



Targeting processors, floating-point units, and NEON

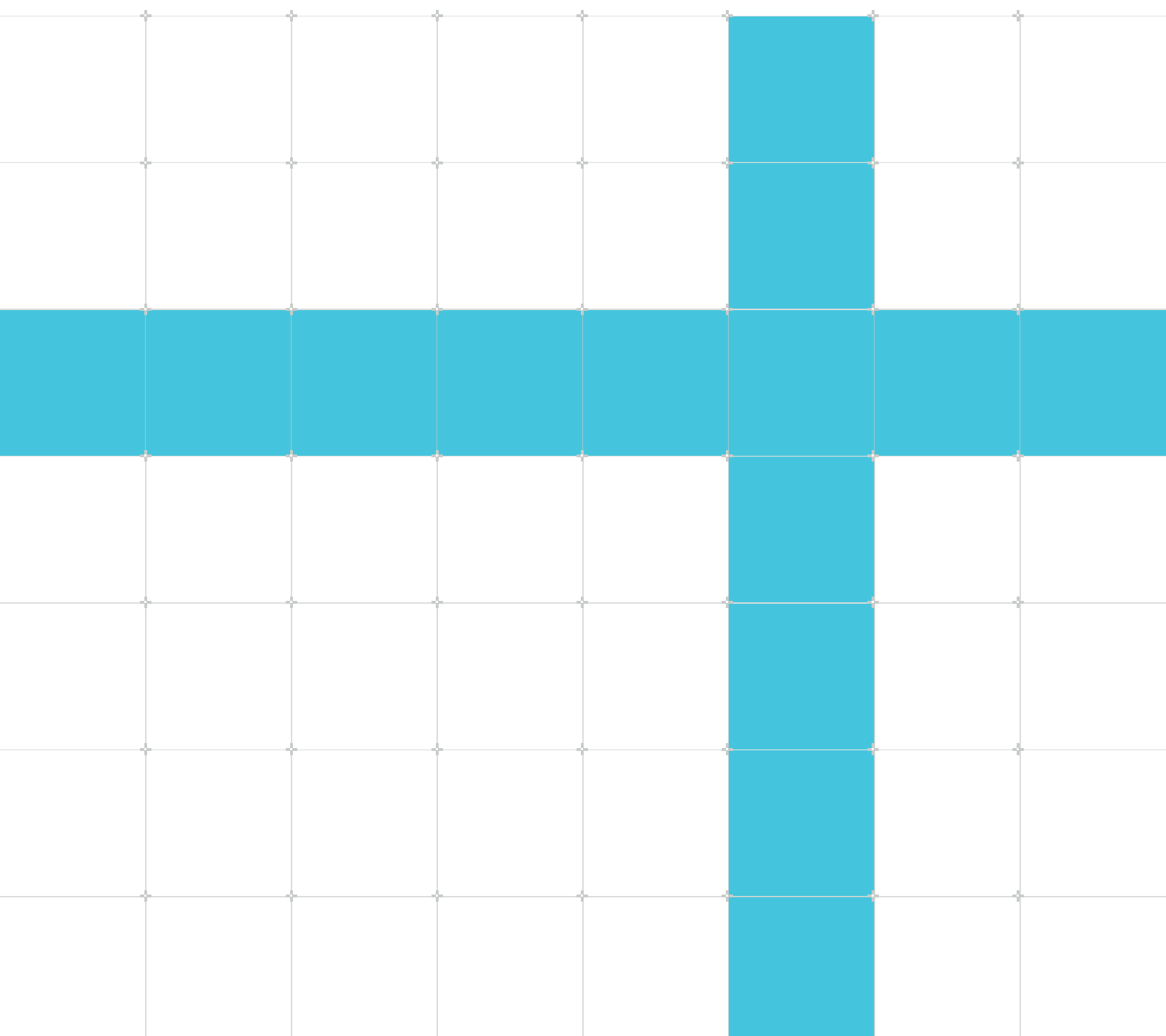
Version 1.0

Non-Confidential

Copyright © 2019 Arm Limited (or its affiliates).
All rights reserved.

Issue 01

102731_0100_01_en



Targeting processors, floating-point units, and NEON

Copyright © 2019 Arm Limited (or its affiliates). All rights reserved.

Release information

Document history

| Issue | Date | Confidentiality | Change |
|---------|-----------------|------------------|---------------|
| 0100-01 | 4 November 2019 | Non-Confidential | First release |

Proprietary Notice

This document is protected by copyright and other related rights and the practice or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of Arm. No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether implementations infringe any third party patents.

THIS DOCUMENT IS PROVIDED "AS IS". ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, patents, copyrights, trade secrets, or other rights.

This document may include technical inaccuracies or typographical errors.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

This document consists solely of commercial items. You shall be responsible for ensuring that any use, duplication or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly

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.

The Arm corporate logo and words marked with ® or ™ are registered trademarks or trademarks of Arm Limited (or its affiliates) in the US and/or elsewhere. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners. Please follow Arm’s trademark usage guidelines at <https://www.arm.com/company/policies/trademarks>.

Copyright © 2019 Arm Limited (or its affiliates). All rights reserved.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

(LES-PRE-20349|version 21.0)

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..... | 6 |
| 2. Selecting the target processor..... | 7 |
| 3. Selecting the target FPU..... | 12 |
| 4. Enabling NEON..... | 13 |
| 5. Further reading..... | 16 |

1. Overview

Arm DS-5 Development Studio tutorial for selecting specific processors with Arm Compiler to maximize performance, selecting FPU and enabling NEON.

This tutorial assumes you have installed and licensed Arm DS-5 Development Studio. For more information, see [Getting Started with Arm DS-5 Development Studio](#).

2. Selecting the target processor

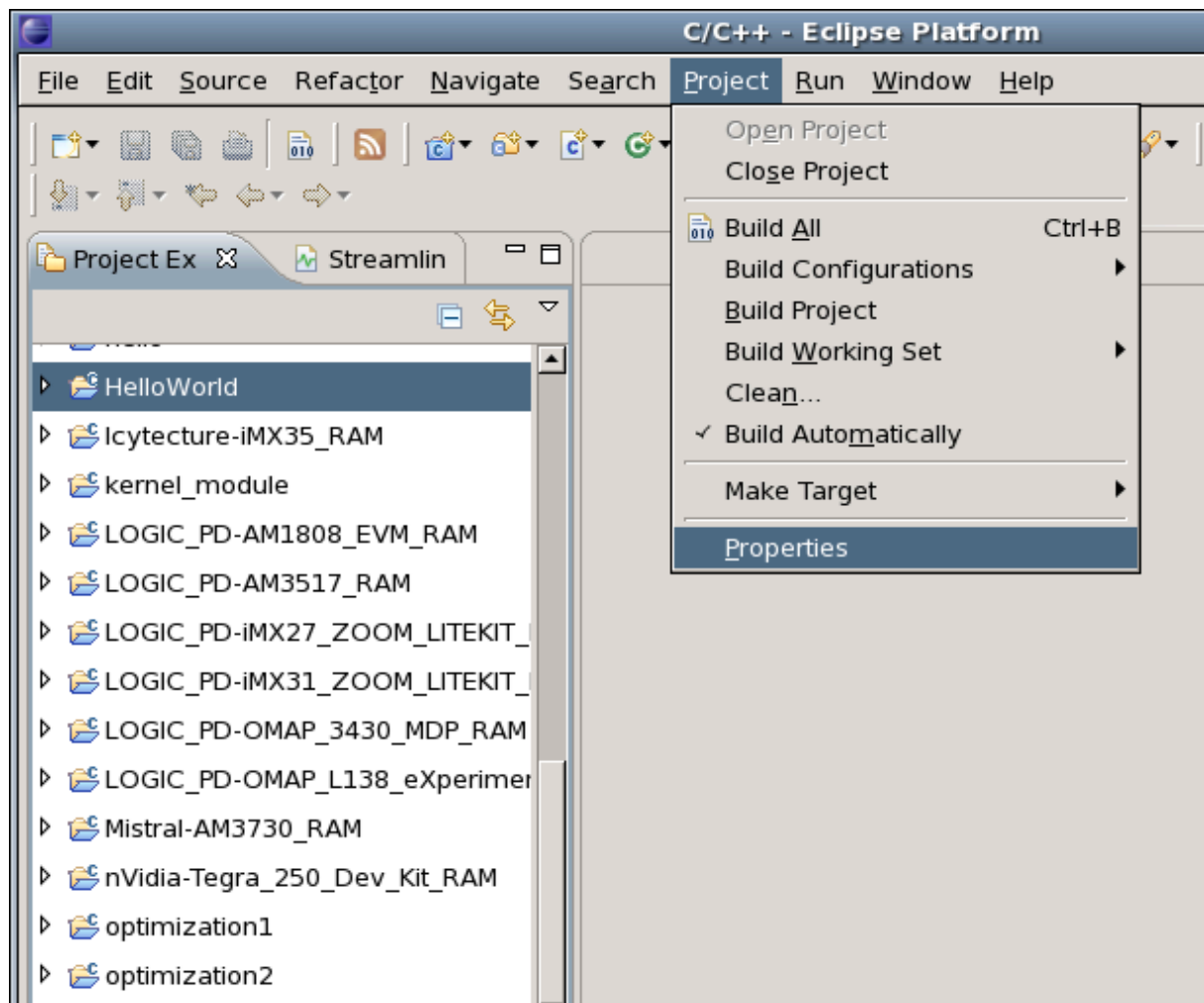
The Arm Compiler lets you target either an architecture or a specific processor target when generating code:

- Specifying an architecture provides the greatest code compatibility. The generated code can run on any processor supporting that architecture.
- Specifying a particular processor provides optimum performance. The compiler can use processor-specific features such as instruction scheduling to generate optimized code for that specific processor.

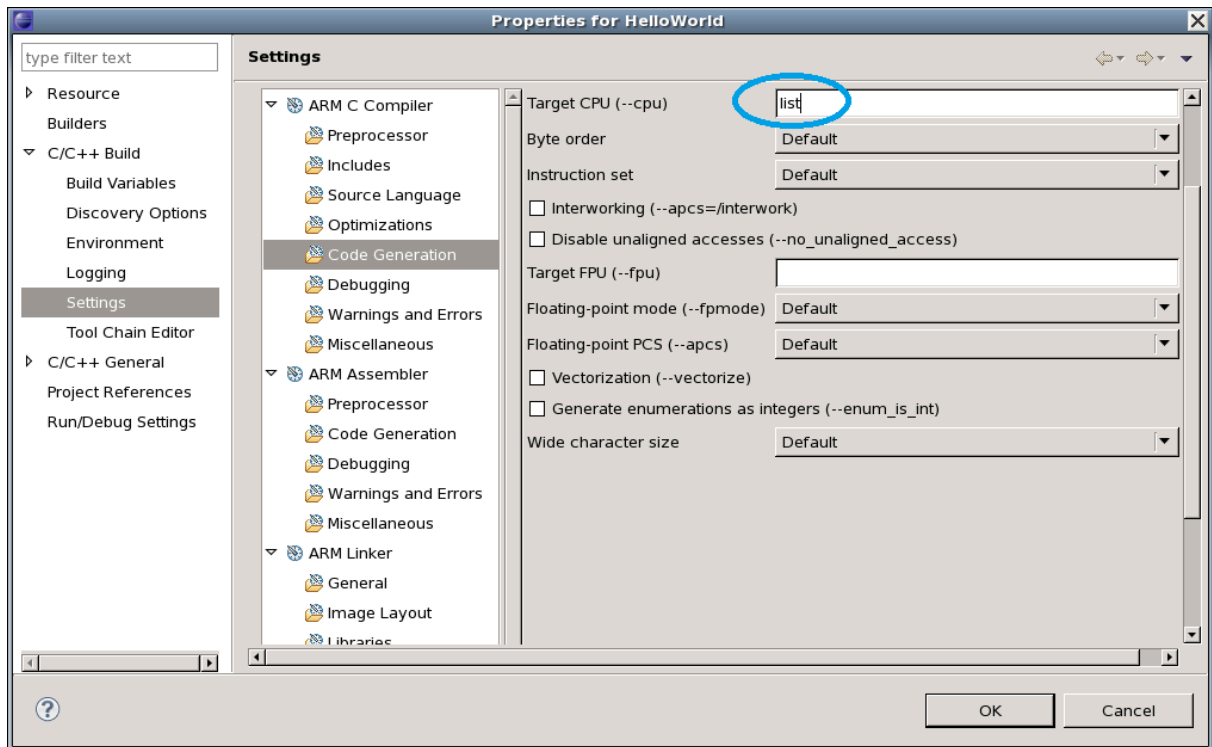
The **-cpu** [command-line option](#) lets you specify the name of either an architecture or a specific processor target.

To configure the **-cpu** option in DS-5:

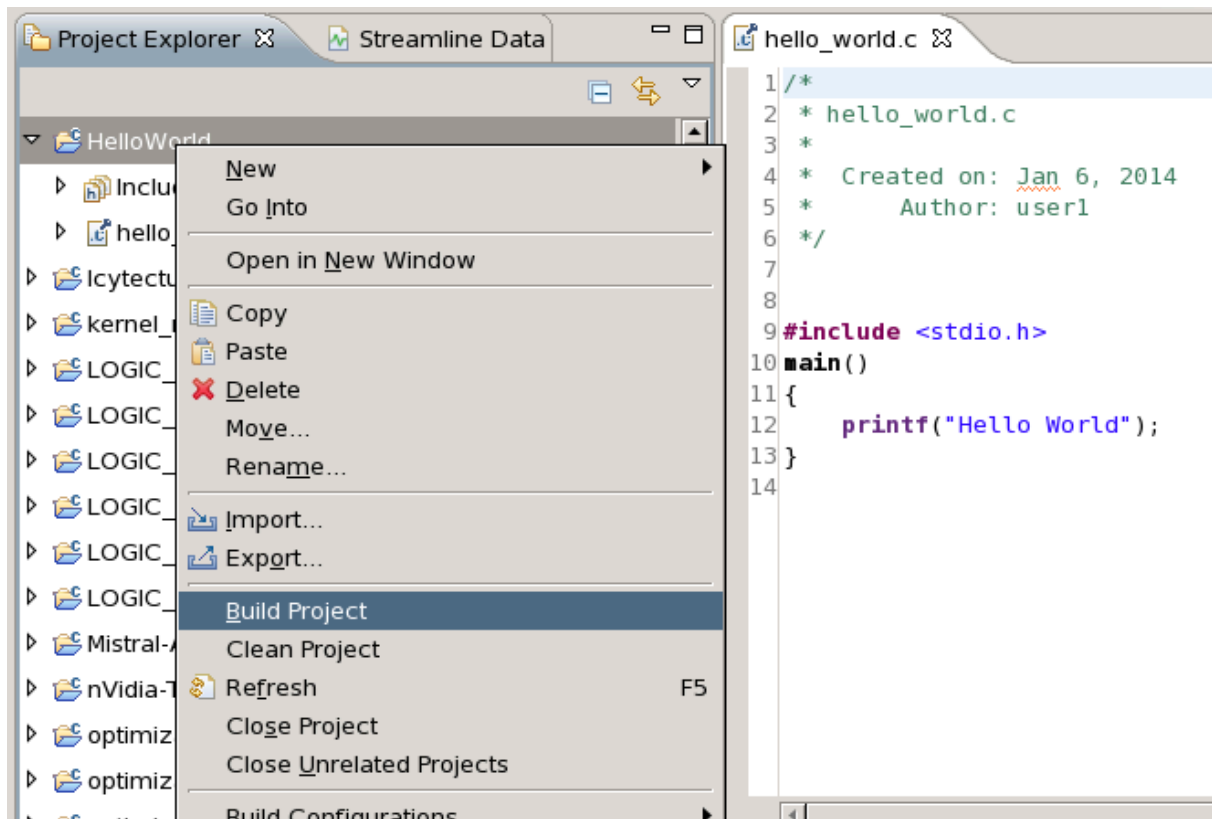
1. Select your project in the **Project Explorer** view.
2. Select **Project > Properties** from the main menu to display the **Properties** dialog box.

Figure 2-1: Project properties menu option

3. Expand **C/C++ Build**, then **Settings** in the **Properties** dialog box.
4. On the **Tool Settings** tab, select **Arm C Compiler > Code Generation** to display the code generation settings.
5. Enter a value for **Target CPU (-cpu)**.

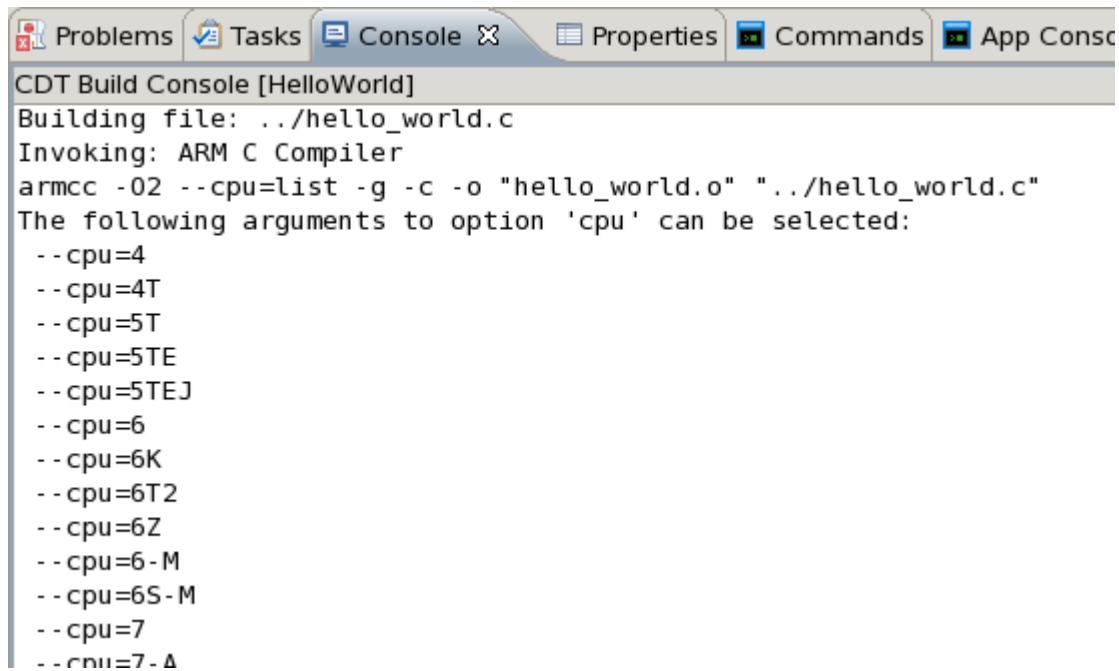
Figure 2-2: Enter a value for the Target CPU

6. Click **OK** to save the settings.

Figure 2-3: Build the project

The **Target CPU (-cpu)** setting lets you configure the **-cpu** option.

You can see a list of all supported architecture and processor names by specifying `list` for the **Target CPU (-cpu)** setting, then building your project. The console (**Window > Show View > Console**) shows the list of architecture and processor names.

Figure 2-4: Console window view

```
CDT Build Console [HelloWorld]
Building file: ../hello_world.c
Invoking: ARM C Compiler
armcc -O2 --cpu=list -g -c -o "hello_world.o" "../hello_world.c"
The following arguments to option 'cpu' can be selected:
--cpu=4
--cpu=4T
--cpu=5T
--cpu=5TE
--cpu=5TEJ
--cpu=6
--cpu=6K
--cpu=6T2
--cpu=6Z
--cpu=6-M
--cpu=6S-M
--cpu=7
--cpu=7-A
```

If the compiled program is to run on a specific Arm architecture-based processor, select the target processor. For example, to compile code to run on a Cortex-A9 processor use the **Target CPU (-cpu)** setting **Cortex-A9**.

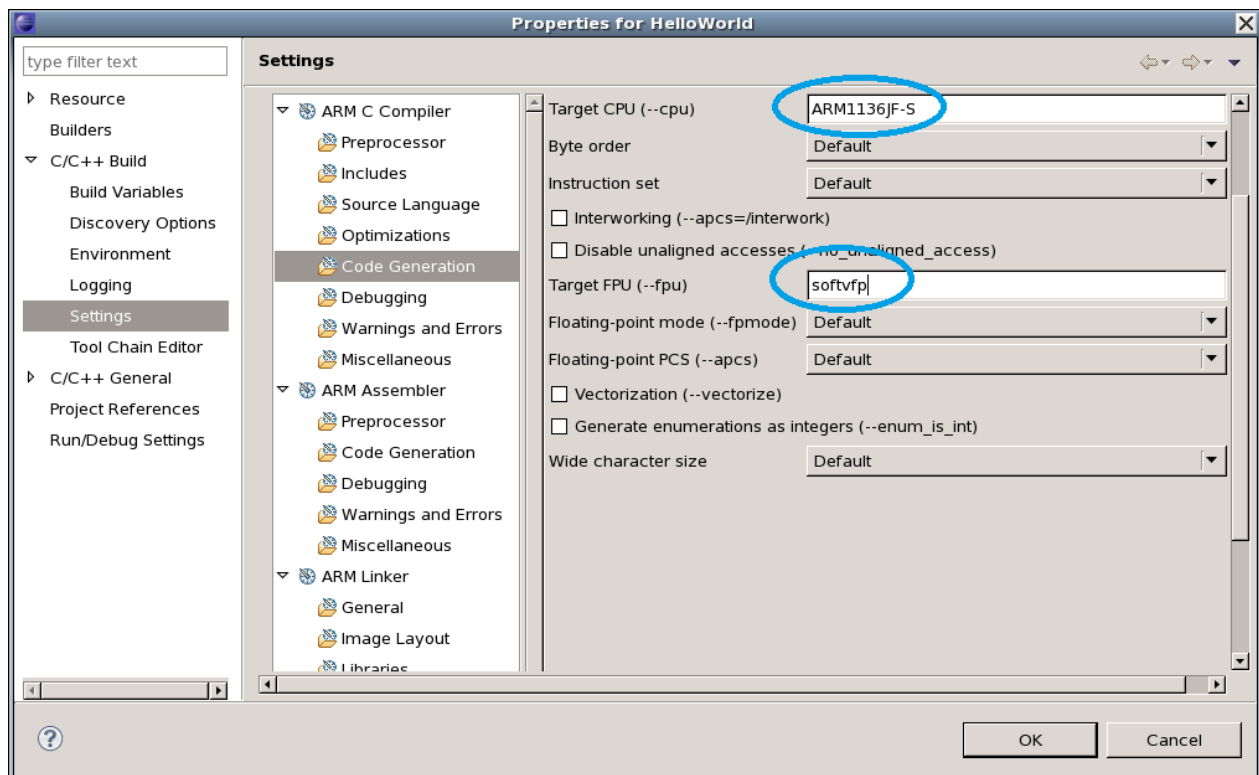
Alternatively, if the compiled program is to run on different Arm processors, choose the lowest common denominator architecture appropriate for the application and then specify that architecture in place of the processor name. For example, to compile code for processors supporting the Armv7 architecture use the **Target CPU (-cpu)** setting **7**.

3. Selecting the target FPU

Every **-cpu** target has an associated implicit Floating-Point Unit (FPU). A full list is available in [Processors and their implicit Floating-Point Units \(FPUs\)](#) in the [Arm Compiler armcc User Guide](#).

However, you can use the **-fpu** [command-line option](#) to override the implicit FPU. For example, the option **-cpu=ARM1136JF-S -fpu=softvfp** generates code that uses the software floating-point library `fp111b`, even though the choice of processor implies the use of architecture VFPv2.

Figure 3-1: Target CPU options



To configure the **-fpu** option in DS-5, use the **“Target FPU (--fpu)”** setting. This setting is in the same location on the Properties dialog box as the **“Target CPU (--cpu)”** setting discussed above.

You can see a list of all supported FPU architectures by specifying `list` for the **“Target FPU (--fpu)”** setting, then building your project. The console (**Window > Show View > Console**) shows the list of FPU architectures.

4. Enabling NEON

Arm NEON technology is the implementation of the Advanced SIMD architecture extension. It is a 64 and 128-bit hybrid SIMD technology targeted at advanced media and signal processing applications and embedded processors.

Specific NEON instructions let you use the NEON unit to perform operations in parallel on multiple lanes of data.

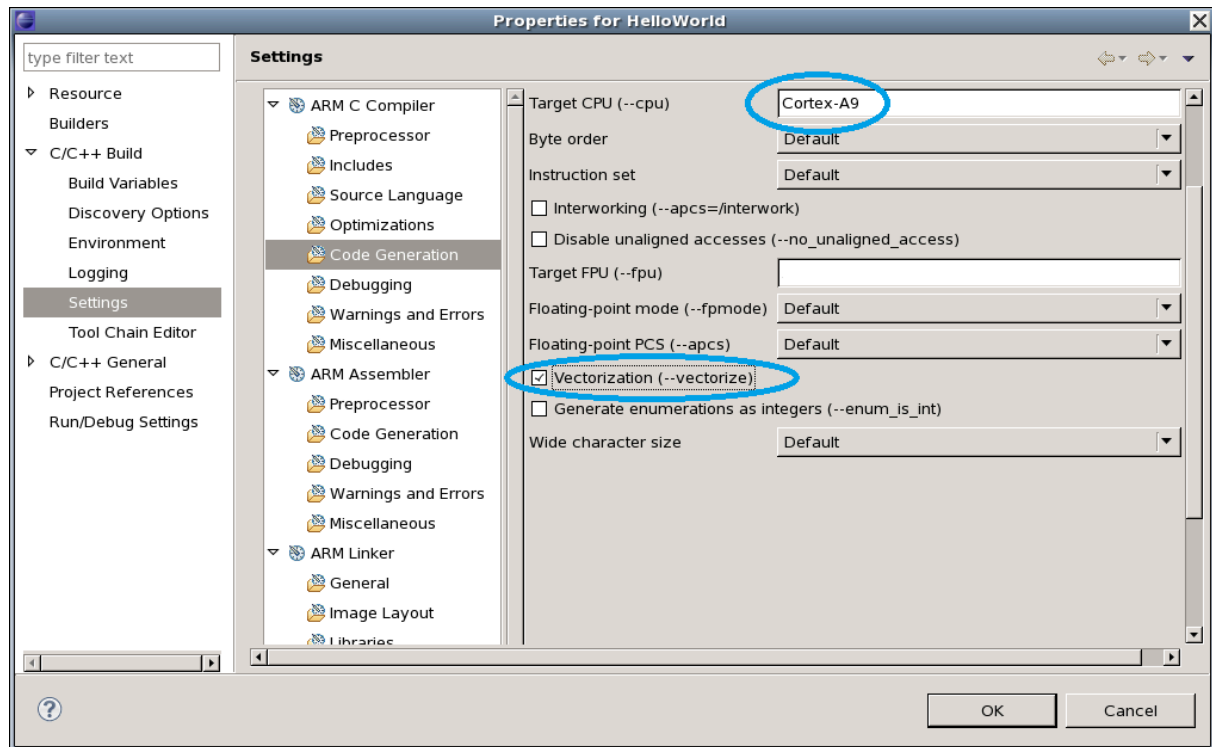
There are a number of different methods of creating code that uses NEON instructions:

- Write assembly language, or use embedded assembly language in C, and use the NEON instructions directly.
- Write in C or C++ using the NEON intrinsics.
- Call a library routine that has been optimized to use NEON instructions.
- Have the compiler use automatic vectorization to optimize loops for NEON.

To enable automatic vectorization you must target a processor that has a NEON unit. The required command line options are:

1. A target **-cpu** that has NEON capability, for example Cortex-A7, Cortex-A8, Cortex-A9, Cortex-A12, or Cortex-A15.

To configure this in DS-5, use **Target CPU (-cpu)** on the **Arm C Compiler > Code Generation** settings in the **Properties** dialog box.

Figure 4-1: Code generation settings

2. **-vectorize** to enable NEON vectorization.

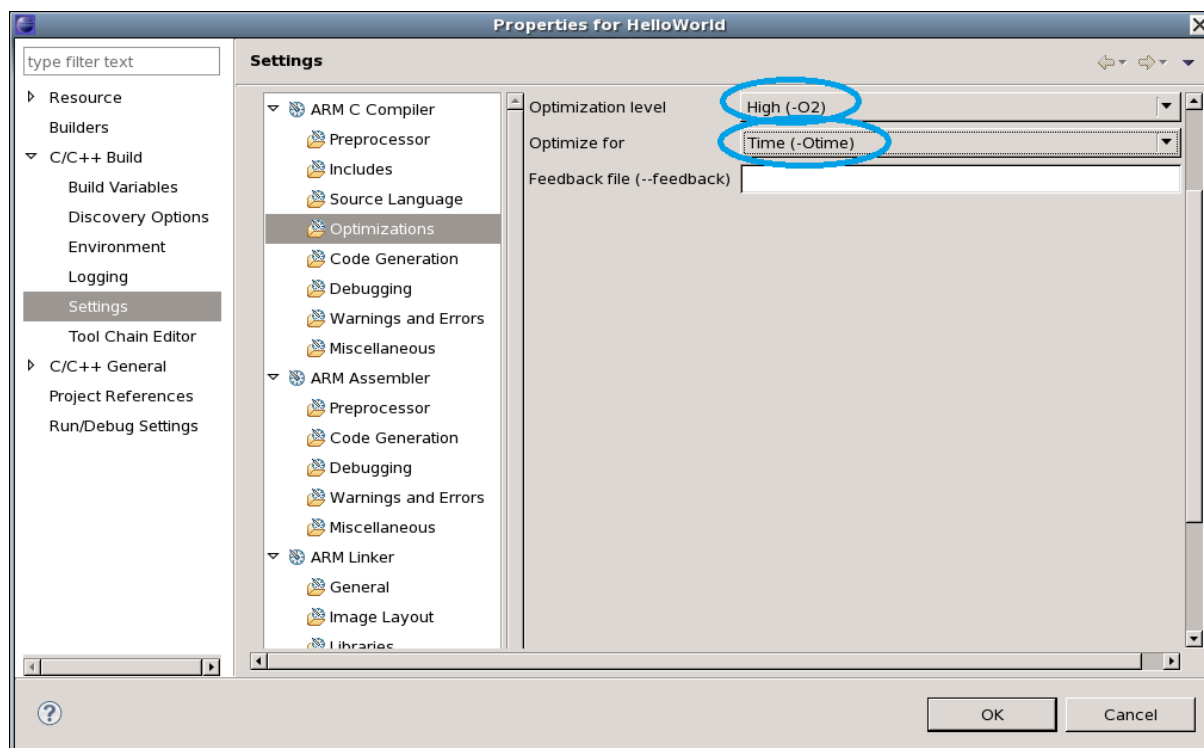
To configure this in DS-5, use **Vectorization (--vectorize)** on the **Arm C Compiler > Code Generation** settings in the **Properties** dialog box.

3. **-O2** (default) or **-O3** optimization level.

To configure this in DS-5, use **Optimization level** on the **Arm C Compiler > Optimizations** settings in the **Properties** dialog box.

4. **-Otime** to optimize for performance instead of code size.

To configure this in DS-5, use **Optimize for** on the **Arm C Compiler > Optimizations** settings in the **Properties** dialog box.

Figure 4-2: Optimization settings**Note**

You may need to [enable the NEON unit](#) before you can use NEON instructions.

5. Further reading

Here are some resources related to material in this guide:

- [Arm Compiler armcc User Guide](#)
- [Using the NEON Vectorizing Compiler](#)