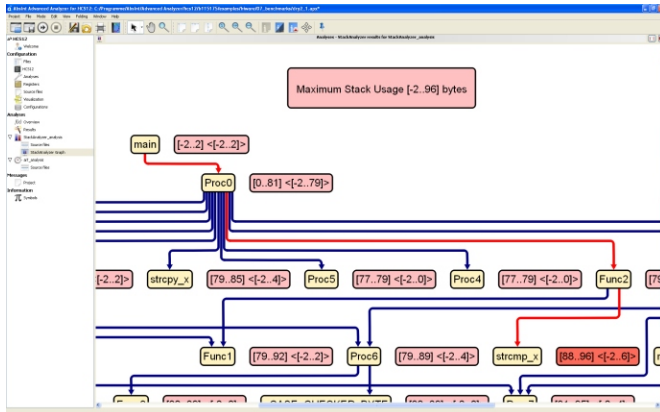


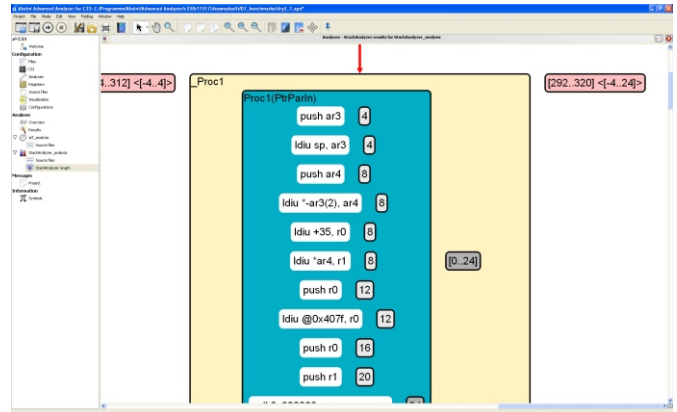
# StackAnalyzer – Stack Usage Analysis

Stack overflow is now a thing of the past.

StackAnalyzer automatically determines the worst-case stack usage of the tasks in your application. Stack height differences are shown as annotations in the call graph and control flow graph.



Call graph with stack usage annotations



Control flow graph with stack usage annotations

## Why do you need StackAnalyzer?

Stack memory has to be allocated statically by the programmer. Underestimating stack usage can lead to serious **runtime errors** which can be difficult to find. Overestimating stack usage means a waste of memory resources.

- **StackAnalyzer** provides **automatic** tool support to calculate the stack usage of your application. The analysis results are valid for **all inputs** and each task execution.
- **StackAnalyzer** does not rely on debug information; it analyzes the binary executable code as executed in the final system.
- The analysis results provide valuable feedback in **optimizing** the stack usage of your application.
- **StackAnalyzer** not only reduces development effort but also helps to **prevent runtime errors** due to stack overflow.

## Supported processors and compilers

- C16x/XC16x/ST10/Tasking
- C16x/ST10/KEIL
- PowerPC/Diab
- PowerPC/gcc (VxWorks)
- PowerPC/GHS
- M68k/gcc/HP68K
- TMS320C3x/TI
- NEC V850/GHS
- ARM/ TI/ARM
- H8 /300+H8S/ 2x00/ IAR
- x86 / gcc
- x86rm / ic86
- HCS12/STAR12/M68HC12/Metrowerks
- HCS12/STAR12/M68HC12/Cosmic
- HCS12X/Metrowerks/Cosmic
- ARC/Metaware
- TriCore/Tasking/gcc
- LEON2/LEON3/gcc

For further targets, please contact us.

## About AbsInt

AbsInt Angewandte Informatik GmbH provides advanced tools and services in the areas of compiler optimization, static program analysis, and worst-case execution time prediction.