Electronic Design

By RICK NELSON, Contributing Editor

Simplify Embedded-System Development with Open-Source Software

Sponsored by Texas Instruments: Embedded software development is a multidimensional effort that you can address by leveraging open-source compiler tools and operating systems.

The design, integration, and deployment of embedded-system software presents many challenges related to the operating system, development environment, middleware, compiler, and other software tools. Although you can purchase commercial software products to meet many of your requirements, you might find that open-source tools offer the optimum approach to some or all or your embedded software needs. But be sure to evaluate the tradeoffs before committing to a specific approach.

Commercial software generally comes with training and support. On the other hand, with open-source software you'll assume more development responsibility—but you can seek support from the open-source development community. Taking a mixed approach, you choose a commercial software package and employ open-source plug-ins or opt for versions of open-source software maintained and supported by a commercial software vendor. Several companies, for example, offer commercial versions of the open-source Linux operating system.

You also want to make sure that the vendor of your target processor supports open-source software. One such vendor is Texas Instruments, which supports the mainline Linux kernel and the kernel.org community organization. TI incorporates kernel.org's most recent, stable kernels into the software-development kits (SDKs) that support its Arm-based embedded processors.

TI provides new features and functionality as well as TI's bug fixes to the kernel.org community so that these improvements can be incorporated into mainline Linux. In turn, TI fully evaluates, documents, tests, and productizes SDKs utilizing mainline Linux kernels for its major processors.

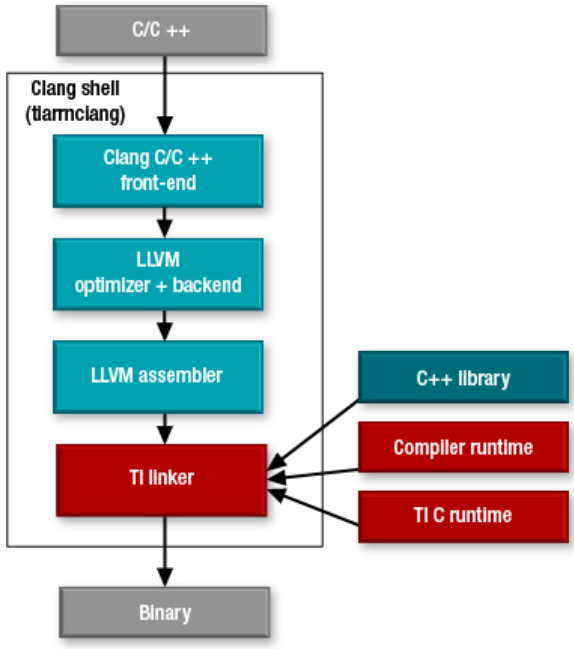
Due to the constant evolution of open-source projects, developers often face the dilemma of when to upgrade to a more recent Linux kernel version. The benefits of new features, functionality, and bug fixes incorporated in the new kernel must be weighed against the costs of migration, which include unexpected effects that can require considerable time and effort to overcome.

For example, discarding a patch during the migration process may have ramifications throughout the software environment, and changes during migration can make maintaining continuity in the code base difficult—thereby potentially compromising a software engineering team's previous development investment. TI looks to reduce the cost of migrations by allowing users to take advantage of a new kernel version's advantages. Its support ensures an efficient development environment and avoids the disruption and distraction that can accompany a migration to a new kernel, enabling orderly migration when necessary.

Open-Source Compiler Tools

In addition, you can leverage open-source software for compiler technology, as exemplified by the LLVM open-source project, which consists of a collection of modular and reusable compiler toolchain building blocks for creating compilers. (The LLVM Foundation emphasizes that LLVM is not an acronym.) A subproject of LLVM is Clang, a C/C++ compiler front end.

Clang offers several advantages, including compatibility with software written for the Gnu Compiler Collection (GCC). Compared with GCC, Clang produces faster, more efficient code that can more easily fit within a memory-constrained device, minimizing the need to choose a costlier



1. Clang, the LLVM optimizer, and the TI linker and C runtime library fit together to produce efficient code.

device with more memory.

LLVM and Clang have benefitted from the support of companies including Apple, Arm, Google, and Microsoft. Texas Instruments' support includes TI Arm Clang, a new set of compiler tools for TI Arm Cortex microcontrollers. TI Arm Clang combines the Clang front end and LLVM optimizer with proprietary technology, such as TI's linker and optimized C runtime library, to deliver optimal code size and minimize runtime footprints. *Figure 1* illustrates how Clang, the LLVM optimizer, and the TI linker and C runtime library fit together.

The TI Arm Clang toolchain produces efficient code as shown in *Figure 2*, which depicts the code size of a selection of software stacks—CoreSDK (including the real-time operating system and drivers), OpenThread, and the IEEE 802.15.4g stack—that are part of the SDKs for SimpleLink MCUs. The figure then compares the code size produced by TI Arm Clang (tiarmclang) to GCC and the previous TI Arm compiler (armcl).

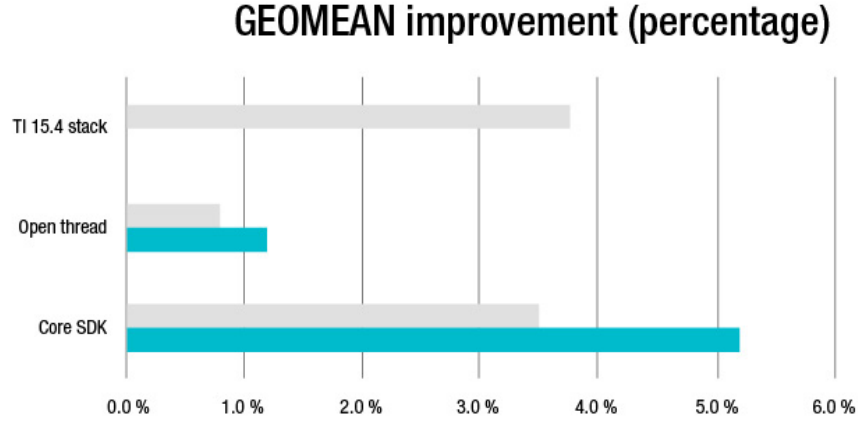
As *Figure 2* illustrates, for the CoreSDK, TI Arm Clang produces code that's 5% smaller than code produced by GCC and 3.5% smaller than code produced by armcl. These percentages may seem minor, but they can determine whether your application will fit into your chosen device's memory. In addition, Texas Instruments has reported that it plans to make further improvements to TI Arm Clang in 2021 that will significantly impact code size.

In addition to being compatible with code written for GCC, TI Arm Clang is supported by the latest SDKs for TI's CC3220, CC3230, CC3235, CC1312, CC1352, CC2642, and CC2652 microcontrollers. The SDKs include sample projects for TI Arm Clang as well as support for previous compilers. (TI has reported it will not introduce new features for armcl, but it will continue regular maintenance releases for bug fixes.) You can use TI Arm Clang today to develop Wi-Fi, Bluetooth Low Energy, Zigbee, IEEE 802.15.4, and other applications.

TI Arm Clang also offers support for code coverage (including function, line, region/statement, and branch coverage), a feature that's becoming increasingly important for functional-safety applications. In addition, current users of armcl will find that porting code or projects to TI Arm Clang is a smooth, simple process. TI Arm Clang uses the same TI linker as armcl; therefore, users needn't modify the linker command file.

Conclusion

Open-source software can offer advantages in many embedded applications, from Bluetooth Low Energy to IEEE 802.15.4 low-rate wireless networks. Nevertheless, it also presents challenges, such as forgoing the training and support that generally come with commercial software. Never-



2. TI Arm Clang (tiarmclang) produces efficient code compared with GCC and the previous TI Arm compiler (armcl).

theless, you may be able to bring the necessary open-source expertise in house with the help of the open-source community.

Open-source software tools span operating systems like Linux to LLVM compiler tools. With respect to the latter, the TI Arm Clang toolchain compiles efficient code while preserving compatibility, allowing you to leverage source code written for GCC and keep your code base portable.