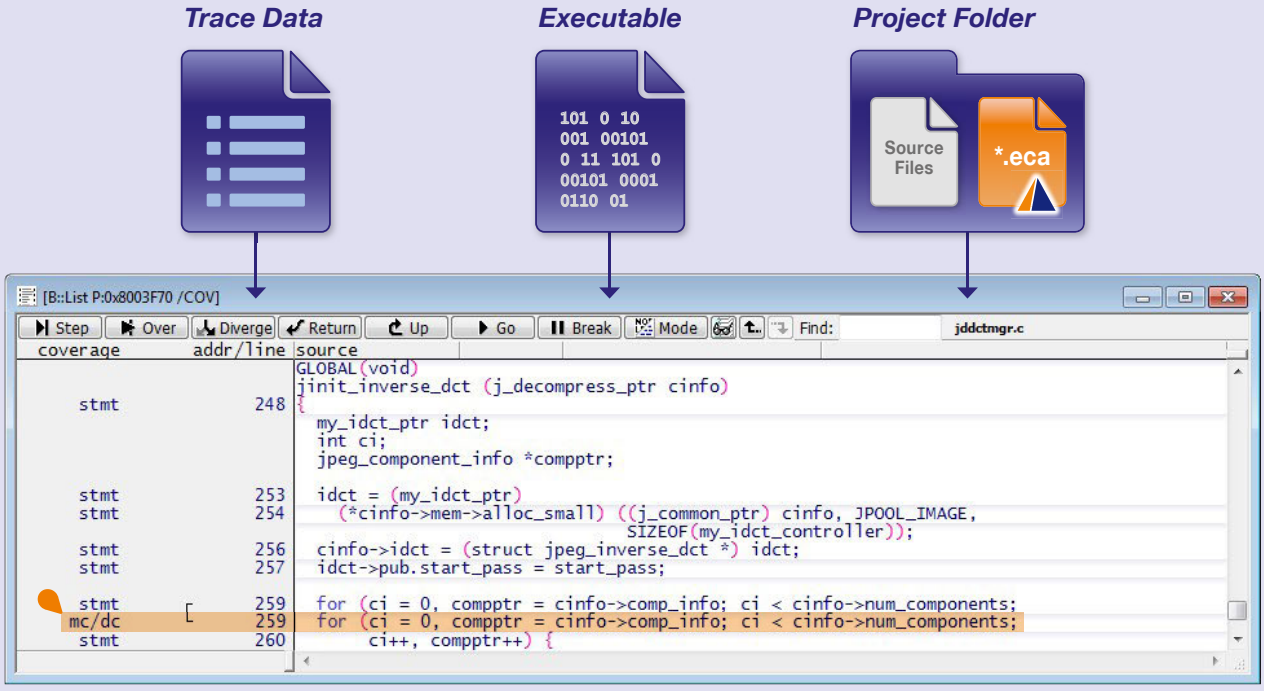
Trace-based MCDC Coverage

MCDC Coverage



In March 2018, Lauterbach will unveil its trace-based MCDC coverage. TRACE32 now supports all important code coverage metrics at the source code level.

Definitions According to DO-178C [3]

Statement Coverage: Every statement in the program has been invoked at least once.

Decision Coverage: Every point of entry and exit in the program has been invoked at least once and every decision in the program has taken on all possible outcomes at least once.

Modified Condition/Decision Coverage: Every point of entry and exit in the program has been invoked at least once, every condition in a decision in the program has taken all possible outcomes at least once, every decision in the program has taken all possible outcomes at least once, and each condition in a decision has been shown to independently affect that decision's outcome. A condition is shown to independently affect a decision's outcome by: (1) varying just that condition while holding fixed all other possible conditions, or (2) varying just that condition while holding fixed all other possible conditions that could affect the outcome.

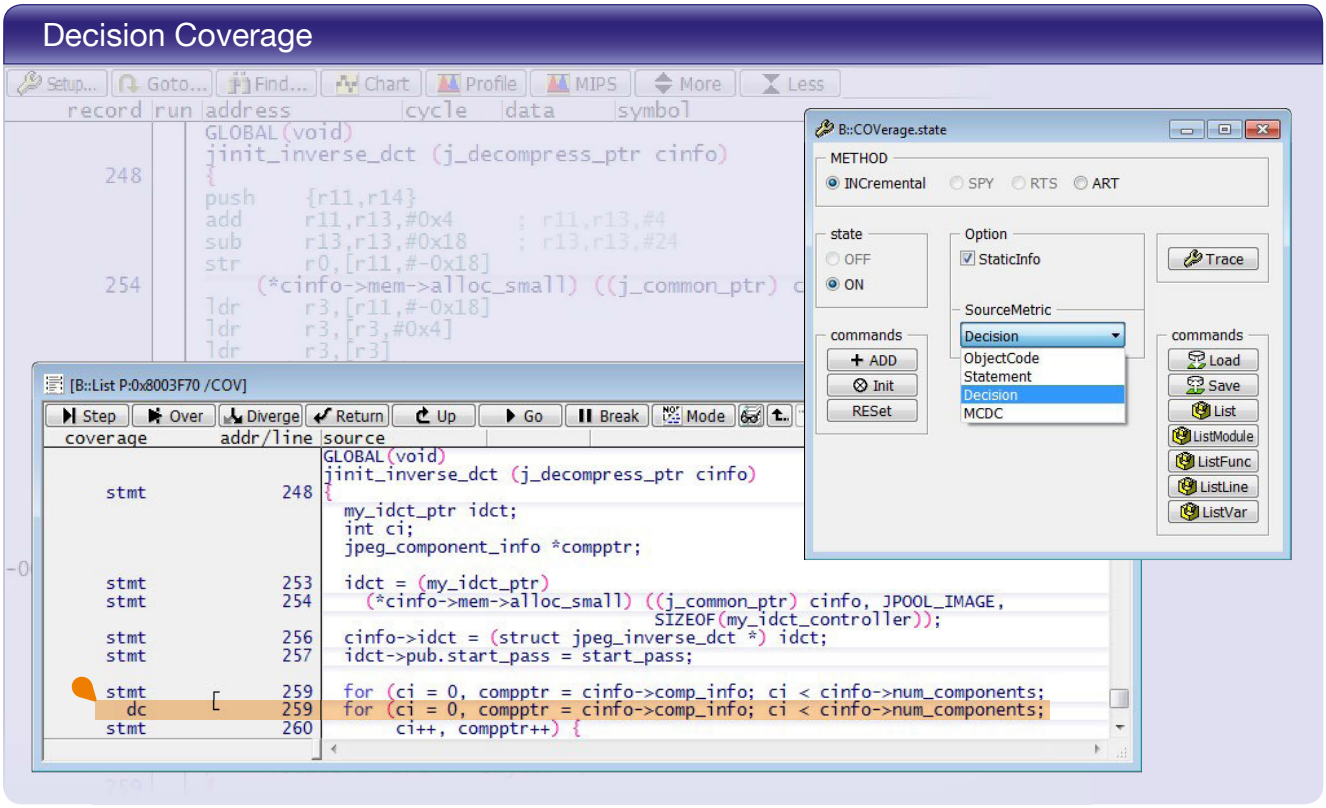
During the development of safety-critical systems, code coverage processes are used in order to demonstrate that the software was tested thoroughly and comprehensively. Software development standards such as ISO 26262 and DO-178C have stipulated that proof of code coverage is a mandatory part of the development cycle.

Trace-based Coverage

As a market leader in real-time trace tools, Lauterbach provides trace-based code coverage measurements which do not require any instrumentation of the target code. The information on the program execution is at first tracked at object code level. This allows the following coverage metrics to be easily proven:

- **Object statement coverage** proves that each assembler instruction has been executed at least once during the test.
- **Object branch coverage** proves that each conditional jump was taken at least once and that it was also not taken at least once.

Code coverage measurements at object code level have always been part of the capabilities of all



TRACE32 trace tools. With the addition of statement and decision coverage in February 2017, Lauterbach tools also provide proof of source code level coverage (see “Decision Coverage” figure on the top of this page). Many customers now also want to use their TRACE32 trace tools for performing the MCDC coverage measurements which are increasingly being mandated.

MCDC Code Coverage

Previously, it was the opinion of many that the proof of object branch coverage was an adequate replacement for the MCDC coverage measurement at the source code level. However, in the aviation industry, the Commercial Aviation Safety Team (CAST) has shown a clear opposition to this view (refer to [1]).

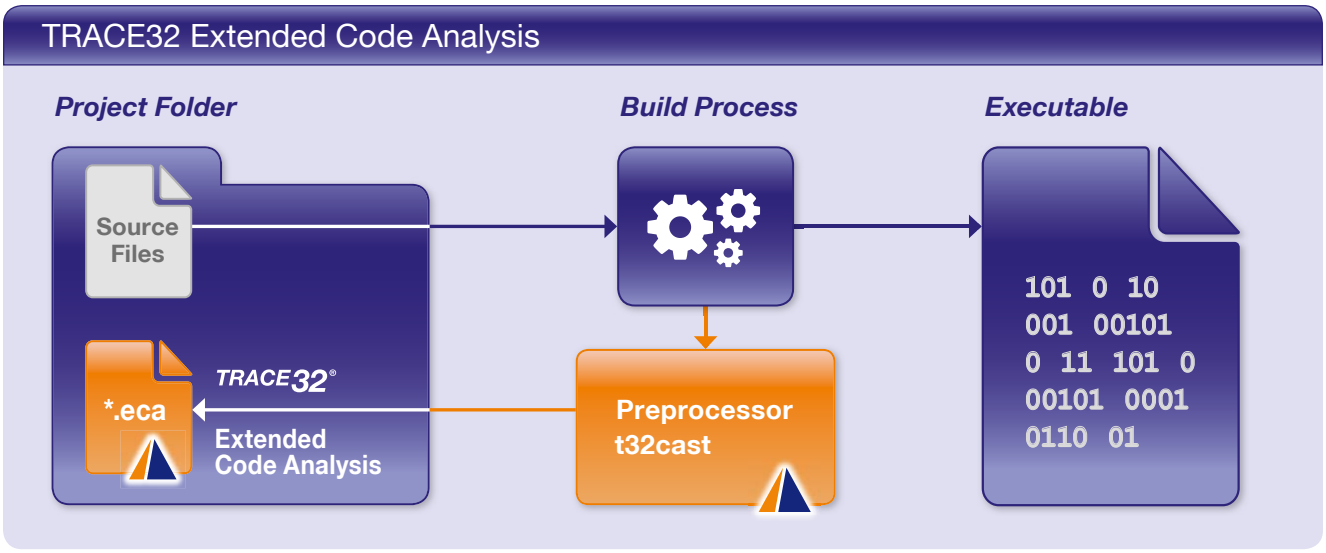
Currently, most people rely on source code instrumentation in order to be able to collect MCDC coverage data. The Lauterbach engineers set themselves the goal of providing MCDC coverage without having to alter the target code in any way. They studied many white papers and publications available on this subject. Each champions its own approach; however, across all of them, there are some fundamental similarities:

1. In order to ensure that MCDC coverage can be proven based on trace data, the structure and the position of a decision within the source code must both be known.
2. At the same time, each condition in the source code must be represented at the object code level by a conditional jump or by a conditional instruction and not by an arithmetic representation of the condition.
3. When the MCDC coverage analysis is performed, the structure and the position of a decision within the object code must be known.

For the implementation of an MCDC coverage measurement without code instrumentation, according to requirement #1, additional information about the source code structure is required. This is currently not part of the debug information generated by the compiler. In addition to this, it must be ensured that the compiler generates the object code in such a way that it fits requirement #2.

AdaCore

MCDC coverage can be proven easily with a trace recording if the compiler and the software performing the MCDC coverage analysis are from the same



provider. The compiler can include the new required information on the structure and the position of each decision within the source code into the debug information. The company AdaCore (refer to [2]) offers such a solution. In addition to this, AdaCore also offers a target emulation solution for generating trace data.

For customers who have to additionally prove an MCDC coverage for the final implementation on the target hardware, AdaCore offers an interface which allows trace data recorded with TRACE32 to be imported for the required analysis.

t32cast

As of March 2018, Lauterbach customers can also prove an MCDC coverage in TRACE32 based on a real-time trace recording. Lauterbach offers the t32cast command line tool for this purpose, which analyzes the C/C++ source code. As a result, the information on the decision structure required for the execution of the MCDC coverage measurements is generated for each source code file. Consequently, the build process must be adjusted in order to allow this information to be generated (refer to "TRACE32 Extended Code Analysis" figure on the top of the page). The t32cast command line tool is compiler-independent and it can be easily integrated into existing build environments.

As soon as the user starts the MCDC coverage measurement in TRACE32, TRACE32 automatically loads all required .eca files, generated by the t32cast tool. Subsequently, TRACE32 is able to map the decisions on source code level to the object code with the help of the debug information.

However, during this process, the user must ensure that the selected compiler represents each condition in the source code by a conditional jump or by a conditional instruction at the object code level, e.g. by disabling optimizations.

Conclusion

The TRACE32 MCDC coverage can be used independently from the compiler and the processor architecture. For those who can't abandon code optimization, even when implementing safety-critical systems, there is no way around reducing the optimization for the trace-based MCDC coverage.

In the future, it would hold significant value, if compilers could be configured so that, at the object code level, decisions are translated into conditional jumps or conditional instructions only, independently from the optimization level selected, and that optimizations selected are performed entirely for all remaining areas.

References

[1] CAST-17 Position Paper (2003, January). Structural Coverage of Object Code

[2] Comar, C., Guitton, J., Hainque, O., & Quinot, T. (2012, May). Formalization and comparison of MCDC and object branch coverage criteria. In ERTS (Embedded Real Time Software and Systems Conference).

[3] RTCA Inc. (2011, December) RTCA/DO-178C Software Considerations in Airborne Systems and Equipment Certification