Technical Article

The Future of Compiler Tools for TI Arm® Cortex®-Based MCUs

JohnS

TI Arm Clang is a new set of compiler tools for TI Arm Cortex microcontrollers and represents the future of the TI Arm compiler. This new toolchain is based on the LLVM project and uses Clang as the C/C++ front end. Ultimately this new toolchain will help customers produce more efficient applications.

LLVM and Clang

LLVM is an open-source project that is a collection of modular and reusable compiler toolchain technologies. In essence, it is an initiative consisting of building blocks for creating compilers.

Clang is a subproject of LLVM and is the C/C++ compiler front end. Both projects have significant momentum, including years of investment from companies like Apple, Arm, Google and Microsoft.

There are a number of advantages of moving to Clang, but the most significant ones are compatibility of software and code size.

Let’s look at compatibility first. There is a huge amount of software available written for use with GCC. To leverage this software, you must use a toolchain compatible with GCC extensions. This leads many people to select GCC as their toolchain. Unfortunately, the side effect of this is that GCC does not produce very efficient code, which means that your application is slower and larger (uses more memory).

Having Clang as the front end of TI's new toolchain enables you to use code written for GCC. Clang supports the vast majority of GCC extensions, meaning that the majority of C code written for GCC can be used with Clang.

For the TI Arm Clang solution, we have combined the Clang front end and LLVM optimizer with proprietary technology, such as our linker and optimized C runtime library, to deliver optimal code size and minimize runtime footprints. This enables a toolchain that is both compatible with GCC and produces efficient code. Figure 1 illustrates how Clang, the LLVM optimizer, and the TI linker and C runtime library fit together.

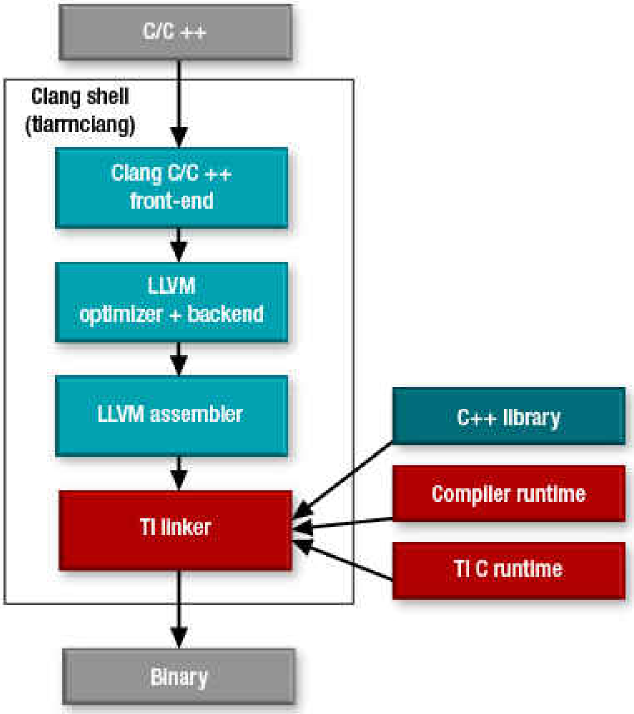

Figure 1. Block diagram of the TI Arm Clang toolchain

The importance of code size

Producing compact efficient code is critical for microcontroller (MCU) applications. Because devices are typically memory-constrained, it can be challenging to fit your application into available memory, leading to trade-offs or having no choice but to select a more expensive device with more memory.

Although the TI Arm Clang toolchain is still new and constantly improving, it is already producing very efficient code. Figure 2 measures the code size of a selection of software stacks (CoreSDK [real-time operating system and drivers], OpenThread, and IEEE 802.15.4g) that are part of the software development kits (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).

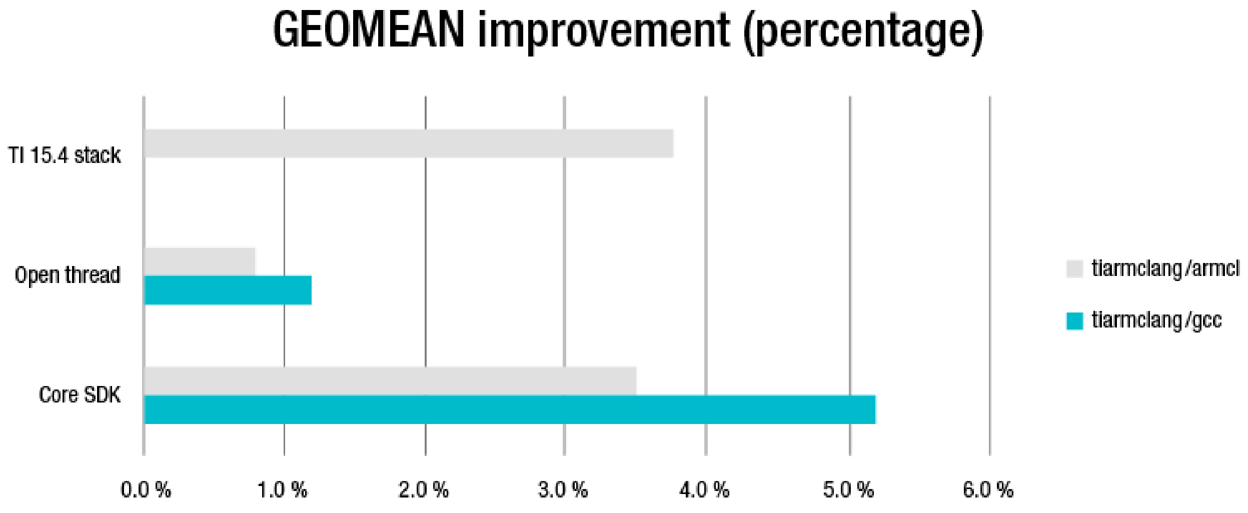

Figure 2. Code-size benchmark chart

As you can see from the figure, the TI Arm Clang toolchain produces more compact code. For the CoreSDK, TI Arm Clang code is 5% smaller than GCC and 3.5% smaller than armcl. These percentages may seem minor, but they can make a difference if your application is going to fit into memory. Beyond this, there are a number of improvements that TI plans to make to TI Arm Clang in 2021 that will have a significant effect on code size.

Software support

Having a new compiler is great, but a compiler is only useful if it is also supported by the software that you want to use. TI Arm Clang is compatible with code written for GCC, and it is supported by the latest Software Development Kits (SDKs) for CC3220, CC3230, CC3235, CC1312, CC1352, CC2642 and CC2652. The SDKs include example projects for TI Arm Clang in addition to previous compiler support. Whether you are looking to develop Wi-Fi®, *Bluetooth*® Low Energy, Zigbee or 15.4 applications, you can use TI Arm Clang today.

Additional benefits

By leveraging Clang, TI also takes advantage of support for current C and C++ language standards such as C++14 and C11, enabling you to take advantage of the latest language features in your application.

TI Arm Clang provides support for code coverage, including function, line, region/statement and branch coverage, which is becoming increasingly important for functional safety applications.

For current users of armcl, porting code or projects to TI Arm Clang is a smooth, simple process. One thing that aids in porting is that TI Arm Clang uses same TI linker as the armcl; therefore, there is no need to modify the linker command file.

Supporting the current TI Arm compiler

TI will continue to provide bug fixes for armcl. We will not introduce new features, but there will be regular maintenance releases for bug fixes. Each TI device family has its own timeline for supporting TI Arm Clang.

Conclusion

Historically, designers had to choose between software compatibility and an efficient compiler. With the new TI Arm Clang toolchain, you get both in a single toolchain. You can now leverage source code written for GCC and keep your code base portable, while also taking advantage of a compiler that produces efficient code.

TI Arm Clang is available for download today. Alternately, when downloading a compatible SDK from Resource Explorer within the Code Composer Studio™ integrated development environment, you also have the option to download the toolchain.

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265