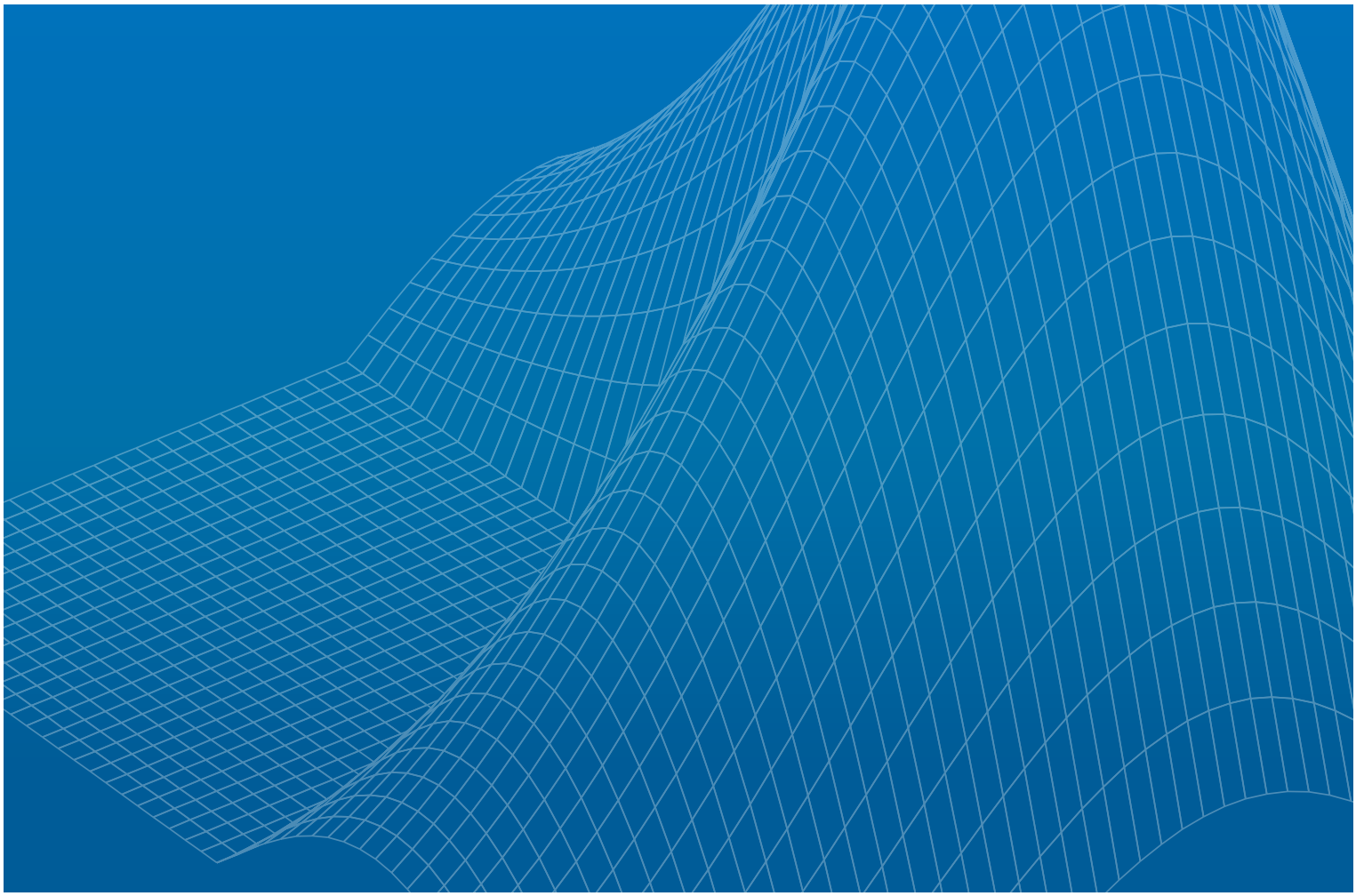


Integrating Static Analysis into Your Embedded Software Development Workflow

Featuring Polyspace Static Analysis Products and the Wind River Software Development Toolchain

POLYSPACE®
Code Verification Products

WIND™
AN INTEL COMPANY

WHITE PAPER

MathWorks®

As software gets more complex, the requirements for software quality increase. This is especially true for real time embedded systems such as GNC, flight data recorders, ADAS, and engine control systems which cannot afford failures. The embedded software development process is therefore quite elaborate and requires an integrated toolchain for automation, efficiency, and quality. The primary components of any such toolchain are the compiler and the integrated development environment. Advanced static analysis tools are increasingly becoming an important and ubiquitous tool in the software developer's arsenal.

One of the key challenges in the adoption of the tools by the software development community has been the lack of integration of static analysis tools into the development environment or the toolchain. In this whitepaper, we will discuss how Polyspace® static analysis tools can be integrated into the Wind River® software development toolchain, as well as the benefits of such an integration.

Wind River Software Development Toolchain

Wind River Diab Compiler

Diab Compiler is a complete toolkit for embedded application development, including C and C++ compilers, assemblers, linkers, utilities, and standard libraries for a variety of target CPU architectures such as Renesas RH850, Infineon TriCore, PowerPC, and ARM.

Wind River Workbench

Workbench is a collection of tools that accelerates time-to-market for advanced developers building embedded devices. It offers an end-to-end, open standards-based suite for device software design, development, debug, test, and management. It enables organizations to standardize on a common environment for software development based on Eclipse framework.

VxWorks

The VxWorks® family of products delivers a real-time operating system with scalability, safety, security, and virtualization capabilities for building intelligent, connected systems.

Polyspace Static Analysis Products

Polyspace products provide a complete static analysis solution to comply with coding standards such as MISRA and CERT C, detect defects and security vulnerabilities, calculate code metrics, and, most importantly, prove the absence of run-time errors in your application.

Polyspace Bug Finder

Polyspace Bug Finder™ identifies run-time errors, concurrency issues, security vulnerabilities, and other defects in C and C++ embedded software. Using static analysis, including semantic analysis, Polyspace Bug Finder analyzes software control, data flow, and interprocedural behavior. By highlighting defects as soon as they are detected, it lets you triage and fix bugs early in the development process.

Polyspace Code Prover

Polyspace Code Prover™ is a sound static analysis tool that proves the absence of overflow, divide-by-zero, out-of-bounds array access, and certain other run-time errors in C and C++ source code. It produces results without requiring program execution, code instrumentation, or test cases. Polyspace Code Prover uses semantic analysis and abstract interpretation based on formal methods to verify software interprocedural, control, and data flow behavior. You can use it on handwritten code, generated code, or a combination of the two. Each operation is color-coded to indicate whether it is free of run-time errors, proven to fail, unreachable, or unproven.

By verifying the dynamic properties of embedded applications, abstract interpretation encompasses all possible behaviors of the software and all possible variations of input data, including how software can fail. It gives proof of code correctness, thereby providing strong assurance of code reliability.

By using bug finding and code proving tools, engineering teams can reduce costs while accelerating the delivery of reliable embedded systems.

Solar Impulse Saves 1-2 Person Years Through Static Analysis

Solar Impulse used Polyspace products on their avionics software, which comprises 260K lines of code. They put Polyspace on the server, where it integrated with Eclipse and ran in the background. In one case, the team needed to find a latent bug in their throttle box, which was generating an incorrect temperature. The error evaded manual code reviews by three engineers. They pinpointed the error with Polyspace products – without test cases or compilation. Solar Impulse reported saving 1-2 person years by using the products.

Challenges in the Adoption of Static Analysis

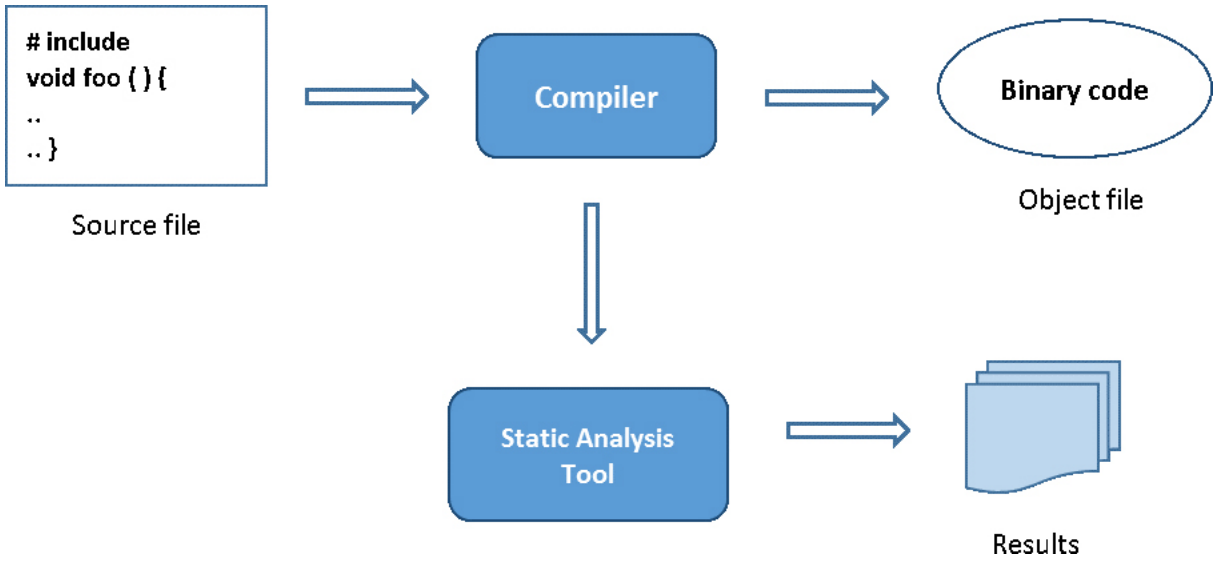
A couple of key challenges that hinder the effective use of static analysis tools are:

- Difficulty in setting up the analysis
- A focus on static analysis in the late stages of the development workflow

Both these challenges are a consequence of the lack of integration of static analysis tools in the developer's toolchain. Setting up an analysis often involves translating the compiler-specific features such as preprocessor options or the manual configuration of the source code and include paths. It can also be a burden on the developer to acclimate and use a different interface to set up their code for analysis or to review the analysis results. As a result, developers resist the use of static analysis tools.

Static analysis, when used, is often pushed to the later stages of the development cycle and often gets handed down to the quality and test teams. This leads to an ineffective workflow because issues are discovered too close to the release and, in some cases, require significant time and engineering resources to resolve issues that would have been highlighted much earlier.

A good solution to address this issue is to integrate static analysis tools into the build process. One way to accomplish this and automate the process is to configure and launch the analysis by reading the compilation process. This can be done either through the command line or by integrating the tool into the developer's IDE and staying in a single interface.



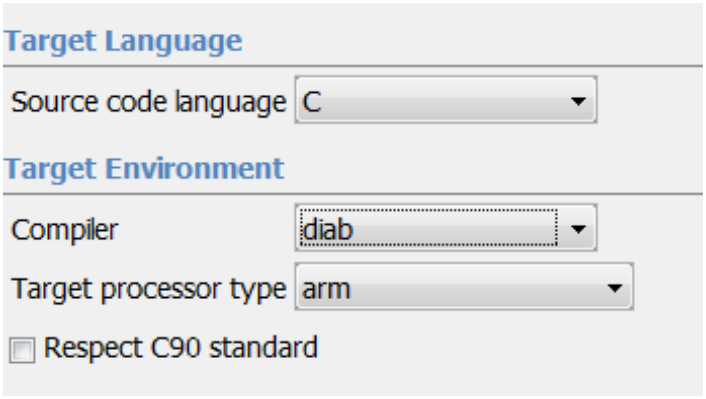
Integrating static analysis into the compilation and build process

Delphi Diesel Systems Gets to Root Cause Faster with Polyspace Products

Because fuel-injection systems increasingly rely on electronics, the reliability of embedded software is crucial. Poor reliability can lead to product recalls. Using Polyspace products, the team at ***Delphi Diesel Systems*** automatically identified run-time errors in the diesel fuel-injection system's embedded software, eliminating time-consuming robustness tests that frequently provided inadequate results.

Integrating Polyspace Static Analysis Tools into the Development Workflow

Polyspace can be integrated into the Wind River software development toolchain. You can plug it into the Workbench IDE, where you can configure Polyspace to read the makefile to extract the compilation settings within the IDE and create a Polyspace project. Polyspace supports the Diab Compiler out of the box, making this a seamless step.



Polyspace supports the Diab Compiler for architectures such as the PowerPC, ARM, and TriCore.

Furthermore, if you are using VxWorks, the support for Diab Compiler permits very fine-grained handling of VxWorks projects. Polyspace products support the VxWorks multi-tasking model. Polyspace can automatically detect and configure the multi-tasking options such as the entry points and the critical sections. The primitives to start a task (taskSpawn) and to lock/unlock a resource (semTake/semGive from semLib.h) are recognized (with -enable-concurrency-detection option) and their semantics are correctly interpreted.

Polyspace Bug Finder extends compiler capabilities to identify programming mistakes or semantic issues as you write code. You can use the output of the Polyspace analysis to reduce manual review costs in code review. You can use Polyspace static analysis results for a defect audit and use the software quality metrics as part of the architecture/design analysis of the code, and avoid defects in the later stages such as through a fix-as-you-go agile approach.

Learn to Integrate Polyspace into Your Workflow

Debunking Misconceptions About Static Analysis

Comprehensive Static Analysis Using Polyspace Products

Explore How Other Engineers Use Polyspace

Solar Impulse Uses Polyspace Static Analysis for Solar Airplane

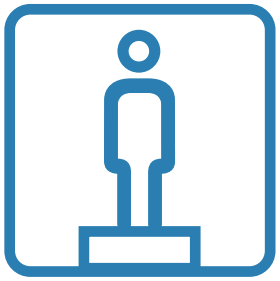
Ralph Paul, Solar Impulse

Introducing Polyspace Products into the Software Development Process

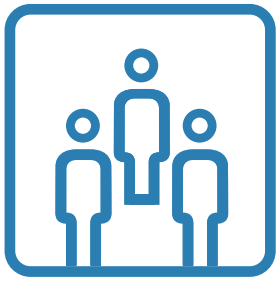
Eileen Davidson, Ford Motor Company

Interested in Learning More?

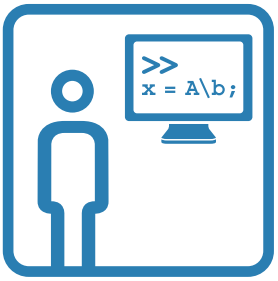
Select your role to explore how Polyspace can help you.



Software Development Manager



Software Engineer or Developer



Test or Quality Engineer

Request a Trial | Speak to an Expert

93106v00 03/17