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