DON’T GET STUCK WITH THE WRONG SET OF DEVELOPMENT TOOLS FOR TRICORE APPLICATIONS

DON'T GET STUCK WITH THE WRONG SET OF DEVELOPMENT TOOLS FOR TRICORE APPLICATIONS

INTRODUCTION

Choosing the right set of embedded software development tools for automotive applications can take quite and effort. Choosing the wrong set of tools can create problems for every project. This paper explores criteria for selecting the right tools as well as what to watch for so you don't get the wrong tools. For this discussion, the popular Infineon AURIX™ TriCore™ hardware is assumed.

Figure 1: Top-quality development tools produce the most efficient results.

But first, what exactly constitutes a "set" of tools for embedded development? Of course, the compiler is the foundation. In addition, you will need a debugger to troubleshoot code, and a linker. Many toolsets include a library with common and very useful functions that can speed both code development and execution. In addition, there are also tools offered that analyze the code for safety, and that optimize the code for compactness and speed. Let's look into these tools in more detail.

COMPILERS ARE CRITICAL

Without the compiler, well any other tool is useless. That's why it's imperative to make an informed decision when first selecting a toolset as well as changing from one to another. There are many compilers available and they are certainly not equal.

What about Open Source?

Compilers based on open source may be attractive as they are often among the less expensive. However, an open-source compiler could introduce headaches to your development process and also may not be able to take advantage of the full performance and safety features of the TriCore hardware. There are three major issues that can affect open-source compilers:

Bugs may take time or may never get fixed: The most prevalent issue with open-source compilers is that the code is not controlled by the vendor selling the compiler. The vendor has control only over the code they developed on top of the open source. Bugs within the open source can be difficult and time consuming to get fixed. In the automotive industry, where safety is critical, choosing an open-source compiler may not be the best way to go.

Open-source compilers may not take advantage of the hardware: The TriCore has many hardware features that allow maximum performance as well as handle safety issues. If the compiler is unaware of the existence of those features and how to exploit them, the processor can be handicapped. For example, most open source compilers lack support for the generic timing module (GTM) in the TriCore. Optimizing code for the multi-core architecture is not available with many open-source solutions either.

Proprietary compilers produce better code: Open source programs are by their nature more generic than proprietary designs, built with one focus. They generally make much more efficient use of memory and have optimization built in for the TriCore processors. The result is faster, more compact code than that produced by open source. That could mean the difference between an application running properly or even fitting in available code space. Plus, some proprietary compilers offer code-acceleration by having specific assembly instructions to allow you to include functions that C/C++ doesn't offer.

COMPILERS AND SAFETY

Safety is the number one concern for automotive electronics. There is currently an avalanche of automation taking place. Already available or several safe driving functions such as following distance, emergency braking, and lane departure. Over-the-air updates will soon replace trips to the dealer's service center for many problems, upgrades and even recalls. And, the autonomous driving car is closer than it appears. This means that the product's hardware and software must meet safety standards. A buggy compiler just won't cut it in this environment.

As mentioned earlier, most hardware offers safety functions onboard the chip. But, if your compiler doesn't support those functions, you are at best stuck writing assembly code; at worst, you may not be able to make the hardware function properly.

While compilers do not have to meet any safety standards, some manufacturers go through the trouble to become ASPICE Level 2 certified.

Automotive SPICE® (ASPICE) provides a standard process for designing and assessing automotive software development. Using these processes leads to better product quality. Choosing a compiler that is ASPICE-certified at Level 2 means you know that compiler has been developed per proven processes that enable you to meet the required safety standards. An ASPICE-certified compiler produces better code and can save money. Correcting bugs and error early in the process costs much less than those found way down the development stream. ASPICE certified development tools help you find and correct errors earlier, often significantly reducing development cost.

The compiler is the root of embedded software development. Be certain the compiler in your toolset has the capabilities you need for the development you do. Skimping on the compiler can make the entire development project costlier. Insist on ASPICE certification as well for the peace of mind and potential cost savings.

THE LINKER: THE OFT-IGNORED KEY TO PERFORMANCE

The linker is an essential part of the compilation process and is the key to extracting peak performance from the TriCore hardware. The linker combines all the object files from the compiler into a single executable. It also allows mapping where to programs, libraries, and data are located in memory as well as mapping what software runs on which core.

A linker that is aware of the TriCore architecture lets you take advantage of all features and allows performance enhancement. Proper use of memory can significantly accelerate code executing. By controlling how near and far memory is utilized, a great deal of cycle time can be saved.

Without a quality, architecture-aware linker it is difficult to get to top performance out of the TriCore hardware. Don't ignore the powerful capabilities of the linker when investigating embedded software development tool suites.

Figure 2: The linker is key to extracting maximum performance from the TriCore architecture.

DEBUGGING

Debugging is essential to embedded software development. No one writes perfect code — at least not every time. The ability to access a debugger when needed and quickly working out bugs makes for an efficient, cost effective development environment. Check the code, fix any bugs, and get back to development.

Unfortunately, that isn't how it works at many companies. Debuggers are generally expensive, especially the hardware probes. Consequently, many companies buy as few debugger licenses as possible to keep costs in line.

What often happens, especially at crunch time, is that many developers need the debugger at the same time, but there just aren't enough licenses to go around. So, they must queue up until a license becomes available. The developer spends down time instead of being productive. Even if the developer begins working on other code, it takes time to move into a different frame of mind and then back to the original code once a license frees up. Plus, many coders develop momentum as they are coding and having to wait on a license can derail that momentum. It all adds up to lost cost and extended development for the project.

A new trend in debugging is a streamlined, standalone debugger. A great deal of the cost of a high-end debugger is associated with timing analysis. However, when a developer simply wants to check code functionality as it is written, these functions are not necessary.

The cost of the streamlined debugger is also streamlined, meaning an enterprise could buy multiple streamlined debuggers for the cost of one high-end debugger. The result is that there are fewer — if any — queues to get a license and development can march on at the most efficient pace. Mix high-end and streamlined debuggers for maximum development efficiency.

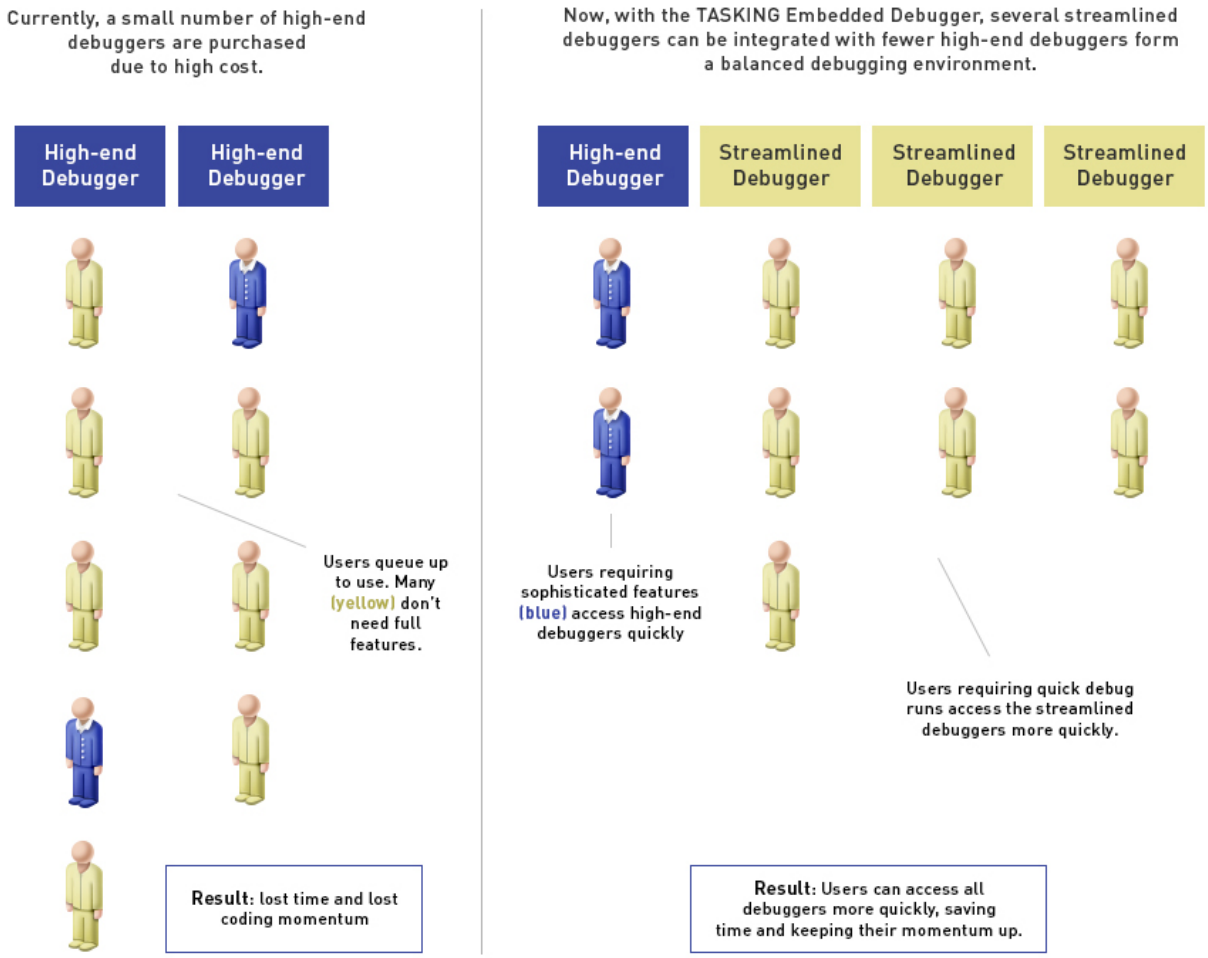

Figure 3: Instead of buying a small number of high-end debuggers, use the same funding to mix high-end and streamlined debuggers for maximum development efficiency.

INTEGRATED VS. FRAGMENTED TOOLSETS

It is certainly possible to buy a compiler from one vendor, a debugger and associated accessories from another, and a library of functions from yet another. Some companies do exactly this. The problem that arises is that when the developer changes from one tool to another, the GUI changes, the way to product works is different, and it may be an effort to get data from one to another.

Whereas, an integrated toolset from a single vendor presents a unified and familiar user interface for each tool within the suite. Data integrity is maintained because the data never "leaves". Integrated tool suites often offer additional tools, such as a code safety checker, that further extend the value. And, from a safety standpoint you want the entire suite to be ASPICE certified.

DON'T RISK SELECTING THE WRONG SET OF TOOLS

Choosing or replacing a set of tools to facilitate software development carries considerable risk. Many factors need to be weighed because you don't want to make the wrong selection. Above all, the selection needs to be made with safety at the forefront.

This paper has presented you with a set of factual criteria to aid in this tool selection. The table below summarizes the risk associated with each of the topics discussed. Choose your development platform based on minimizing risk and maximizing performance.

Table 1: Mitigating risk when buying software development tools

Risks	Lower Risk	Higher Risk
Compiler takes advantage of TriCore hardware performance and safety	Proprietary Compiler	Open-source Compiler
Linker allows selection of cores and memory mapping	Proprietary Compiler	Open-source Compiler
Too few debugger licenses force developers to wait	Mix high-end and streamlined debuggers for maximum efficiency	Fewer number of only high-end debuggers
Fragmented tool sets can reduce efficiency or corrupt data	Integrated tool suite from a single vendor	Mix of tools from multiple vendors
Find errors early and be assured of a quality development process	ASPICE Level 2 Certification	No ASPICE certification