
# EMBEDDED COMPILERS

## FOR THE NEEDS OF TODAY'S AND TOMORROW'S AUTOMOTIVE INDUSTRY

# EMBEDDED COMPILERS FOR THE NEEDS OF TODAY'S AND TOMORROW'S AUTOMOTIVE INDUSTRY

Jan Schlemminger, Field Application Engineer

TASKING is dedicated to providing compilation tools of the highest performance and quality to embedded processing markets; specifically, in the scope of this white paper, to the automotive sector.

Central to the TASKING offering are high-end optimising compilers that are dedicated to producing code of the utmost safety and efficiency for a range of popular microprocessor and microcontroller architectures including (but not limited to) ARM Cortex, Infineon TriCore/AU RIX, Renesas RH850, and Power Architecture - most of which are in widespread use in automotive powertrain, body electronics and ADAS systems.
TASKING maintains close relationships with the respective silicon vendors, as an intimate knowledge of target architectures is an essential prerequisite to producing the most efficient and high-quality code.

TASKING has a substantial market share with automotive OEMs and Tier-one suppliers in every market worldwide. The company's evolution in this space has taken place over the same timescale in which the automotive industry has seen a dramatic increase in the number and complexity of processor-based systems per vehicle - and in the extent to which those systems have become safety- and mission-critical.

Over the last 30 years, approximately, the automotive sector's use of processors has progressed from 8-bit devices that were very often programmed without benefit of compilation techniques at all (assembler coding) to 32-bit machines with exacting real-time performance constraints.

Also over that timescale, a change in emphasis has taken place in the competitive landscape for embedded tools in the automotive sector. In an earlier market phase, differentiating factors were very much focused on performance metrics; on code size, on memory usage and on execution speed - the outcome of optimisations carried out during the code compilation process. To some extent, these aspects have been joined by system power usage: this may be less of a key issue in automotive designs compared to other market sectors, but is nevertheless important as processor-based systems proliferate in vehicles and finite electrical power resources are stretched.

But surpassing other aspects, today there is a much greater level of attention paid to correctness of output code, and to safety-related issues.

Over the past three decades, a number of trends have played out in parallel. In the automotive industry, there was a marked increase in quality levels (in general) in the 1980s; and in the 1990s, many previously-mechanical systems were replaced, or assisted, by electronic variants. Electronic fuel injection and anti-lock braking are two notable examples. Microcontroller-based electronic control units (ECUs) became the norm to control subsystems in a car, and the number of ECUs per vehicle multiplied. After the year 2000 complexity escalated rapidly; multi-core processors were required to fulfill the computational and safety requirements of powertrain applications, and there was a corresponding (and continuing) increase in complexity of the algorithms applied in chassis control and ADAS systems.

At the same time, in the wider embedded-code-creation context, awareness of the need for a structured and disciplined approach to software engineering was recognised, as the vulnerability of all types of safety-critical systems to software errors became apparent.

Over that same timescale, the automotive industry has had an equally steep curve to navigate, in terms of increasing product quality. On top of internally generated quality programmes, and competitive pressures, have come external quality standards, to which compliance must be demonstrated. A prime example is ISO 26262.

Quality in code compilation means, at its simplest, absence of bugs. Product quality is measured via exhaustive testing and comparing the results against the expected results and also the results obtained by toolsets of other suppliers. Compilers should generate the correct output in response to (any) legitimate input. Developing a bug-free product requires having mature processes in place, otherwise it is not possible to develop and maintain a product that is re presentative of the state of the art. Suppliers to the automotive industry typically implement their processes in conformance with ASPICE compliance level 2.

From 1985 to 2010 all compiler vendors applied the same test approach. In the early days, the International Standardisation Organisation (ISO) standardised the C language, and also sup plied a test suite to verify the correctness of a compiler implementation. This was a large suite of small test programs that cover all features defined in the ISO-C standard. Part of the specification of the C language is also a list of (nearly 200) “undefined” behaviours, and a smaller but still substantial set of “unspecified” behaviours. The integrated MISRA C and CERT C code checkers can be used to verify possible occurrences of most of those behaviours.

The entire (-compiler community - open-source and commercial alike - had a chastening experience with the release, in 2011 of the open-source tool Csmith (University of Utah) together with the results of many compiler tests the University group ran with the tool.

Csmith is a “randomised test-case generator… using differential testing”. It was able to cover a far wider range of “corner cases” of inputs to compilation, and it used different compilers to check each other’s outputs. In short, it inspired a step-change in testing methodology and (in some cases) product quality, almost overnight.

One of the surprising findings that emerged from that episode was that certain compilers showed erroneous behaviour irrespective of the level of optimisation effort requested by the programmer. Nevertheless, verifying correct behaviour through the steps of optimisatopn, especially intensive optimisation, is a critical aspect of ensuring bug-free compiler operation. Optimisations have the power to alter how the code executes a function; it is essential to ensure that what the code does is unchanged, and is so in response to all possible inputs.

TASKING continues to deliver the highest levels of product quality in its tools; however, it must be recognised that testing combined with process improvements - when applied to extremes - cannot absolutely guarantee that a product is perfect. At the present time, TASKING is experimenting with formal verification. That is, deriving a comprehensive, formally correct and rigorous mathematical description of a code function, both before and after a process such as optimisation - and confirming that the two are equivalent. TASKING is now at a stage where the company has a prototype where it can be guaranteed, with a machine-checked mathematical proof, that the register allocation phase of its compiler is correct.

## ISO 26262

The Standard ISO 26262 “Road vehicles - Functional Safety” aims to ensure the safety of electrical and electronic systems in road vehicles. In common with other ISO safety-related standards, it does not aspire to the creation of systems that cannot or will not fail. Rather, it explicity recognises that failures will occur, and seeks to ensure that the response of the overall system to a failure will in all circumstances result in a safe outcome. It is therefore concerned with ensuring that the design process employed is such as to result in a safe system.

| QUALIFICATION METHODS | ASIL A + TCL 2/3 | ASIL B + TCL 2/3 | ASIL C TCL 2 | ASIL C TCL 3 | ASIL D + TCL 2/3 |
|---|---|---|---|---|---|
| a Increased confidence from use | ++ | ++ | ++ | + | + |
| b Evaluation of the tool development process | ++ | ++ | ++ | + | + |
| c Validation of the software tool | + | + | + | ++ | ++ |
| d Development in accordance with a safety standard | + | + | + | ++ | ++ |

That standard describes the qualification methods for each sub-part:
highly recommended (++) and recommended (+)

A section within ISO 26262 categorises the likely consequences of a sub-system failure in terms of ASIL (Automotive Safety Integrity Level) from A (least serious) through to D (immediate threat to life). The design processes that build the code for each of the systems listed in below table must be structured to demonstrate compliance with the appropriate ASIL level(s).

The key word in the preceding paragraphs is “process”. The standard governs the process of design of the automotive product. As an input to that process, a (for example) compiler must be documented to show that it has, itself, been designed following a process that takes account of possible sources of failure, and mitigates - or, more appropriately in the case of a software tool, detects - any such failure. It must also be supported in such a way that the automotive design team can in turn show that their process follows the guidelines of ISO26262; the manufacturer/design house must establish the required level of confidence in the soft ware tools it proposes to use.

Alongside its suite of software tools, TASKING offers a structured programme to assist an automotive system designer with documenting the processes to ISO 26262, in the form of Tool Validation Services, a Compiler Qualification Kit and ISO 26262 Compiler Qualification Services.

TASKING software development tools (as is the case for all commercially available compilers, and as already indicated) are not themselves developed according to a safety standard. Instead, TASKING is adjusting its processes in accordance to Automotive SPICE, with the target of ASPICE level 2. Implementing processes at A-SPICE level 2 is not required for ISO 26262 soft ware tool validation, but saves future efforts of executing a complete tool validation for every product update.

In this context it may be noted that TASKING builds compilers for each of its targeted core architectures on a common core technology (called Viper ,,VX”). This yields a minimum-change route and strategy to produce an optimised (and optimising) toolset for each target, which likewise minimises the task of documentation and standards-oriented support.

For example, TASKING has recently extended C compiler support for the Renesas RH850 device family; v2.2r1 of the TASKING C compiler suite for the RH850 architecture sup ports later RH850 microcontroller variants, adds code optimisation improvements, and uses an integrated on-chip debugger. TASKING based it on VX compiler technology, with industry-leading code optimisation techniques and proven-in-use track records in automotive applications such as powertrain, body control, chassis control, including safety critical applications.

New code generation optimisations have been added to this new compiler release, such as MIL linking and code compaction (reverse inlining), techniques that have proven to be highly effective in TASKING's compilers for automotive microcontrollers. Measurements on a wide range of different application sources, including powertrain applications, show improvements up to 17% in code compactness. Along with efficient code generation, allowing for fast and compact applications, standards issues are addressed with integrated code analysers for MISRA-C:1998, C:2004 and C:2012 guidelines and the CERT C secure coding standard; and the release is backed by TASKING's ISO 26262 Support Program. The RH850 microcontroller family offers rich functional safety and embedded security features needed for new and advanced automotive applications. TASKING supports these features and application requirements th rough its ISO 26262 Support Program.

Also recently added is a new C compiler for an IP (intellectual property) function, the Generic Timer Module (GTM-IP MCS) from Robert Bosch GmbH. The GTM IP module forms a generic timer platform for complex applications in the automotive industry such as powertrain, power steering, chassis and transmission control. The GTM provides a wide range of timer functions for instance counters, multi-action capture/compare, PWM functions, duty-cycle measure ment and many more. The GTM also features internal RISC-like programmable cores for data processing and complex output sequence generation. The IP is designed to run with minimal CPU interaction and to unload the CPU from handling interrupt service requests as much as possible.

Generic interfaces and the hierarchical system architecture make the GTM an ideal solution as an IP core to sit alongside a variety of microcontroller architectures, such as the AURIX (Infineon Technologies), RH850 (Renesas) and Power Architecture (Freescale and STMicroelectronics).

While first generation GTM silicon is currently available through semiconductor vendors, Bosch has already developed the third generation that will deliver significant functional enhancements. These improvements not only enabled the development of the TASKING C compiler, but will also allow for the GTM's features set to be better exploited with C language programming. Bosch has supported TASKING on the development of a dedicated C compiler, based on the Viper "VX" compiler technology.

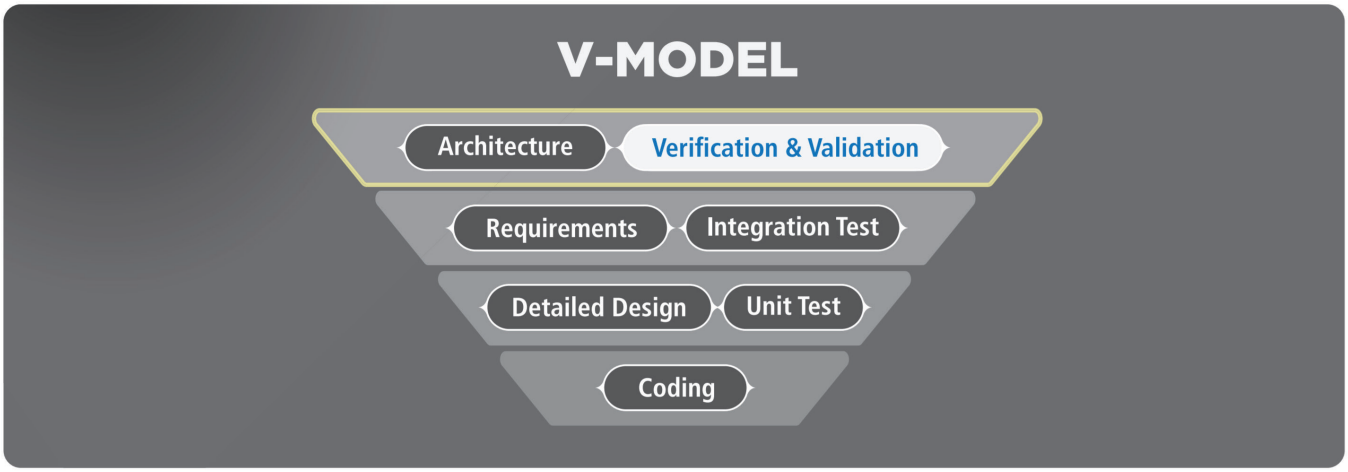
In contrast to other general compiler technologies, TASKING's Viper compiler technology is well suited for the development of compilers for specialised cores like the GTM, as it was developed from the ground up for embedded systems with memory constraints and performan ce challenges. TASKING's ISO 26262 Support Program will also cover the new GTM compiler via the dedicated toolset offerings for the various main microcontrollers, enabling customers to achieve certification for functional safety standards such as ISO 26262 and others.

## TOOL VALIDATION TO ISO 26262-811.4.9

ISO 26262 provides hardly any guidance about tool validation. Section 11.4.9.2 contains all guidance and is limited to three sentences. TASKING's automotive OEM and tier-one customers approach ISO 26262 tool validation in various ways.

In an attempt to unify the different tool validation approaches of its customers TASKING uses, as a framework, the guidance provided by the standard "DO-330 - Software Tool Qualification Considerations", which is intended to provide tool qualification guidance to the avionics domain, but can also be applied in other domains such as automotive (DO-330 1.2). Guidance for commercial-off-the-shelf (COTS) software tool qualification is provided (DO-330 11.3) where the qualification activities are partly performed by the tool developer and partly by the tool user.

The basis for compiler validation to match ISO 26262-8 11.4.9 addresses the top of the V-model only (Figure below). That is, the requirements phase, and the verification/validation steps. The validation measures must demonstrate that the software tool complies with its specified requirements. The tool requirements specifications under development by TASKING cover all features listed in the user manuals, the requirements expressed in tool related standards such as in, for example, "ISO/IEC 9899 - Programming languages - C", and requirements related to the compiler design and implementation. The verification and validation efforts executed by TASKING cover all requirements which fulfill the ISO 26262-8 11.4.9.2.a guidance that the requirements coverage metric shall be 100%. Although ISO 26262 does not provide guidance regarding code coverage, TASKING strives to reach near 100% decision coverage. Additionally, the latest academic contributions that advanced the state of the art in compiler testing are applied to uncover errors in highly complex code optimisation algorithms by maximising the path coverage.




Validation of the Software Tool

TASKING®

Barthstrasse 4,
80339 Munich
Germany

Phone: +49 (0)89 6933 4500
Technical Support: support@tasking.com