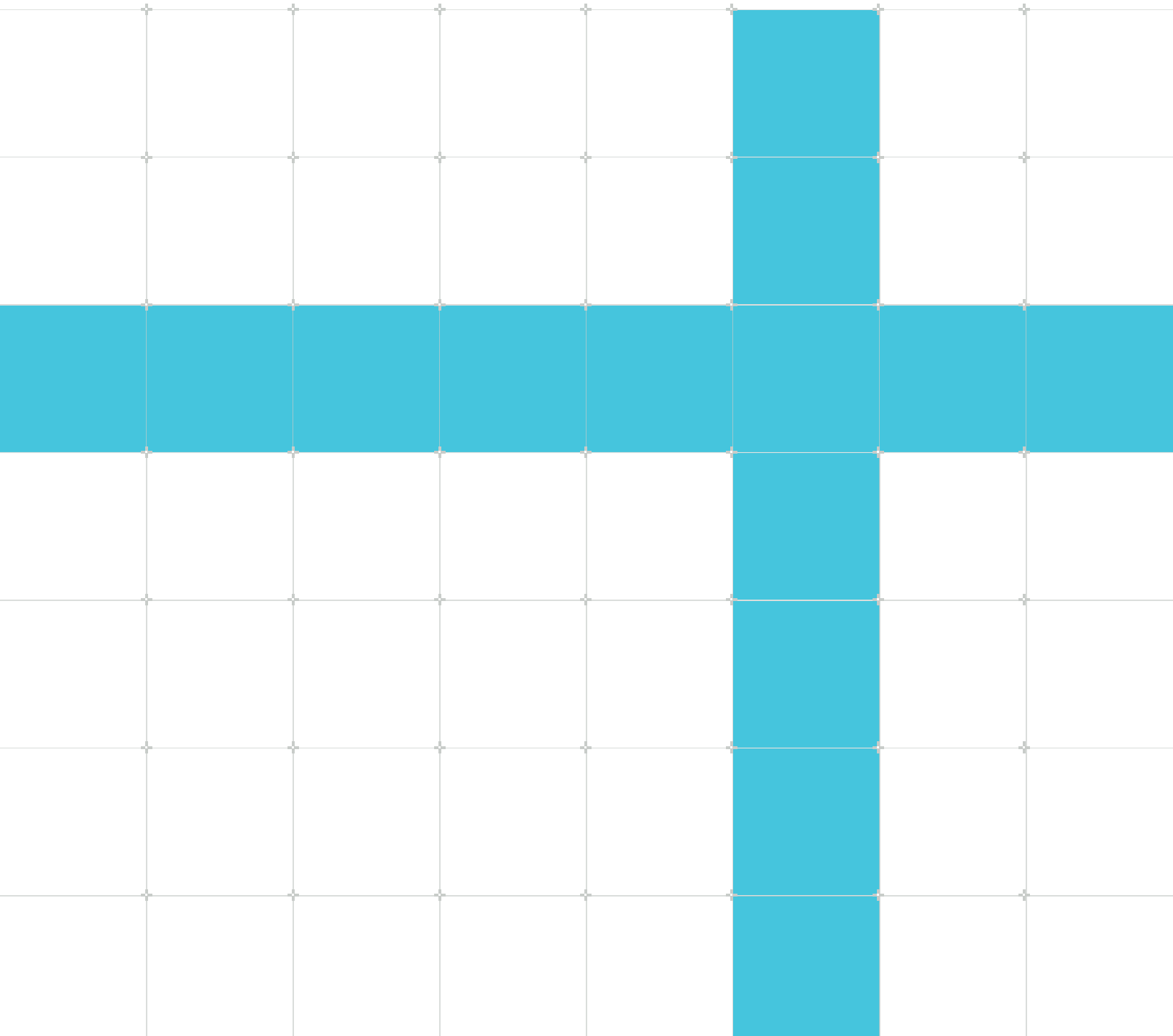
arm

Compiling C and C++ code for Arm

Version 1.0

Issue 02
102445_0100_02_en



Compiling C and C++ code for Arm

Release information

Document history

Issue	Date	Confidentiality	Change
0100-02	4 March 2019	Non-Confidential	Initial release

Proprietary Notice

or indirectly, in violation of such export laws. Use of the word "partner" in reference to Arm's customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of the Agreement shall prevail.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

(LES-PRE-20349)

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Unrestricted Access is an Arm internal classification.

Product Status

The information in this document is Final, that is for a developed product.

Feedback

Arm® welcomes feedback on this product and its documentation. To provide feedback on the product, create a ticket on https://support.developer.arm.com

To provide feedback on the document, fill the following survey: https://developer.arm.com/documentation-feedback-survey.

Inclusive language commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used language that can be offensive. Arm strives to lead the industry and create change.

We believe that this document contains no offensive language. To report offensive language in this document, email terms@arm.com.

Contents

1. Overview

When writing a C or C++ application, you'll need to compile it to machine code using a compiler toolchain. You can then run this compiled executable code on an Arm-based processor, or simulate it with a model.

Bare metal compilation

The compiler toolchain includes the following components:

- A compiler to translate C and C++ source code into machine code.
- An assembler to translate assembly language source code into machine code.
- A linker to combine multiple machine code modules into a single executable file.

Available toolchains include:

- Arm Compiler 6. The latest and most efficient Arm C/C++ compilation toolchain, based on the armclang compiler. Arm Compiler 6 maximizes the potential of Arm Cortex and Neoverse processors and architectures, from Armv6-M to Armv8-A 64-bit Arm: Evaluate as part of Arm Development Studio
- Arm Compiler 5. The previous generation Arm C/C++ compilation toolchain, based on the armcc compiler. Arm Compiler 5 provides stability and superb code size for legacy projects up to and including Armv7.
- The GNU toolchain. An open source, community-developed toolchain. The GNU toolchain provides a low-cost mechanism for developing on Arm platforms.

All these toolchains can be used standalone, from the command line, or integrated into Arm Development Studio or Keil MDK IDE environments.

Linux compilation

The common programming languages are well-supported on Arm - with most open-source tools available in packages provided by your Linux distribution. Commercial compilers for C++, C and Fortran are available from Arm in the Arm Allinea Studio.

The Arm commercial and GNU open-source compilers are tuned extensively for Arm servers and partner silicon, and are evolving rapidly. The highest performance is achieved using the most recent versions of these tools - which are not normally the default for Linux distributions. Read about some of this work in GNU GCC 8 and glibc 2.27.

Find more information on getting started with Linux compilation here.

Find more information about compilers and languages here.

2. Hello World with Arm Compiler 6

Are you looking for examples that you can follow to get you started with Arm Compiler 6?

- Bare-metal Hello World C using the Armv8 model is a tutorial that shows you how to build Hello World with Arm Compiler 6 and debug it on the Armv8 Fixed Virtual Platform (FVP). This tutorial uses Arm Compiler 6 within the Arm Development Studio environment.
- Compiling a Hello World example in the Arm Compiler User Guide shows you how to build and inspect an executable image from C/C++ source files. This tutorial uses standalone Arm Compiler 6 from the command line.

Evaluate as part of Arm Development Studio

3. Hello World with Arm Compiler for Linux

Are you looking for examples that you can follow to get you started with compiling C/C++ code with Arm Compiler for Linux?

- The Get Started topic in the Arm C/C++ Compiler Developer and Reference Guide describes how to compile and run a “Hello World” program.
- Using the compiler provides some useful information about how to use the compiler and how to better optimize your code using the supported compiler options.

Arm C/C++ Compiler is available alongside Arm Fortran Compiler and Arm Performance Libraries, in a single product called Arm Compiler for Linux.

4. Hello World with GCC

Looking for examples you can follow to get you started with GCC?

- Getting Started with Arm DS-5 CE and Armv8-A Foundation Platform uses the open-source GCC compiler within the Arm Development Studio environment. This tutorial guides you through the process of creating a simple bare-metal “Hello World” application and finally running it on the Armv8-A Foundation Platform provided with DS-5 Community Edition.
- The following video tutorials provide another example of creating a “Hello World” application with GCC. Explore Armv8 Model using DS-5 Development Tool
  - Video 1: Introduction to Armv8 Architecture and DS-5
  - Video 2: Install and Setup DS-5
  - Video 3: Create a Project for Armv8 Model
  - Video 4: Run an Application on Armv8 Model

5. Learn more about the Arm Compiler toolchain

The different tools in the compiler toolchain all work together to transform your source code into an executable binary that can run on an Arm-based processor. Understanding the different functions of these tools will help you to build your application successfully.

The following resources provide more information about Arm Compiler, for embedded software development:

- Introduction to Arm Compiler 6 provides an overview of the entire toolchain.
- The Arm Compiler User Guide provides introductory information and examples.
- Learn more about getting started with Linux compilation from the Arm Compiler Reference Guide.
- More detailed information about the individual tools in the Arm Compiler toolchain is provided by the full Arm Compiler documentation set.
- Embedded Systems Fundamentals with Arm Cortex-M based Microcontrollers: A Practical Approach by Dr Alexander G. Dean includes information about the tools used to develop embedded software, including the compiler toolchain.
- Some Arm training courses include software development with Arm Compiler.

The following resources provide more information about Arm Compiler for Linux, for Server and HPC software development:

- Arm Compiler for Linux provides product information, details the available documentation, and describes how to license or get support for the product.
- Release history describes the highlights of each release of the product, including copies of their Release Notes.
- The Arm C/C++ Compiler Developer and Reference Guide describes how to use the Arm C/C++ Compiler component of Arm Compiler for Linux.