do not listen on localhost only (default behavior on Linux).

2.2.2 Additional configuration options in debug server mode

A GDBInit file can be supplied to Ozone-Sim to prepare perform run-time configuration of the GDB interface (for example, configure semihosting) or perform other preparatory steps. The GDBInit file can contain all supported *monitor* commands (without specifying *monitor* keyword).

Options	
--single-run	Enables single run mode, which means it will terminate after the GDB client connection has been closed.
--no-single-run	Causes the debug server to continue running after a GDB client connection has been closed (default).
--x	Specify a GDBInit file. Executed only when the first client connects to the server.
--xc	Specify a GDBInit file. Executed each time a client connects to the server.

2.3 Running in standalone mode

In standalone mode, the simulator needs to be provided with an Elf executable which is loaded into memory. The simulator will create memory segments for the sections contained in the Elf file. It will start executing the provided Elf file and quit execution once an exception occurs or the application exits via a `SYS_EXIT` semihosting call.

For regression tests, it is suggested to use the semihosting `SYS_EXIT` functionality to provide a return code. See sections *Semihosting* and section *Loading an Elf file* for details.

Options	
--standalone	Launch Ozone-Sim in standalone mode.

2.4 Configuring the simulated MCUs

The simulator can be configured by specifying a known MCU name, a known core name or via a list of features which constitute a core. When the simulator is configured using an MCU name, memory segments will be configured accordingly. In all other cases, memory segments need to be configured manually or can be derived from a provided Elf file.

If no MCU/core/feature options are provided, a “default Cortex-M7” core with ample memory is used.

2.4.1 Known MCUs

The following MCUs can be enabled by specifying the argument **--chip** <chip-name>:

- STM32F407VE: MCU used on the SEGGER Cortex M trace reference board.
- STM32H743ZI: MCU used on the SEGGER STM32H7 trace reference board.
- K66P144M180SF5RMV2: MCU on the Segger emPower board.
- XC7Z007S: Zynq7007S Cortex-A9 MCU.
- default: A Cortex-M7 with flash memory banks of size `0x00200000` at base addresses `0x00000000`, `0x01000000`, `0x02000000`, `0x04000000`, and `0x08000000`, as well as RAM segments of the same size at `0x10000000`, `0x20000000`, `0x30000000`, and `0x40000000`.

Options	
--chip <chip-name>	Name of the chip to simulate.

2.4.2 Specifying processor features

For maximum flexibility, processor features can be specified individually by prefixing them with **+** as a command line argument. They can also be provided to adjust or fine-tune the behavior of known chips or CPUs (see below). By specifying a feature with the prefix **!** on the command line, it will be disabled.

Options	
+<feature>	Enable architecture property for the simulated CPU.
!<feature>	Disable architecture property for the simulated CPU.
--endian <little|big>	Select the endianness of the simulated CPU, default: little endian.
--help-arm32	Lists ARM32 related features.
--help-riscv	Lists RISC-V related features.

2.4.2.1 Configuring RISC-V CPUs

For RISC-V based CPUs, 32-bit and 64-bit variants can be enabled using the **+RV32** and **+RV64** flags, respectively. Extensions can be enabled using the flags listed in section *Supported RISC-V extensions*.

For example, a 32-bit RISC-V CPU with the **M** extension can be configured like this:

```
ozone-sim +RV32 +m […]
```

2.4.2.2 Configuring ARM CPUs

ARM CPUs can be configured in several ways:

Providing a CPU core flag

ARM Cortex CPUs can be directly enabled using the flags listed in section *Supported ARM32 CPUs*. For example, to configure an ARM Cortex-M4, the following command can be used:

```
ozone-sim +arm-cortex-m4 […]
```

Providing an architecture and feature flags

ARM CPUs can also be configured by specifying an architecture from the list of *Supported ARM instruction sets*. Additional features from the lists of *ARM32 floating point features* and *Miscellaneous ARM32 features* can be used to fine-tune the CPU.

The first feature flag provided has to be either an architecture or CPU name starting with **arm…** to allow Ozone-Sim to detect that an ARM CPU is being configured. For example, to configure a Cortex-M4, the following arguments can be used:

```
ozone-sim +armv7e-m +vfp4d16sp +hwdiv +hwdiv-arm +mp […]
```

2.4.3 Configuration of memory

The simulator differentiates between executable and non-executable memory, because for executable memory, additional memory for an instruction cache needs to be reserved. The instruction cache requires roughly 24 bytes per byte of simulator memory.

Memory can additionally be configured as writable. Both memory addresses and sizes are provided as hexadecimal numbers (leading `0x` can be omitted).

Options

--mem R[W][X],<start-addr>,<size>	Specifies an optionally writable and executable memory segment starting at <start-addr>, with size <size>.
--mem R[W][X],<start-addr>-<end-addr>	Specifies an optionally writable and executable memory segment ranging from <start-addr> to <end-addr>.

2.4.4 Configuration of the vector table

For known MCUs which are specified via the **--chip** argument, the location of the vector table is configured automatically. When the simulator is configured using core or feature options, the vector table is assumed to be located at the start of the first executable memory segment.

The vector base address can be configured manually using the **--vector** <base-address> argument, with <base-address> being a hexadecimal number.

Options	
--vector <base-address>	Address of vector table (hexadecimal).

2.4.5 Sample configuration

To configure an STM32F407VE, the following equivalent command line arguments can be used. Since the vector table is located at the start of the first executable memory segment, the **--vector** argument does not need to be specified.

Specifying the chip name:

```
ozone-sim.exe --chip STM32F407VE
```

Specifying the core and the memory layout:

```
ozone-sim.exe +arm-cortex-m4 --mem RX,0x08000000,0x00080000
    --mem RWX,0x10000000,0x00010000 --mem RWX,0x20000000,0x00020000
```

Specifying the features of the core and the memory layout:

```
ozone-sim.exe +armv7e-m +vfp4d16sp +hwdiv +hwdiv-arm +mp
    --mem RX,0x08000000,0x00080000 --mem RWX,0x10000000,0x00010000
    --mem RWX,0x20000000,0x00020000
```

2.5 Additional configuration options

Options	
--halt-on-loop	Halt the simulator when an instruction branches to itself, for example in a placeholder for an exception handler or when a program runs into a terminating endless loop. When this condition is encountered, the simulator will announce that the target has stopped to the GDB client or stop execution when running in standalone mode.
--trap-div0	Generate an exception on division by 0.
--excdbg <nSteps>	Enable exception debugging with <nSteps>.

2.6 Logging

Ozone-Sim can write a detailed logfile. In GDB server mode, it includes all interactions with the GDB clients. In standalone mode, it contains the semihosting output and the reason why the simulation was terminated.

Options	
--log <filename>	Write a log to <filename>.

2.7 Loading an Elf file

An Elf life can be loaded by Ozone-Sim on startup both in GDB server mode and in standalone mode. The path to the Elf file has to be provided as the last command line argument to the simulator. The simulator will automatically allocate memory regions corresponding to the segments inside the Elf file.

ozone-sim --standalone [--bss] [--stackpointer <sym>] <elf-file>

Options	
--bss	Creates memory segments for the BSS segments in Elf files.
--stackpointer <sym>	Initialize stackpointer to symbol <sym>.

Chapter 3

Semihosting

Semihosting file access can either be handled by the simulator or by the GDB client. The simulator will only perform file access in the directory provided via the **--file-root** <path> option.

The semihosting command line arguments can be provided using the **--arg** <argument> option. Multiple command line arguments can be specified by specifying this argument multiple times. For example, a regression test executable might need the name of the test cases to execute. This could be achieved with the following statement:

```
ozone-sim --standalone --bss --arg multiplication
    --arg addition --arg quicksort CortexM-regression.elf
```

The semihosting exit function supports both the exit codes provided in the *semihosting specification* as well as codes in the range 0 - 255. In standalone mode, a semihosting exit call will cause the simulator to exit with the code provided to the semihosting call.

Options	
--arg <arg>	Give argument to main(). Can be specified multiple times.
--file-root <dir>	Allow semihosting file IO in <dir>.

Chapter 4

Supported ARM instruction sets

The simulator is able to execute the following instruction sets:

- ARMv5
- ARMv6-M
- ARMv7-M/ARMv7E-M
- ARMv7-A/-R
- ARMv8-M
- ARMv8-A/-R (AArch32)
- ARMv8.1-M (no half-precision foating point, supports only a subset of MVE)

These architectures can be enabled using the following feature specifiers:

Architecture flag	Description
+armv4	ARMv4 architecture
+armv4t	ARMv4t architecture
+armv5t	ARMv5t architecture
+armv5te	ARMv5te architecture
+armv5tej	ARMv5tej architecture
+armv6	ARMv6 architecture
+armv6-m	ARMv6m architecture
+armv6j	ARMv7a architecture
+armv6k	ARMv6k architecture
+armv6kz	ARMv6kz architecture
+armv6s-m	ARMv6sm architecture
+armv6t2	ARMv6t2 architecture
+armv7-a	ARMv7a architecture
+armv7-m	ARMv7m architecture
+armv7-r	ARMv7r architecture
+armv7e-m	ARMv7em architecture
+armv7k	ARMv7a architecture
+armv7s	ARMv7a architecture
+armv7ve	ARMv7ve architecture

Architecture flag	Description
+armv8-a	ARMv8a architecture
+armv8-m.base	ARMv8m Baseline architecture
+armv8-m.main	ARMv8m Mainline architecture
+armv8-r	ARMv8r architecture
+armv8.1-m.main	ARMv81m Mainline architecture

4.1 Supported ARM32 CPUs

The following ARM32 CPU types can be enabled using flags:

CPU flag	Description
+arm-cortex-a12	CPU arm-cortex-a12
+arm-cortex-a15	CPU arm-cortex-a15
+arm-cortex-a17	CPU arm-cortex-a17
+arm-cortex-a32	CPU arm-cortex-a32
+arm-cortex-a35	CPU arm-cortex-a35
+arm-cortex-a5	CPU arm-cortex-a5
+arm-cortex-a53	CPU arm-cortex-a53
+arm-cortex-a57	CPU arm-cortex-a57
+arm-cortex-a7	CPU arm-cortex-a7
+arm-cortex-a72	CPU arm-cortex-a72
+arm-cortex-a73	CPU arm-cortex-a73
+arm-cortex-a8	CPU arm-cortex-a8
+arm-cortex-a9	CPU arm-cortex-a9
+arm-cortex-m0	CPU arm-cortex-m0
+arm-cortex-m0plus	CPU arm-cortex-m0plus
+arm-cortex-m1	CPU arm-cortex-m1
+arm-cortex-m23	CPU arm-cortex-m23
+arm-cortex-m3	CPU arm-cortex-m3
+arm-cortex-m33	CPU arm-cortex-m33
+arm-cortex-m35p	CPU arm-cortex-m35p
+arm-cortex-m4	CPU arm-cortex-m4
+arm-cortex-m52	CPU arm-cortex-m52
+arm-cortex-m55	CPU arm-cortex-m55
+arm-cortex-m7	CPU arm-cortex-m7
+arm-cortex-m85	CPU arm-cortex-m85
+arm-cortex-r4	CPU arm-cortex-r4
+arm-cortex-r4f	CPU arm-cortex-r4f
+arm-cortex-r5	CPU arm-cortex-r5
+arm-cortex-r52	CPU arm-cortex-r52
+arm-cortex-r52plus	CPU arm-cortex-r52plus
+arm-cortex-r7	CPU arm-cortex-r7
+arm-cortex-r8	CPU arm-cortex-r8

4.2 ARM32 floating point features

The following features control floating point behavior. They can be specified with either **+** to enable them or with **!** to disable them. Note that these flags have to be specified after the CPU or architecture flags.

FP flag	Description
+d32	Extend FP to 32 double registers
+fp-armv8sp	Enable ARMv8 FP with no double precision
+fpregs	Enable FP registers
+fp-armv8d16sp	Enable ARMv8 FP with only 16 d-registers and no double precision
+fp64	Floating point unit supports double precision
+fp-armv8d16	Enable ARMv8 FP with only 16 d-registers
+fpregs64	Enable 64-bit FP registers
+fp-armv8	Enable ARMv8 FP
+vfp2	Enable VFP2 instructions
+vfp2sp	Enable VFP2 instructions with no double precision
+vfp3	Enable VFP3 instructions
+vfp3d16	Enable VFP3 instructions with only 16 d-registers
+vfp3d16sp	Enable VFP3 instructions with only 16 d-registers and no double precision
+vfp3sp	Enable VFP3 instructions with no double precision
+vfp4	Enable VFP4 instructions
+vfp4d16	Enable VFP4 instructions with only 16 d-registers
+vfp4d16sp	Enable VFP4 instructions with only 16 d-registers and no double precision
+vfp4sp	Enable VFP4 instructions with no double precision

4.3 Miscellaneous ARM32 features

The following additional ARM32 features are supported. They can be specified either with **+** to enable them or with **!** to disable them. Note that these flags have to be specified after the CPU or architecture flags.

Feature	Description
+acquire-release	Has v8 acquire/release (lda/ldaex etc) instructions
+db	Has data barrier (dmb/dsb) instructions
+dfb	Has full data barrier (dfb) instruction
+dsp	Supports DSP instructions in ARM and/or Thumb2
+hwdiv	Enable divide instructions in Thumb
+hwdiv-arm	Enable divide instructions in ARM mode
+mve	Support M-Class Vector Extension with integer ops
+mve.fp	Support M-Class Vector Extension with integer and floating ops
+neon	Enable NEON instructions
+noarm	Does not support ARM mode execution
+ret-addr-stack	Has return address stack

Feature	Description
+strict-align	Disallow all unaligned memory access
+thumb2	Enable Thumb2 instructions

Chapter 5

Supported RISC-V extensions

Ozone-Sim supports 32-bit and 64-bit RISC-V instruction sets. The supported extensions are listed in the table below along with the flags required to activate them.

Extension	Flag	Description
A	**+a**	Atomic Instructions
C	**+c**	Compressed Instructions
D	**+d**	Double-Precision Floating-Point
E	**+e**	Embedded Instruction Set with 16 GPRs
F	**+f**	(Single-Precision Floating-Point
M	**+m**	Integer Multiplication and Division

Chapter 6

Support

How to Report Bugs

Users are kindly asked to include as much as possible of the following information in Ozone-Sim bug reports (in order of importance):

- Ozone-Sim (and Ozone, if used) version number
- Command line parameters supplied to Ozone-Sim
- A detailed description of the problem
- The operating system and OS version of the PC
- A logfile of interactions with a GDB client, see section *Logging* for details.

Users without a support agreement with SEGGER are kindly asked to report bugs at the general room of *SEGGER's forum*.

Users which are entitled to support should use the contact information below.

Contact Information

SEGGER Microcontroller GmbH

Ecolab-Allee 5
D-40789 Monheim am Rhein

Germany

Tel. +49 2173-99312-0
Fax. +49 2173-99312-28
E-mail: support@segger.com
Internet: www.segger.com

