

Home (/)

Products (/product/)

Download (/download/)

Events (/events/)

Search Keil...

+Go

Support (/support/)

Videos (http://www2.keil.com/video)

Technical Support

- Overview (/support/)
- Search (/home/searchhelp)
- Contact (/support/contact.asp)
- Assistance Request (/support/request.asp)
- Feedback (/support/feedback.asp)

On-Line Manuals

- Product Manuals (/support/man/)
- Document Conventions (/support/man/conventions.asp)

Compiler User Guide

- Preface (armcc_deb1353593789871.htm)
- Overview of the Compiler (armcc_chr1359124191020.htm)
- Getting Started with the Compiler (armcc_chr1359124194109.htm)
- Privacy Policy Update (armcc_chr1359124210536.htm)
- Arm's Privacy Policy has been updated. By continuing to use our site, you consent to Arm's Privacy Policy. Please view our Privacy Policy (company/privacy) to learn more. (armcc_chr1359124220881.htm)
- The compiler also optimizes and transfers (armcc_chr135912421178.htm)
- Compiler optimization for code size versus speed (armcc_chr1359124221443.htm)
- Compiler optimization levels and the debug view (armcc_chr1359124221739.htm)
- Select the target processor at compile time (company/cookiepolicy). (armcc_chr1359124221958.htm)
- Debugging FPU messages again (armcc_pge1416481958654.htm)
- Change Settings (company/cookiesettings/) in C code

Home (/) / Compiler User Guide

(armcc_chr1359124221958.htm) (default.htm) (armcc_chr1359124221443.htm) **Compiler optimization levels and the debug view**

Home (default.htm) » Compiler Coding Practices (armcc_chr1359124220881.htm) » Compiler optimization levels and the debug view

4.3 Compiler optimization levels and the debug view

The precise optimizations performed by the compiler depend both on the level of optimization chosen, and whether you are optimizing for performance or code size.

The compiler supports the following optimization levels:

0
Minimum optimization. Turns off most optimizations. When debugging is enabled, this option gives the best possible debug view because the structure of the generated code directly corresponds to the source code. All optimization that interferes with the debug view is disabled. In particular:


- Breakpoints can be set on any reachable point, including dead code.
- The value of a variable is available everywhere within its scope, except where it is uninitialized.
- Backtrace gives the stack of open function activations that is expected from reading the source.


Note


Although the debug view produced by `-o0` corresponds most closely to the source code, users might prefer the debug view produced by `-o1` because this improves the quality of the code without changing the fundamental structure.


Note


(armcc_chr1359124222426.htm)


 Loop unrolling in C code
(armcc_chr1359124222660.htm)


 Compiler optimization and the volatile keyword
(armcc_chr1359124222941.htm)


 Code metrics
(armcc_chr1359124223206.htm)


 Code metrics for measurement of code size and data
(armcc_chr1359124223455.htm)


 Stack use in C and C++
(armcc_chr1359124223721.htm)


 Benefits of reducing debug information in objects
(armcc_chr1359124223955.htm)


 Methods of reducing debug information in objects a
(armcc_chr1359124224235.htm)

 Guarding against multiple inclusion of header file
(armcc_chr1359124224501.htm)


 Methods of minimizing function parameter passing o
(armcc_chr1359124224750.htm)


 Returning structures from functions through regist
(armcc_chr1359124225000.htm)


 Functions that return the same result when called
(armcc_chr1359124225265.htm)


 Comparison of pure and impure functions
(armcc_chr1359124225530.htm)


Privacy Policy Update
(armcc_chr1359124225530.htm)


 ARM's Privacy Policy has been updated by ARM Limited. Use our site, you consent to ARM's Privacy Policy.
(armcc_chr1359124225842.htm)


 ARM's Privacy Policy (privacy) to learn more about ARM's privacy and data handling.
(armcc_chr1359124226043.htm)

 Compiler decisions on function inlining
(armcc_chr1359124226326.htm)

Accept and hide this message
 Automatic function inlining and static functions
(armcc_chr1359124226591.htm)

 This site uses cookies to store information on your device and to improve our site, you consent to our use of cookies.
(armcc_chr1359124226825.htm)

 ARM's Privacy Policy (privacy) to learn more about ARM's privacy and data handling.
(armcc_chr1359124227000.htm)

Don't show this message again
 Restriction on overriding compiler decisions about compiler settings
(armcc_chr1359124227340.htm)

Dead code includes reachable code that has no effect on the result of the program, for example an assignment to a local variable that is never used. Unreachable code is specifically code that cannot be reached via any control flow path, for example code that immediately follows a return statement.

1

Restricted optimization. The compiler only performs optimizations that can be described by debug information. Removes unused inline functions and unused static functions. Turns off optimizations that seriously degrade the debug view. If used with `--debug`, this option gives a generally satisfactory debug view with good code density. The differences in the debug view from `-O0` are:

- Breakpoints cannot be set on dead code.
- Values of variables might not be available within their scope after they have been initialized. For example if their assigned location has been reused.
- Functions with no side-effects might be called out of sequence, or might be omitted if the result is not needed.
- Backtrace might not give the stack of open function activations that is expected from reading the source because of the presence of tailcalls.

The optimization level `-O1` produces good correspondence between source code and object code, especially when the source code contains no dead code. The generated code can be significantly smaller than the code at `-O0`, which can simplify analysis of the object code.

2

High optimization. If used with `--debug`, the debug view might be less satisfactory because the mapping of object code to source code is not always clear. The compiler might perform optimizations that cannot be described by debug information.

This is the default optimization level.

The differences in the debug view from `-O1` are:

- The source code to object code mapping might be many to one, because of the possibility of multiple source code locations mapping to one point of the file, and more aggressive instruction scheduling.
- Instruction scheduling is allowed to cross sequence points. This can lead to mismatches between the reported value of a variable at a particular point, and the value you might expect from reading the source code.
- The compiler automatically inlines functions.






















3

Maximum optimization. When debugging is enabled, this option typically gives a poor debug view. ARM recommends debugging at lower optimization levels.

If you use `-O3` and `-Otime` together, the compiler performs extra optimizations that are more aggressive, such as:

- High-level scalar optimizations, including loop unrolling. This can give significant performance benefits at a small code size cost, but at the risk of a longer build time.
- More aggressive inlining and automatic inlining.

These optimizations effectively rewrite the input source code, resulting in object code with the lowest correspondence to source code and the worst debug view. The `--loop_optimization_level=option` controls the

	Compiler modes and inline functions (armcc_chr1359124227574.htm)
	Inline functions in C++ and C90 mode (armcc_chr1359124227808.htm)
	Inline functions in C99 mode (armcc_chr1359124228026.htm)
	Inline functions and debugging (armcc_chr1359124228276.htm)
	Types of data alignment (armcc_chr1359124228525.htm)
	Advantages of natural data alignment (armcc_chr1359124228744.htm)
	Compiler storage of data objects by natural byte a (armcc_chr1359124228978.htm)
	Relevance of natural data alignment at compile time (armcc_chr1359124229212.htm)
	Unaligned data access in C and C++ code (armcc_chr1359124229461.htm)
	The __packed qualifier and unaligned data access i (armcc_chr1359124229695.htm)
	Unaligned fields in structures (armcc_chr1359124229929.htm)
	Performance penalty associated with marking whole (armcc_chr1359124230195.htm)
	Privacy Policy Update (armcc_chr1359124230216.htm)
	Unaligned pointers in C and C++ code (armcc_chr1359124230216.htm)
	Arm's Privacy Policy has been updated. By continuing to use our site, you consent to use our Privacy Policy. (armcc_chr1359124230216.htm)
	Unaligned Load Register (LDR) instructions generate a warning (armcc_chr1359124230710.htm)
	Compiler support for floating-point arithmetic (armcc_chr1359124231193.htm)
	Default selection of hardware or software floating-point arithmetic (armcc_chr1359124231427.htm)
	Example of hardware and software floating-point arithmetic (armcc_chr1359124231692.htm)
	Don't show this message again (armcc_chr1359124231926.htm)
	Change Settings (company/cookiesettings/)

amount of loop optimization performed at `-O3 -Otime`. The higher the amount of loop optimization the worse the correspondence between source and object code.

For extra information about the high level transformations performed on the source code at `-O3 -Otime` use the `--remarks` command-line option.

Because optimization affects the mapping of object code to source code, the choice of optimization level with `-Ospace` and `-Otime` generally impacts the debug view.

The option `-O0` is the best option to use if a simple debug view is required. Selecting `-O0` typically increases the size of the ELF image by 7 to 15%. To reduce the size of your debug tables, use the `--remove_unneeded_entities` option.

Related concepts

4.12 Benefits of reducing debug information in objects and libraries (armcc_chr1359124223955.htm)

Related reference

4.13 Methods of reducing debug information in objects and libraries (armcc_chr1359124224235.htm)

7.34 --debug, --no_debug (armcc_chr1359124909829.htm)

7.35 --debug_macros, --no_debug_macros (armcc_chr1359124910063.htm)

7.52 --dwarf2 (armcc_chr1359124915882.htm)

7.53 --dwarf3 (armcc_chr1359124916116.htm)

7.116 -Onum (armcc_chr1359124935804.htm)





















7.121 -Ospace (armcc_chr1359124936615.htm)





















7.122 -Otime (armcc_chr1359124936849.htm)

7.140 --remove_unneeded_entities, --no_remove_unneeded_entities (armcc_chr1359124942060.htm)

Related information

ELF for the ARM Architecture (<http://infocenter.arm.com/help/topic/com.arm.doc.ih0044-/index.html>)

-  [Limitations on hardware handling of floating-point \(armcc_chr1359124232160.htm\)](#)
-  [Implementation of Vector Floating-Point \(VFP\) supp \(armcc_chr1359124232426.htm\)](#)
-  [Compiler and library support for half-precision fl \(armcc_chr1359124232675.htm\)](#)
-  [Half-precision floating-point number format \(armcc_chr1359124233455.htm\)](#)
-  [Compiler support for floating-point computations a \(armcc_chr1359124233705.htm\)](#)
-  [Types of floating-point linkage \(armcc_chr1359124233939.htm\)](#)
-  [Compiler options for floating-point linkage and co \(armcc_chr1359124234220.htm\)](#)
-  [Floating-point linkage and computational requireme \(armcc_chr1359124234516.htm\)](#)
-  [Processors and their implicit Floating-Point Units \(armcc_chr1359124234797.htm\)](#)
-  [Integer division-by-zero errors in C code \(armcc_chr1359124235078.htm\)](#)
-  [Software floating-point division-by-zero errors in \(armcc_chr1359124236294.htm\)](#)
-  [Privacy Policy Update \(armcc_chr1359124236400.htm\)](#)
-  [Software floating-point division-by \(armcc_chr1359124236778.htm\)](#)
-  [Software floating-point division-by-zero debugging \(armcc_chr1359124237028.htm\)](#)
-  [New language features of C99 \(armcc_chr1359124237339.htm\)](#)
-  [New C99 features in C99 \(armcc_chr1359124237527.htm\)](#)
-  [Information in C99 and C99 \(armcc_chr1359124237761.htm\)](#)
-  [Compound literals in C99 \(armcc_chr1359124238000.htm\)](#)
-  [Designated initializers in C99 \(armcc_chr1359124238200.htm\)](#)
-  [Hexadecimal floating-point numbers in C99 \(armcc_chr1359124238525.htm\)](#)

-  Flexible array members in C99
(armcc_chr1359124238837.htm)
-  `__func__` predefined identifier in C99
(armcc_chr1359124239071.htm)
-  inline functions in C99
(armcc_chr1359124239337.htm)
-  long long data type in C99 and C90
(armcc_chr1359124239586.htm)
-  Macros with a variable number of arguments in C99
(armcc_chr1359124239851.htm)
-  Mixed declarations and statements in C99
(armcc_chr1359124240101.htm)
-  New block scopes for selection and iteration state
(armcc_chr1359124240335.htm)
-  `_Pragma` preprocessing operator in C99
(armcc_chr1359124240616.htm)
-  Restricted pointers in C99
(armcc_chr1359124240881.htm)
-  Additional library functions in C99
(armcc_chr1359124241146.htm)
-  Complex numbers in C99
(armcc_chr1359124241427.htm)
-  Boolean type and `_Bool` in C99
(armcc_chr1359124241645.htm)
-  Extended integer types and functions in C99
(armcc_chr1359124241911.htm)
-  Floating-point environments in C99
(armcc_chr1359124242207.htm)
-  `snprintf` family of functions in C99
(armcc_chr1359124242331.htm)
-  type-generic math macros in C99
(armcc_chr1359124242891.htm)
-  wide character I/O functions in C99
(armcc_chr1359124243049.htm)
-  How to prevent uninitialized data
(armcc_chr1359124243221.htm)
-  Change Settings
(company/cookiesettings.htm)
-  Compiler Diagnostic Messages
(armcc_chr1359124243783.htm)

- Using the Inline and Embedded Assemblers of the AR (armcc_chr1359124245889.htm)
- Compiler Command-line Options (armcc_chr1359124898004.htm)
- Language Extensions (armcc_chr1359124953729.htm)
- Compiler-specific Features (armcc_chr1359124965789.htm)
- C and C++ Implementation Details (armcc_chr1359125008566.htm)
- What is Semihosting? (armcc_pge1358787045051.htm)
- Via File Syntax (armcc_chr1359125030640.htm)
- Summary Table of GNU Language Extensions (armcc_chr1359125031592.htm)
- Standard C Implementation Definition (armcc_chr1359125032122.htm)
- Standard C++ Implementation Definition (armcc_chr1359125036178.htm)
- C and C++ Compiler Implementation Limits (armcc_chr1359125037582.htm)

Products (/product/)

Development Tools

Arm (/Arm/)

C166 (/c166/)

C51 (/c51/)

C251 (/c251/)

Privacy Policy Update

Arm's Privacy Policy has been updated. By continuing to use our site, you consent to Arm's Privacy Policy. Please review our Privacy Policy (/company/privacy) to learn more about our collection, use and transfers of your data.

Accept and hide this message important information

This site uses cookies to store information on your computer. By continuing to use our site, you consent to our cookies (/company/cookiepolicy).

Don't show this message again

Change Settings (/company/cookiesettings/)

Hardware & Collateral

ULINK Debug Adaptors (/ulink/)

Evaluation Boards (/boards2/)

Product Brochures (/product/brochures.asp)

Device Database (/dd2/)

Distributors (/distis/)

Downloads (/download/)

MDK-Arm (/demo/eval/arm.htm)

C51 (/demo/eval/c51.htm)

C166 (/demo/eval/c166.htm)

C251 (/demo/eval/c251.htm)

File downloads (/download/file/)

Support (/support/)

Knowledgebase (/support/knowledgebase.asp)

Discussion Forum (/forum/)

Product Manuals (/support/man/)

Application Notes (/appnotes/)

Contact

Distributors (/distis/)

Request a Quote (/product/prices.asp)

Sales Contacts (/company/contact/)

Cookie Settings (/company/cookiesettings) | Terms of Use (/company/terms) | Privacy (/company/privacy) | Accessibility (/company/accessibility) | Trademarks (https://www.arm.com/company/policies/trademarks) | Contact Us (/company/contact/) | Feedback (/support/feedback.asp)

Copyright (/company/terms) © 2005-2019 Arm Limited (/company) (or its affiliates). All rights reserved.

ARM