SEGGER

The Embedded Experts

Software Tools

Embedded Studio/PRO

The leading multi-platform IDE

Efficient & flexible toolchain

Embedded Studio comes with SEGGER and GCC toolchains: Ready-to-run. Developers can also seamlessly switch to Embedded Studio from other IDEs. In single chip system, code space is a precious resource. SEGGER's own tool-chain squeezes code to the last byte without compromising on execution time.

Source editor

The first-class Source Code Editor supports syntax highlighting, automatic code indentation, and matching bracket highlighting. Additionally, it includes code completion, configurable code, and comment templates.

Integrated debugger

The seamlessly integrated J-Link/J-Trace debug probes provide the foundation that boosts Embedded Studio's powerful features. With the core simulator, applications can be tested before the hardware is available. The debugger tracks the application's OS to visualize what the tasks are up to or how much stack each task uses. OS awareness can easily be added for any operating system.

Project management

Managing & organizing projects in one place with the project manager facilitates a project setup to fit the developers' needs. With multi-project solutions, dynamic folders, and property inheritance, Embedded Studio offers the utmost in flexibility.

Embedded Studio PRO package

With the powerful integrated development environment Embedded Studio, extensive support for the most widely used microcontrollers, a complete embedded software suite emPower OS, and the industry-leading J-Link PLUS debug probe SEGGER offers a one-stop solution for your projects.

IDE for Linux

SEGGER's Linux Studio makes top-rated development environment available to Linux developers.

Key features

- All-in-one solution
- SEGGER's Friendly License (SFL)
- GCC & SEGGER C/C++ tool-chains included
- Multi-threaded build minimizes build times
- Multi-platform: Windows, Linux or Mac

USE CASES

➔ Cost-effective one-stop solution for embedded systems software development

➔ Comprehensive embedded operating system

www.segger.com/embedded-studio

www.segger.com/linux-studio

emFloat, emRun & emRun++

Performance tuning

Runtime library performance can have a huge impact on application speed. emRun's highly advanced low-level implementations can also be fine-tuned for speed or size. With assembly optimized variants, performance can be even more optimized by using the target platform to its full potential.

Memory requirements

emRun offers significant savings in flash memory, partly by having some functions are hand-coded in assembly language, but mostly through a structure that minimizes internal library dependencies.

Library verification

We created a verification test suite to test emRun. It checks the entire functionality of all library functions, including the entire floating point library with all corner cases.

Modern C++ features

emRun++ implements classes and functions to C++ standards. It also supplements the language features and incorporates the complete feature set of the C++17 standard defined by ISO.

Exception handling

C++ defines the use of exceptions. In C, avoiding a fault requires manual recovery and that an error be passed up to all callers.

Low-level support

C++ compilers define an application binary interface (ABI) which, for example, defines how objects are arranged, how name mangling works, or how virtual functions are implemented.

Dynamic memory allocation

Modern C++ applications rely on dynamic memory allocation. Objects are present in memory only while they are being used.

Key features

emRun

- High performance, with time-critical routines written in assembly language
- Significant code size reduction
- Configurable for high speed or small size
- Includes SEGGER's optimized floating-point library emFloat
- Designed for use with various toolchains
- EABI compatible functions
- Minimum RAM usage
- No heap requirements
- No viral licensing, no attribution clause

emRun ++

- Comprehensive C++ standard library
- Compatibility with common C++ standards, C++17
- Complete integration with emRun
- Dynamic memory management, optimized for embedded systems
- Exception handling, including target unwinding on all supported targets

emFloat

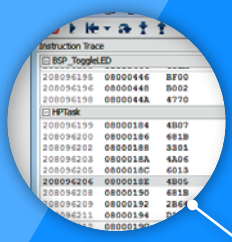
- Small code size, high performance
- Plug-and-play: Can easily replace default floating point library, delivering better performance with less code.
- Flexible licensing, for integration into user applications or toolchains.
- C-Variant can be used on any 8/16/32/64-bit CPU.
- Hand-coded, assembly-optimized variants for RISC-V and Arm
- Fully reentrant
- No heap requirements

www.segger.com/emrun

Ozone

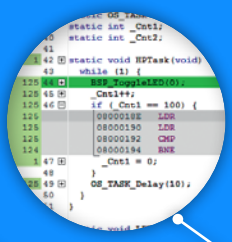
The J-Link debugger and performance analyzer

Ozone is a full-featured multi-platform debugger and performance analyzer for embedded applications. With Ozone it is possible to debug any embedded application at C/C++ source and assembly level. Ozone can load applications built with any toolchain/IDE or debug the target's resident application without having any code available. Ozone includes extensive debug information windows and makes the best use of the of performance of the J-Link debug and J-Trace trace probe. The user interface is designed to be used intuitively and is fully configurable. All windows can be moved, re-sized and closed. Ozone is available for Windows, macOS and Linux.



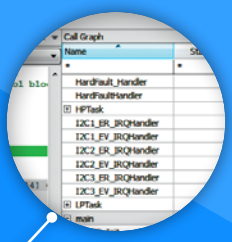
Instruction trace

The basic information provided by trace data is the instruction trace. When the target is halted, Ozone shows the most recently executed instruction in its Instruction Trace Window. This allows you to analyze what your system did last.



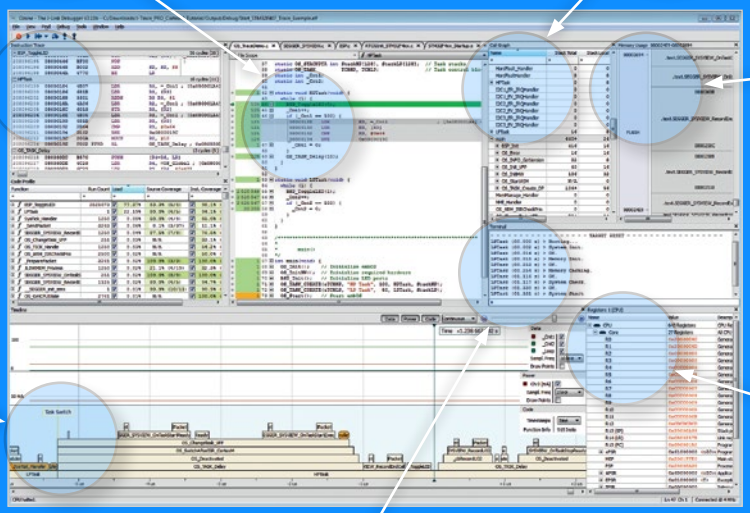
Source code

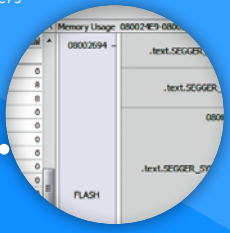
The Source Code Viewer enables navigation through the target application. It shows the current program executions and lets the developer modify the target behaviour. The inline disassembly provides deeper insight per source line. Source Code can be directly edited in the Source Code Viewer.



Call Graph

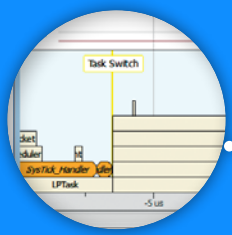
Call Graph shows static information about paths of function calls in your application to analyze call depth and stack requirements. It highlights recursions and the use of function pointers





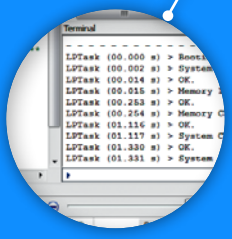
Memory usage

Ozone's Memory Usage Window provides a graphical representation of the memory content of the embedded application. It provides a quick overview where symbols are placed and how much space is used.



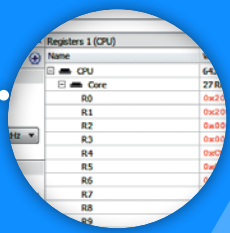
Timeline

Ozone can display runtime information of the embedded application in the unified Timeline window. The Code Timeline shows the instruction trace data as a graphical representation of the Call Stack over time. The Data Sampling tracks symbols and arbitrary C style expressions at time resolutions of down to 1 microsecond and visualizes the values in the Data Timeline. Using the Power Sampling feature of J-Link, the Power Timeline captures and displays the power consumption of the target device.



Terminal

Ozone can capture printf-output by the embedded application via SEGGER's Real Time Transfer (RTT) technology that provides extremely fast IO coupled with low microcontroller intrusion.



Registers

The current CPU registers are shown in Ozone's Registers Window.
In addition to the basic CPU registers, Ozone can also display memory-mapped peripheral registers (SFRs).

www.segger.com/ozone

SystemView

Analyzing embedded systems

Extended debugging & analysis

Analyzing the runtime behavior of a system and the interaction between tasks and amongst interrupts can be employed for various purposes, such as the documentation of new systems, validation of desired behavior, examination of erratic behavior such as in interrupt handling, or for the analysis and improvement of performance. SystemView PRO provides advanced filters and unlimited event recording.

Continuous real-time recording

Enabled by the J-Link Real-Time Transfer (RTT) technology, SystemView monitors events, interrupts, etc. in real-time. The data transfer rate of 2 MBytes/s clearly surpasses the bandwidth of comparable solutions. The overhead is less than 1 µs per call (measured for a 200 MHz Cortex-M). The technology requires only the standard debug interface. The extended debug and analysis capabilities are available for Cortex-M0-based systems as well.

With a J-Link debug probe and the SEGGER Real-Time Transfer technology (RTT), SystemView can continuously record target execution in real-time, while the target is continues running. This enables the analysis of the system behavior over a longer period of time without requiring a large target memory buff.

Single-shot recording

When the target device does not support RTT or when no J-Link is used, SystemView can still record data in its buffer until it is filled. Single-shot recording offers insight into the startup sequence or an event-triggered activity in any system for a decent amount of time.

Post-mortem mode

Post-mortem analysis is similar to single-shot recording, with one difference: SystemView events are continuously recorded. When the target buffer is filled, older events are overwritten, Reading the buffer provides the latest recorded events. Post-mortem analysis can provide information of long-time system tests and help analyze system crashes.

Key features

- Capture tasks, interrupts, timers, resources, API calls & user events
- Live analysis & visualization of captured data
- Minimally system intrusive
- CPU-independent
- Multi-platform: Linux, macOS & Windows