TASKING®

VX-TOOLSET FOR TRICORE™

Product Overview

A COMPLETE DEVELOPMENT SOLUTION

The TASKING® VX-toolset for TriCore™ (Release v6.3) is a complete solution for code development for AURIX™ multi-core hardware from Infineon®. The VX-toolset produces fast and compact code, not just for the TriCore, but the other cores within AURIX™, including the GTM, SCR, HSM, and SCR. This best-in-class compiler performance coupled with a decades-long industry track record at both large and small enterprises, plus ASPICE Level 2 certification provide the assurance you need for your development projects.

Here is a summary of TASKING VX-toolset for TriCore features. More detailed discussions of these features follow this introduction.

- Great performance for both manually written and model based/auto generated code
- Multi-core optimization with universal linker support for all cores plus language extensions
- Profiling and debugging (both hardware and software)
- Safety - Safety Kit - MISRA - ASPICE Level 2 certified
- Premium support for current and old versions available
- Easy to get started and begin benchmarking with pin configurator
- Very broad support for MCAL and industry standard RTOSs
- Compatible with relevant third party timing analysis tools

EASY TO DEPLOY, EASY TO USE

The TASKING VX-toolset uses the industry's favorite Eclipse™ integrated development environment (IDE) that allows users to instantly begin working in a familiar environment. Eclipse ties all the tools together and makes them full power accessible with ease. Eclipse also makes it easy to integrate the VX-toolset into custom-built systems. For customers who already use TASKING compilers, cross-linking allows you safely combine different TASKING compiler and linker versions in your project. For example, You want to compile your application with the latest TASKING compiler version but you are forced to use an older version because MCAL is not tested with this version.You could use crosslinking to link the MCAL and Application code which are compiled with different TASKING compiler versions and reduce software requalification costs. Whitepapers, webinars, and forums provide further education and learning opportunities to ensure effective and efficient use of TASKING tools.

PICK THE BUNDLE TO MATCH YOUR NEEDS

You can choose the best fit for your application development activities from four bundles: Standard, Professional, Premium and Enterprise Editions. Along with the standard Eclipse, C/C++ compiler toolset and simulator modules, additional functionality includes options such as C compilers for the HSM, GTM, SCR and PCP cores, on-hardware debugging through an OCDS solution and a USB-to-JTAG wiggler.

MULTI-CORE SUPPORT

The VX-toolset provides two advanced methods for multi-core program development for the AURIX™ microcontroller, which includes:

- **Compile time core association:** Crosscore access is detected by the compiler in an early stage. This allows for the best control of code and data destination on the various cores. Stricter coding discipline is asked from the developer, resulting in better quality control while delivering optimal program execution.
- **Link time core association:** The destination core is selected in the final step, delivering ultimate programming flexibility and allowing for straightforward reusability of existing code.

For ultimate flexibility and program execution, both methods can be used together.

The linker also supports references between TriCore core memories and GTM/MCS memories, and between TriCore core memories and SCR memory. The GTM/MCS memories (RAM) and the SCR memory (RAM) can be initialized automatically by the TriCore start-up code from the TASKING run-time library.

CROSS-LINKING

Cross-linking allows linking of object code built with an earlier toolset release into a project that is being developed with a more recent toolset release. With releases v6.2r2 and v6.3r1 of the toolset, TASKING guarantees compatibility of code developed with versions v4.2r2 and v5.0r2 of the product, under specific conditions. Through this guarantee the user can re-use application code or use third party code developed and validated with an older compiler release. This will give the user more flexibility and more products to choose from, like MCAL libraries, Real-Time Operating Systems and communication stacks from parties like Infineon Technologies, ETAS, Elektrobit, and Vector.

C COMPILERS

Using the latest compiler technologies from TASKING, all VX-toolset C compilers are reliable, compliant, best-in-class, complete, compatible and easy to use to generate the most optimal code. The TASKING VX-compilers are tested for ISO C99/C11 and ISO C++ conformity against authoritative validation suites, such as Perennial® and Plum Hall®. In addition, the optimization techniques of the compilers are tested with large real-world applications (for example, powertrain and body control sources), as well as industry benchmark standards such as Nullstone and EEMBC.

Compiler Checklist		
	TASKING	Competition
TriCore C compiler	✓	✓
GTM C compiler	✓	Some
HSM C compiler	✓	Some
SCR C compiler	✓	✗
Eclipse-based debugger	✓	✗
AURIX™ Pin Mapper and Conflicts Solver	✓	✗
Best-in-class for code size and speed	✓	✗

TRICORE C++ COMPILER

The VX-toolset Includes a C++ compiler for TriCore core programming. The compiler supports all language extensions up to the C++11 standard. Full C++11 libraries are provided as well. With Viper compiler technology, the TASKING VX-toolset for TriCore generates code with the smallest footprint and fastest execution. Depending on the specific requirements of your TriCore application, optimizations can be further tweaked for smaller code sizes or higher execution speeds.

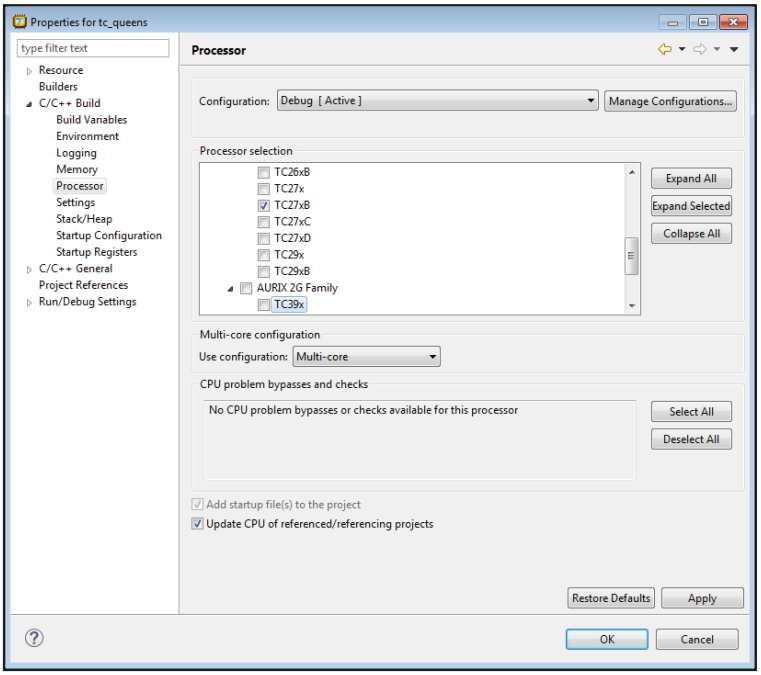

Figure 1: The compiler is easy to set up and use for the entire TriCore family.

Compiler optimizations include:

- Partial Redundancy Elimination (PRE) detects and eliminates repeating (sub-) expressions.
- Various Loop and Jump optimizations speed up execution and reduce code size.
- Control-flow and code-reduction optimizations remove dead code and perform transformations to minimize jumps.
- Function inlining replaces calls to small functions with inlined copies of the function code.
- Peephole optimizations replace instruction sequences with equivalent but faster and/or shorter sequences, or remove obsolete instructions.
- Inter-procedural register allocation
- Application wide code compaction (also called reverse inlining)
- Application wide speed optimizations by “MIL linking”

Syntax and Semantic Checks

The compiler offers a vast array of syntax and semantic checks that warn about potential undesirable effects or bugs in your program. Early fixing of source code problems when reported by the compiler generally only takes minutes compared to hours, or days, when the problem is discovered at run time.

Examples of compile-time checks include:

- Validating printf and scanf format strings against the type of the actual arguments
- Detection of reads from uninitialized memory locations
- Detecting unused variables
- “Value tracking”, which is used to detect errors such as:
  - Array subscript out of bounds
  - Constant conditions

Runtime Error Checking

The runtime error checking capabilities in the compiler can reveal runtime errors when they first occur. The kind of errors found by runtime error checking are typically hard to find since they manifest themselves through secondary effects or, in the worst case, will not manifest at all prior to your product being shipped. By identifying the source line where the error first occurs, the runtime error checking facilities reduce the time spent in the debugger and increase the quality of your software. You can specify whether the application will terminate or continue when an error is detected.

These optional checks are implemented by generating additional code and/or enabling additional code in the standard C library. Runtime error checking has a nominal effect on code size and execution speed and can be enabled on a module-by-module basis, making it practical for use in debugging large applications.

The following types of checks are provided:

- Bounds checking verify all pointer operations to detect buffer overflows and other illegal operations including:
  - Accessing uninitialized or null pointers
  - Accessing objects outside their declared bounds
  - Illegal pointer arithmetic
- Malloc / free checks uncover dynamic memory allocation errors including:
  - Write to freed memory
  - Multiple calls to free
  - Passing an invalid pointer to free
- Report an unhandled case value in a switch without a default part.

CPU Functional Problem Support

Semiconductor vendors regularly publish microcontroller errata sheets reporting deviations from the electrical and timing specifications. As an integral part of best practice architecture support, the TASKING VX-toolset for TriCore provides bypasses and checks for identified silicon defects. CPU functional problem support is provided throughout the complete toolset, including:

- C-compiler bypasses adapt code generation in order to avoid the identified erratic instruction sequences.
- Assembler checks warn the assembly programmer for suspicious or erroneous instruction sequences.
- Protected C-library sets are built with bypasses for all identified CPU functional problems.

If reliability of your embedded application is essential, be sure to put support for CPU functional problems on your list of compiler selection criteria. Through its close partnership with semiconductor vendors, TASKING offers the most comprehensive support for this with its compilers.

Static Code Analysis

Static code analysis is a method used to verify all possible paths within a software program without actually executing the program. A static code analysis tool can efficiently locate defects including out of bound array access, memory allocation errors, arithmetic over and underflows, and inconsistent code fragments that go unnoticed during dynamic tests or peer reviews. Static code analysis can be applied early in the software development process, and can be applied on incomplete and incorrect code bases and when no test-cases need to be developed.

TASKING has integrated static code analysis functionality for CERT C and MISRA C in its C compilers, with the advantage that such an analyzer is aware about specific embedded software issues such as: The existence of special function registers, the use of inline assembly language C-language extensions such as pointer and memory space qualifiers to address multiple address spaces DSP specific data types such as circular buffers, and fixed point data types.

Intrinsics

Certain assembly instructions have no equivalence in C. Intrinsic functions let you benefit from inclusion of those assembly instructions in your application. Intrinsic functions are predefined functions for which the compiler generates highly efficient assembly code. The compiler supports a wealth of intrinsic functions, including dedicated functions for AURIX™ TC3xx family, enabling you to get the maximum efficiency out of the MCU.

The compiler always inlines the corresponding assembly instructions in the assembly source. Although it is possible to inline assembly code by hand, intrinsic functions use registers more efficiently and their use prevents your C source code from becoming less readable.

C Compiler for GTM

The TASKING TriCore toolset includes a fully integrated C compiler that supports the third generation Generic Timer Module core from Bosch® which is present on AURIX™ TC3xx family. Programming a complex core like the GTM in C makes you more productive both in the development phase as well as the maintenance stage. In addition to an optimized programming approach with the TASKING TriCore compiler, the GTM C compiler also supports the “C array” output format, enabling interoperability with third party toolsets. If you develop an AURIX™ TC3xx application with the C compilers for the TriCore and GTM from the VX-toolset for TriCore, you can re-use your GTM application code in an RH850 or Power Architecture® based project with a C compiler for such MCU from another vendor. (The GTM compiler is also available as a separate TASKING product.)

The first GTM core generation present on the AURIX™ variants is fully supported by means of the included MCS/GTM assembler.

C Compiler for HSM

For programming the Hardware Security Module of the AURIX™ microcontrollers, the TASKING VX-toolset includes a fully integrated and dedicated C/C++ compiler. This compiler is based on the standard C compiler for the Cortex-M series from TASKING, and it can easily be accessed from the TriCore tool set’s IDE.

C Compiler for SCR

All AURIX™ TC3xx devices as well as select AURIX™ TC2xx derivatives, such as the TC26x series, have a dedicated 8-bit Standby Controller (SCR) on board. This controller is based on the Infi neon XC800 microcontroller and TASKING has developed a new and highly optimizing C compiler to program this core with its limited memory space. This exclusive compiler based on VX technology from TASKING, generates more efficient code than traditional XC800 compilers and is therefore the ultimate programming tool to deal with the limited resources of the Standby Controller. It is fully integrated in the VX-toolset for TriCore and does not require a third-party compiler solution.

C Compiler for PCP

TASKING offers a unique C compiler for the TriCore Peripheral Control Processor (PCP). Despite the limited functionality and restricted instruction set of the PCP, we have been able to develop a fully functional C compiler. The C compiler delivers code at an unexpectedly high performance level and provides several special extensions for PCP programming.

CERT C

The CERT C/C++ secure coding standard is defined by the Computer Emergency Readiness Team (CERT), founded by the US government. TASKING® is one of the first vendors to provide a CERT C coding guidelines analyzer built into a C compiler for embedded software development.

MISRA C

MISRA C is driven by the Motor Industry Software Reliability Association and guides programmers in writing more robust C-code by defining selectable C-usage restriction rules. Through a system of strict error checking, the use of error-prone C-constructs can be prevented. The TASKING C compiler offers the industry's first support for MISRA-C:1998, MISRA-C:2004 and the latest MISRA-C:2012 guidelines, including Amendment 1: Additional security guidelines for MISRA C:2012, dated April 2016.

INTEGRATED DEBUGGER

The integrated debugger has been redesigned from the ground up and is ready for trends like integrated kernel awareness and multi-core debugging. Utilizing the Eclipse IDE workbench, it comes as a plug-in with a seamless integration to the editing environment. With the VX-toolset for TriCore it provides two execution environments serving various debugging needs.

Debugging Via Infineon DAS Support

Making the most of the On-Chip-DebugSupport (OCDS) facilities built into the Infi neon TriCore microcontrollers, our debugger offers an accessible, high quality in-circuit-emulation functionality. The VX-toolset has been tested and qualified with the Infi neon Debug Access Server (DAS) solution. The DAS environment is the universal emulation access software architecture for all Infi neon microcontroller families. Extensive support for DAS is guaranteed by Infi neon and, as a result, TASKING has adopted this debug standard. Through DAS, the TASKING TriCore debugger is compatible with Infi neon TriBoard Starter Kits and Application Kits with an on-board wiggler through USB cable. This debugger is also compatible with the Infineon DAS miniWiggler debug probe, enabling a very cost-effective debug solution for on-hardware testing for custom hardware or other evaluation boards.

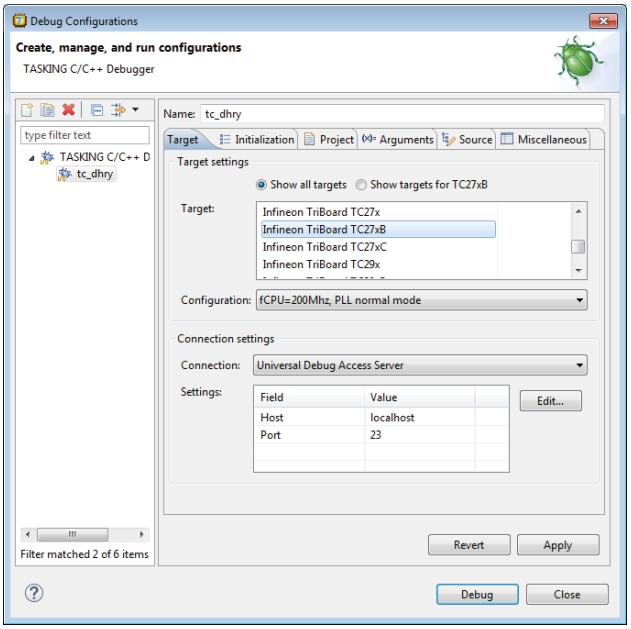

Figure 2: Configuring the debugger is simple, with most selections menu based.

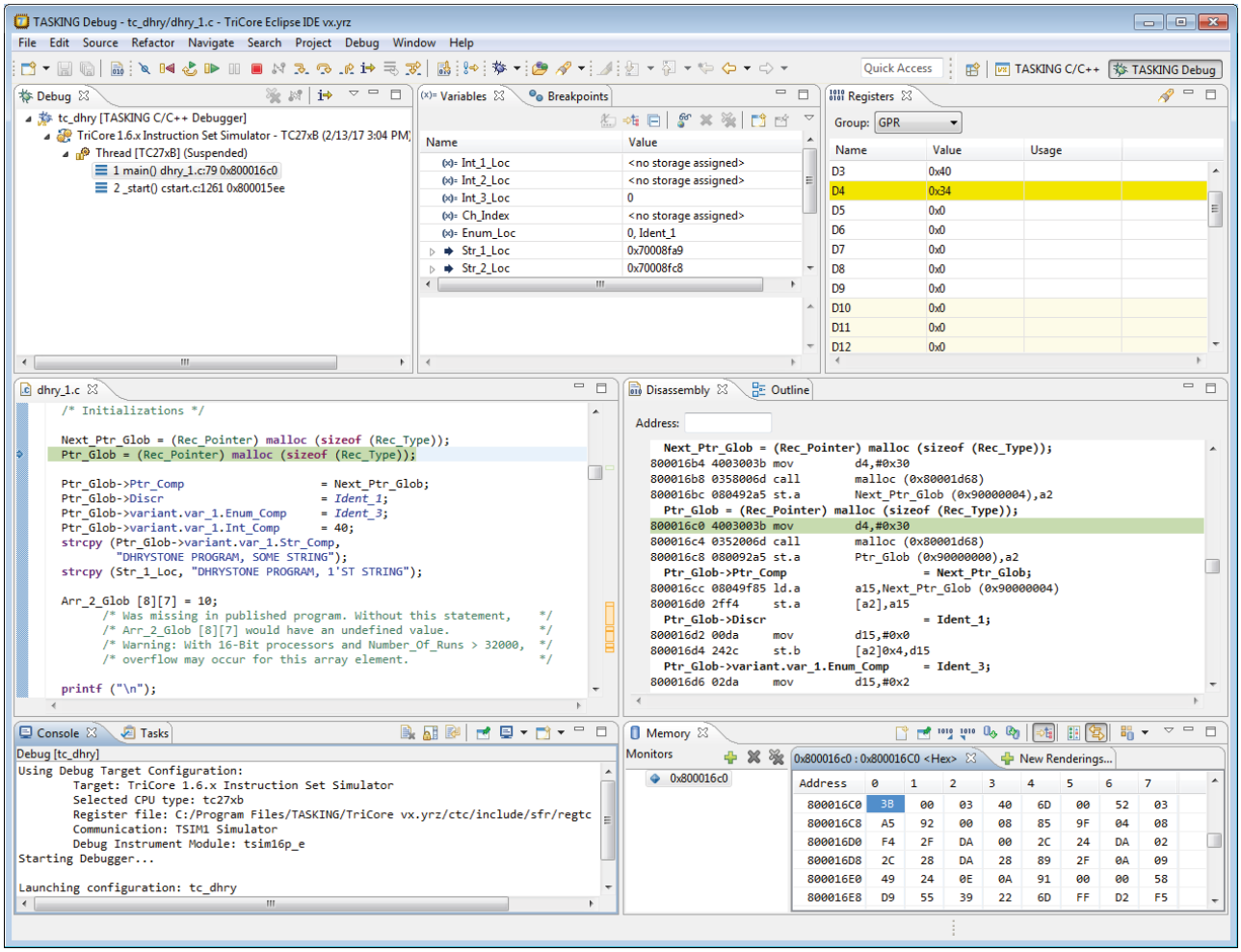

Figure 3: The debugger offers an easy-to-use environment.

TriCore™ Instruction Set Simulator Debugging

The TriCore simulator debugger features instruction set simulation, allowing you to extensively debug and regression test your application on your PC, even before your target hardware is available. A plug-in for instruction set simulation of the PCP is also included.

Code Profiling

In addition to the profiling features built into the debugger, the compiler also has a profiler that uses code instru–mentation. Code profiling can be used to determine which pieces of your code execute slower than expected and which functions contribute to the overall execution time of a program. A profile can also tell you which functions are called more or less often than expected. The advantage of this code profiling option in the compiler is that it can give a complete call graph of the application annotated with the time spent in each function and basic block.

AURIX™ CONFIGURATION TOOLS

Advanced microcontrollers are equipped with a large number of on-chip peripheral modules, but the limited number of pins on the chip usually does not allow all modules to be used simultaneously. The TASKING Pin Mapper functionality removes the developer's complex challenge of configuring chip hardware registers that are used for assigning the peripheral module signals to the physical pins.

The Pin Mapper provides an interactive visual representation of the pin layout within the toolset IDE, through which the developer can configure and review properties of the pins. The Pin Mapper reports errors or warnings for possible connection conflicts, saving the developer from the tedious task of maintaining an overview of the pin assignments in spreadsheets. You can solve pin conflicts by hand by making other connections, but when there are many conflicts or in situations where most port pins are in use, it can be quite cumbersome and complex to solve conflicts. The Pin Conflicts Solver can automate this process and will solve most conflicts for you. Through the graphical editor and code generator from the TASKING Software Platform, you can quickly configure (Infi neon iLLD) low-level drivers, various C files and header files, as well as the RTOS for use in final product applications. This significantly simplifies the required steps to program an advanced microcontroller.

ECLIPSE IDE

The IDE, built on the Eclipse framework, provides a seamless workbench for the complete tool chain including the debugger of the VX-toolset. The IDE provides facilities for project configuration and management, C/C++ and assembly code-aware editing, build management, debugging, profiling and more. It provides functionality to help you set up your embedded TriCore project and configure your target board settings to debug your project on hardware.

The Eclipse editor supports C, C++, assembly language and header files with syntax highlighting, auto completion, context assistance and tool tips. As you would expect from a de facto standard IDE, it provides full support for all relevant source code version control systems. The Eclipse environment provides a single platform for many diff erent embedded-product toolsets from diff erent vendors. The standardization on an industry-wide IDE significantly reduces your learning curve, removes the barriers of changing development tools for diff erent architectures, increases your productivity, and ultimately reduces the time to market with your end product. The availability of plug-in modules to enhance or extend the feature set of the Eclipse IDE ensures that you can build the workbench according to your development needs. With the concept of the open Eclipse framework, third party tool vendors can now develop plugins that tightly integrate into various IDEs from diff erent vendors, unlike proprietary IDEs where custom connectivity needs to be created.

The IDE in the VX toolset is based on the Mars release of Eclipse and the C/C++ Development Tools (CDT). TASKING has built the integration blocks for the toolset and extensions to Eclipse to make the whole environment a coherent workbench. Plus, with the Eclipse IDE, it is easy to integrate into third party, custom built systems.

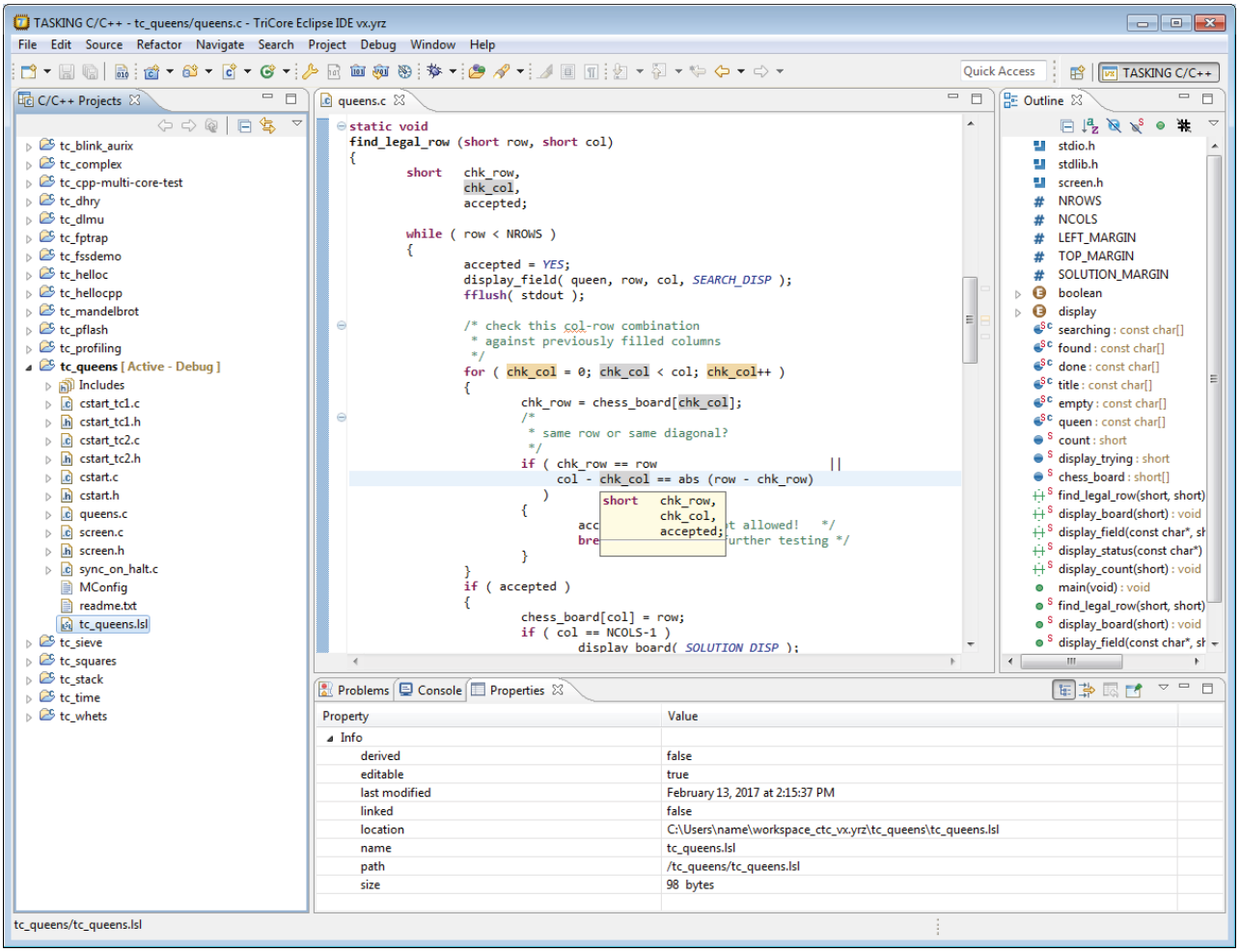

Figure 4: The TASKING VX toolset uses the de facto standard Eclipse IDE, making an easy learning curve.

AUTOMOTIVE SPICE CAPABILITY LEVEL 2 (ASPICE CL2)

The TASKING VX-toolset for TriCore is developed using an ASPICE CL2 process. Automotive SPICE is a framework for designing and assessing software development processes. Effective implementations lead to better processes and better product quality. It was developed by the consensus of several major car manufacturers. Automotive SPICE has become a standard in the international automotive industry.

WHICH BUNDLE IS THE BEST FOR YOU?

The TASKING VX-toolset from TASKING® is available in targeted bundles – Standard, Professional, Premium and Enterprise Editions, allowing you to choose the best fit for your application development activities.

The Enterprise Edition is the best choice if you want to have all essential development tools around the C/C++ compiler integrated into one environment. In addition to a software simulator, it offers an on-hardware debugging solution – using anOCDS solution with a USB-to-JTAG wiggler. The OCDS debugger is the most cost-effective on hardware debug solution you can get. The truly unique parts of the Enterprise Edition are the C compilers for the HW Security Module (HSM), the GTM, the SCR and the PCP, as well as the Code Safety Checker for ASIL. If you plan to write your code for the additional cores in C language, this Enterprise Edition is the best and only option on the market.

The Premium Edition is nearly as comprehensive, with the exception of the GTM compiler and the integrated Code Safety Checker.

Target architecture support

The TASKING VX-toolset supports all TriCore™ derivatives. From within the Eclipse IDE you can easily select the TriCore™ device of your choice for your project:

AURIX™ TC3xx family: TC39x, TC38x, TC37x, TC36x, TC35x, TC33x; AURIX™ TC2xx family: TC21x, TC22x, TC23x, TC23x_ADAS, TC26x, TC27x, TC29x; TriCore™ devices: TC1130, TC1166, TC1167, TC1184, TC1197, TC1337, TC1367, TC1387, TC1736, TC1746 [TC1782bd], TC1724, TC1728, TC1748 [TC1798bd], TC1762, TC1764, TC1766, TC1767, TC1768 [TC1387bd], TC1782, TC1791, TC1792, TC1793, TC1796, TC1798

TASKING active relationship with Infineon® Technologies enables us to support new derivatives already in the toolset prior to their availability in volume.

The Professional Edition provides many features of the Premium Edition, including the OCDS debugger and HSM C compiler, but without the SCR and PCP C compilers. If you develop your application based on an AURIX™ with HSM co-processor, or a TriCore derivative without the PCP, this Professional Edition is an attractive solution to consider. Also, if you are fine with programming your TriCore with PCP in assembly code instead of C, this package is a good choice. If you purchase this bundle, be sure to add the optional USB-to-JTAG miniWiggler in case your hardware board comes without an on-board wiggler.

The Standard Edition is the perfect bundle for C/C++ programming and debugging with a simulator. The MCS/GTM unit of the AURIX™ can be programmed through the included assembler, whereas the linker is fully prepared for multi-core TriCore development. You can upgrade to the one of the other Editions at a later time, offering you all included functionality under a unified interface.

TASKING®

www.tasking.com