AN12284

LPC55xx CoreMark 在 Cortex-M33 上的移植指南

版本 1 — 2019 年十二月

应用笔记

目录

1 简介

EEMBC 开发的 CoreMark 是一个简单且精准的基准测试，专门用于测试嵌入式微处理器内核的性能。运行 CoreMark 会产生一个得分，方便用户在不同的处理器之间进行比较。

LPC55xx 是一款基于 Arm®Cortex®-M33 的微控制器。该 MCU 片上外设包括：

- 一个 Arm®Cortex®-M33 协处理器
- CASPER 加密/FFT 引擎
- 用于 DSP 功能的 PowerQuad 硬件加速器
- 高达 320 KB 的片上 SRAM，高达 640KB 的片上闪存
- PRINCE 模块，用于实时片上闪存加密/解密
- 高速和全速 USB 主机和设备接口
- 可实现全速 SDIO/MMC
- 五个通用定时器（MRT）
- 一个 SCTimer/PWM
- 一个 RTC/警报定时器
- 一个 24 位多速率计时器（MRT）
- 窗口看门狗定时器（WWDT）
- 九个灵活可配的串行通信外围设备（可以配置为 USART，SPI，高速 SPI，I2C 或 I2S 接口）
- 可编程逻辑单元（PLU）
- 一个 16 位 1.0Msamples/sec ADC
- 比较器和温度传感器

与 Cortex-M4 内核的微处理器相比，Cortex-M33 在相同的条件下性能提高了 18.2%，同时提高了电源效率。Cortex-M33 官方 CoreMark 为 4.02 CoreMark/MHz，Cortex-M4 官方 CoreMark 为 3.40 CoreMark/MHz。

本应用笔记描述了如何将 CoreMark 代码移植到 LPC55(s)xx，这涉及到软件和硬件的配置，包括内存分配，编译器设置和电路板配置。本笔记还描述了如何在 Cortex-M33 上测量 CoreMark 得分以及结果，包括 CoreMark 得分和以 µA/MHz 为单位的动态平均功耗。与此同时，还提供了基于不同软件开发工具（Keil MDK，IAR EWARM 和 MCUXpresso IDE）的 CoreMark 工程，以供参考。

2 将 CoreMark 库集成到 SDK2.0 框架

本应用笔记相关的软件包中包含了基于 SDK2.0 的项目框架。它可以帮助测试人员使用 CoerMark 库源代码快速建立对 LPC55(s)xx进行基准测试的工程。首先，请访问：http://www.eembc.org/coremark，单击 Download 链接，如 图 1 所示，然后按照页面上的说明进行操作。

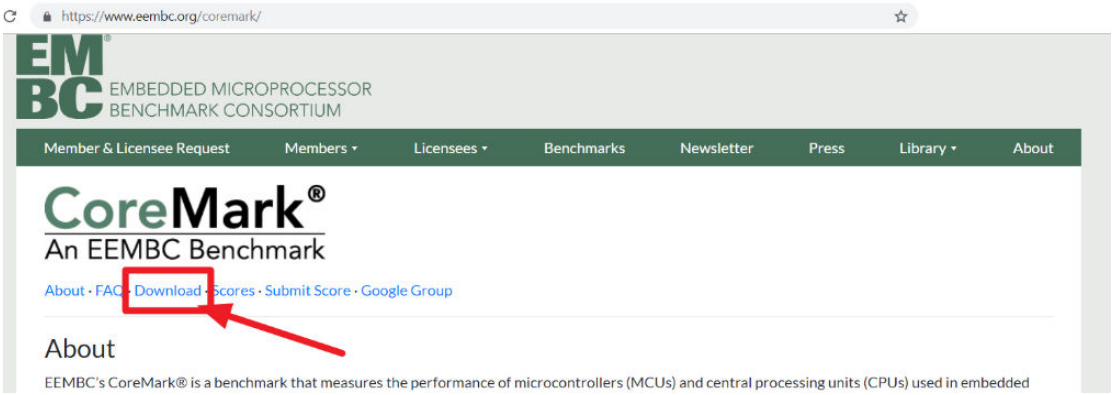

图 1. EEMBC CoreMark 下载链接网页

查看许可条款，请阅读自述文件和文档文件。自述文件提供有关解压缩和构建发行版的分步说明。该文档有助于熟悉整个应用笔记中使用的 CoreMark 术语。

2.1 将 CoreMark 库移植到 CoreMark 框架中

每个 IDE 工程包中有两种 CoreMark 项目测试设置。一个从片上闪存执行 CoreMark 应用程序，另一个从内部 SRAMX 执行 CoreMark 应用程序。

CoreMark 测试项目设置名称为：

1. `run_in_flash_xxmhz`：Cortex-M33 从片上 Flash 执行 CoreMark 测试。
2. `run_in_ramx_xxmhz`：Cortex-M33 从片上 Flash 执行 CoreMark 测试。

CoreMark 工程不同 IDE 的位置是：

- Keil MDK IDE：
  - lpc5500_coremark_mdk\coremark.uvprojx.eww
- IAR Workbench IDE：
  - lpc5500_coremark_iar\coremark.eww

每个工程配置中都有四个频率设置：12 MHz，48 MHz，96 MHz 和 150 MHz。

根据工具链的不同，项目配置介绍如下所示。CoreMark 框架需要从 EEMBC 网站上下载源代码并添加到 CoreMark 文件夹下。

2.1.1 适用于 Keil MDK/IAR EWARM/MCUXpresso IDE 的 CoreMark 框架

1. 必须先将 run_in_xxxx_xxmhz 项目设置成活动状态，然后才能添加 CoreMark 文件。

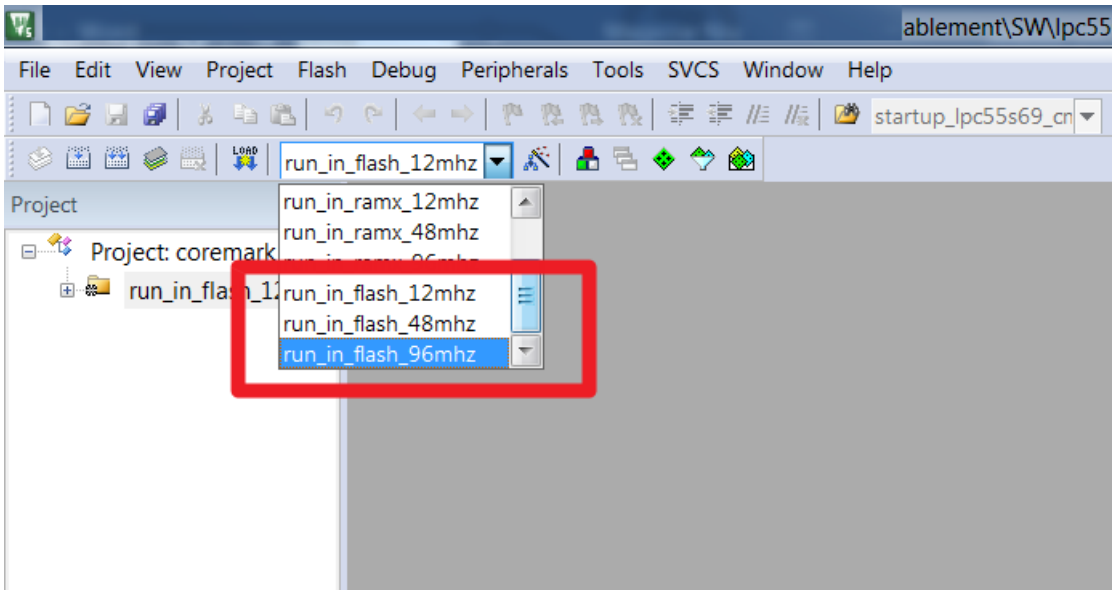

图 2. Keil MDK CoreMark 框架

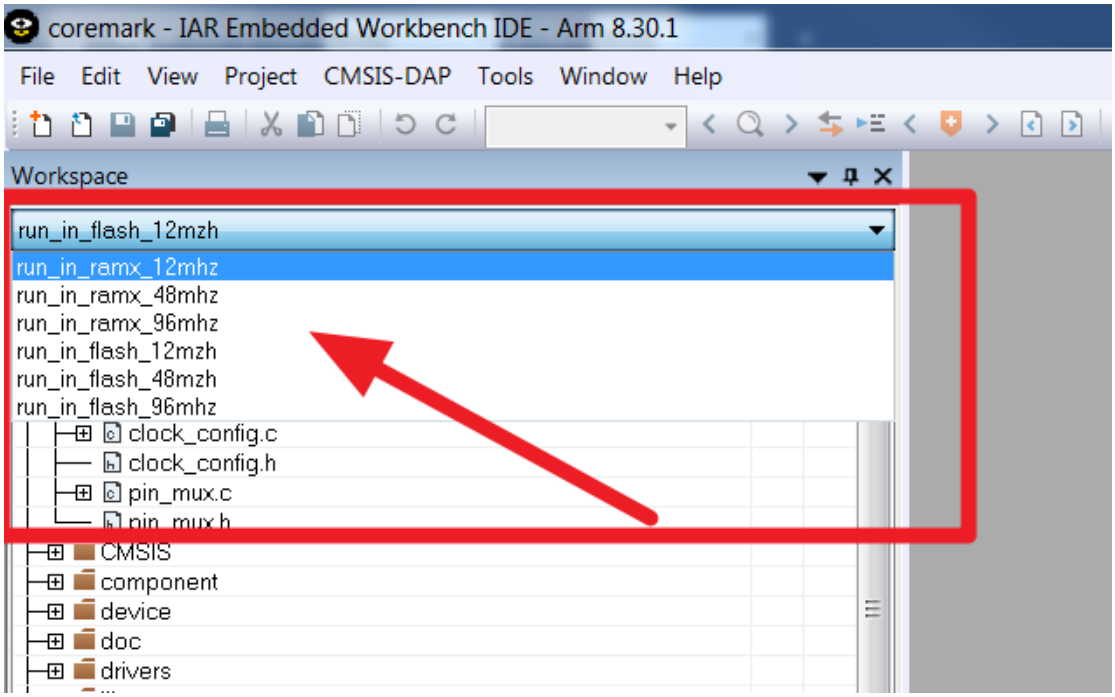

图 3. IAR EWARM 工作区

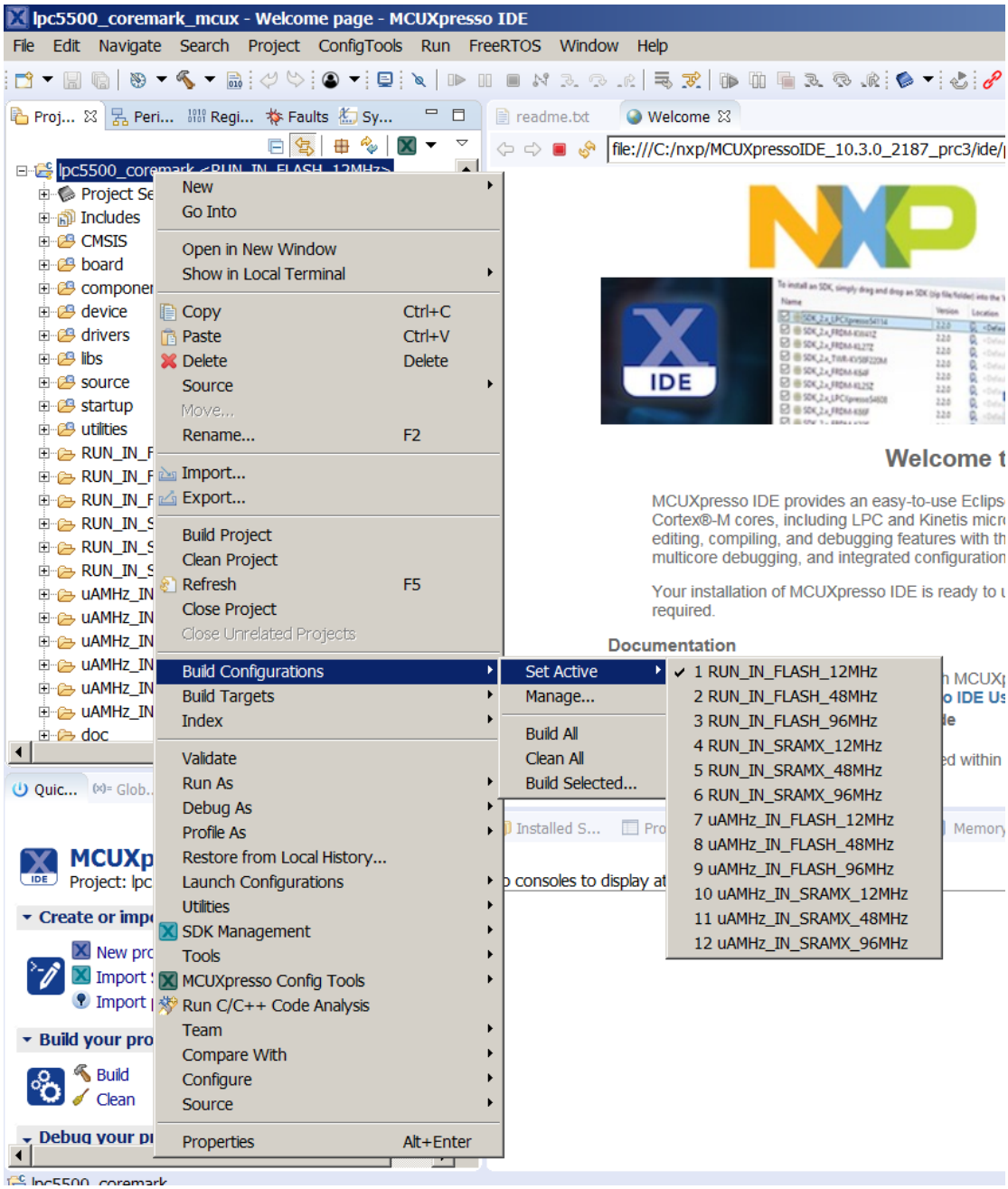

图 4. MCUXpresso 项目配置选择

2. 从在 EEMBC 中下载的 CoreMark 软件包中复制以下文件：

   - `core_list_join.c`
   - `core_main.c`
   - `core_matrix.c`
   - `core_state.c`
   - `core_util.c`
   - `coremark.h`

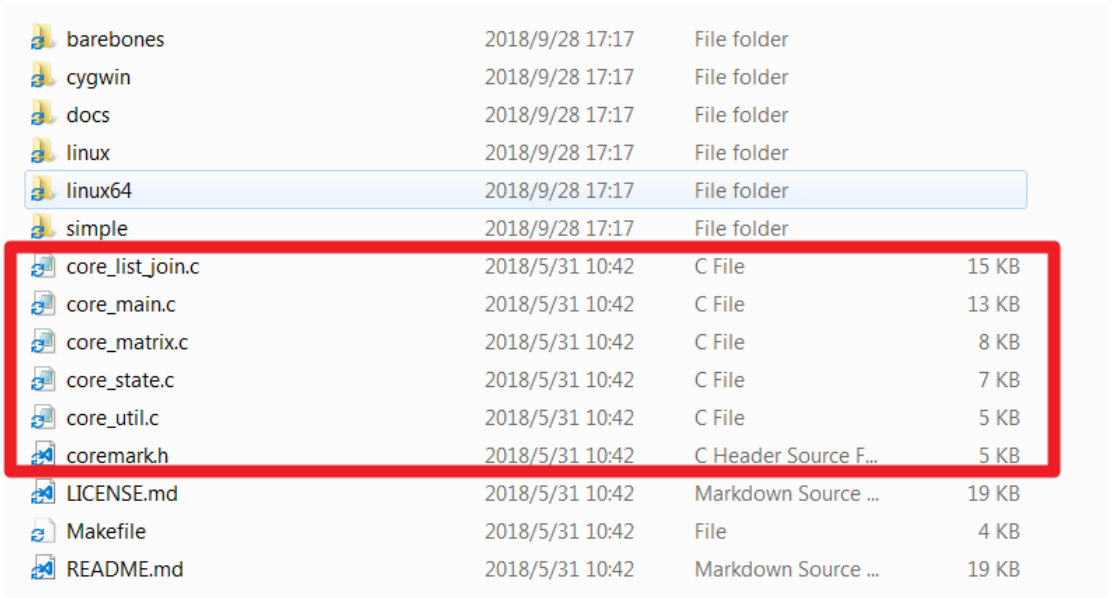

图 5. 要复制的 CoreMark 文件

- 对于 Keil MDK，将这些文件放在 *lpc5500_coremark_mdk\source* 目录中。
- 对于 IAR，将这些文件放在 *lpc5500_coremark_iar\source* 目录中。
- 对于 MCUXpresso，将这些文件放在 *lpc5500_coremark_mcux\source* 目录中。

3. 将文件 ee_printf.c，core_portme.c 和 core_portme.h（位于 port_lpc5500 文件夹下）复制到以下文件夹位置。
   - 对于 Keil IDE，将文件放在 *lpc5500_coremark_mdk\source* 中。

     通过双击组将文件添加到 Keil MDK 项目框架中的文件目录中。

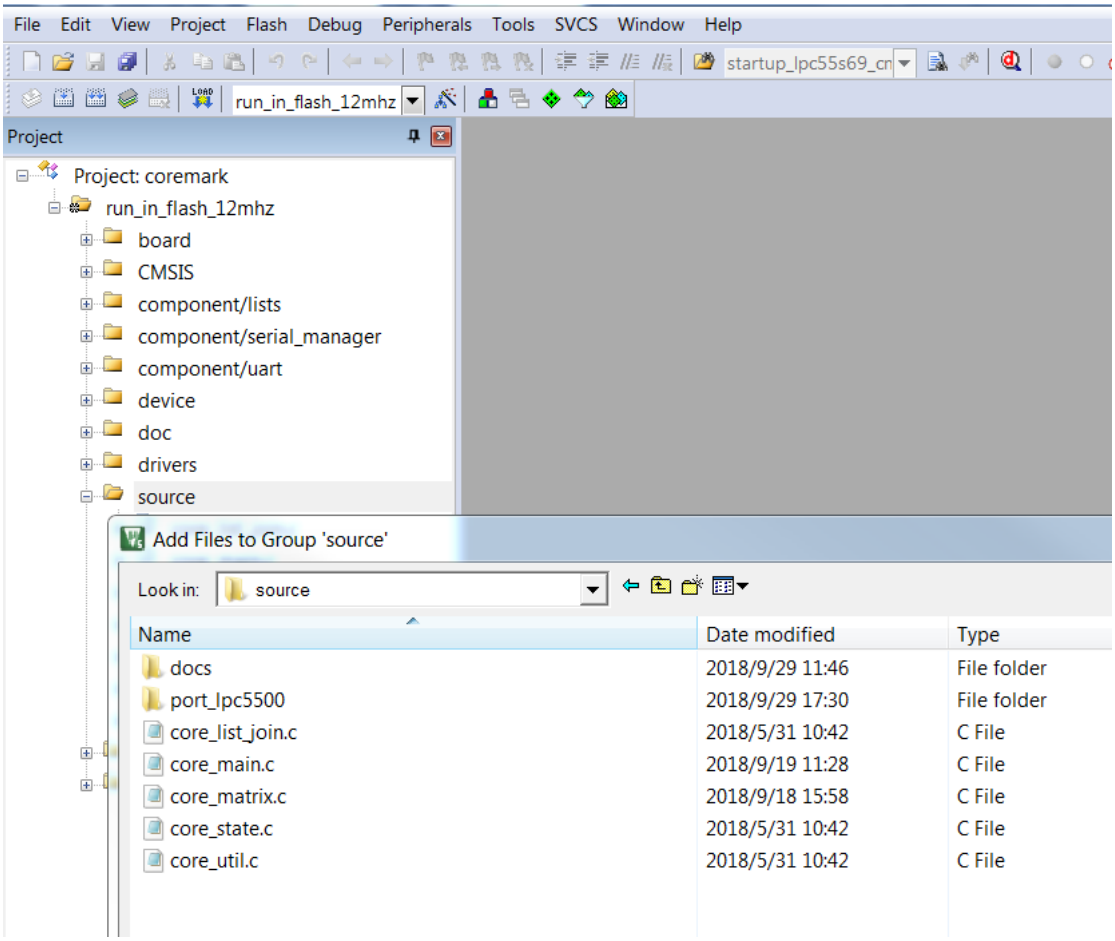

图 6. 在 Keil MDK 中添加文件

- 对于 IAR 嵌入式工作台，将文件放在 *lpc5500_coremark_iar\source* 中。

  双击文件组，将文件添加到 IAR 项目框架中的相应组源中。

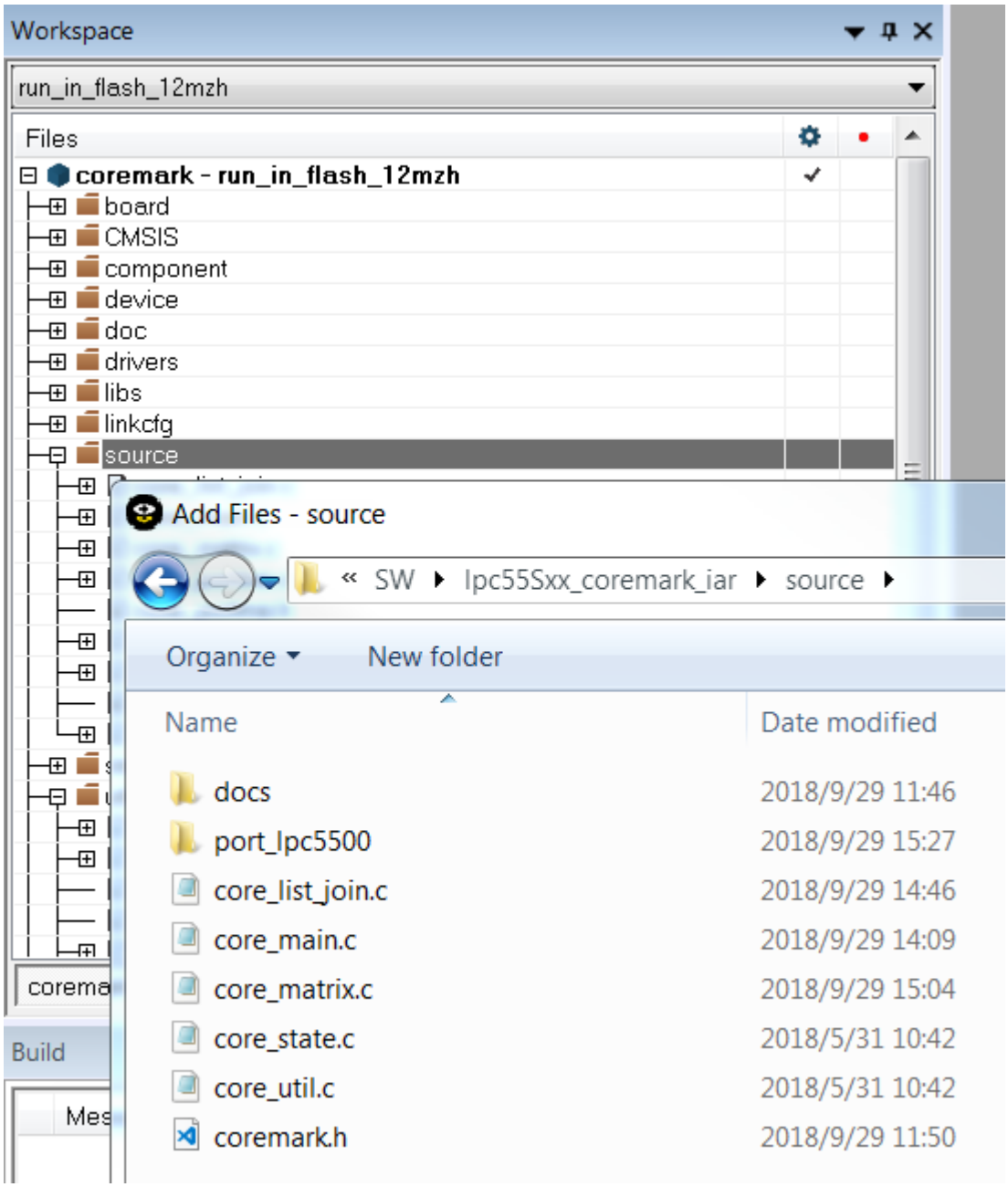

图 7. 在 IAR EWARM 工作区中添加文件

- 对于 MCUXpresso 将文件放在 *lpc5500_coremark_mcux\source* 中。

  单击 Refresh，将文件添加到 MCUXpresso 项目框架中的各个组源中。

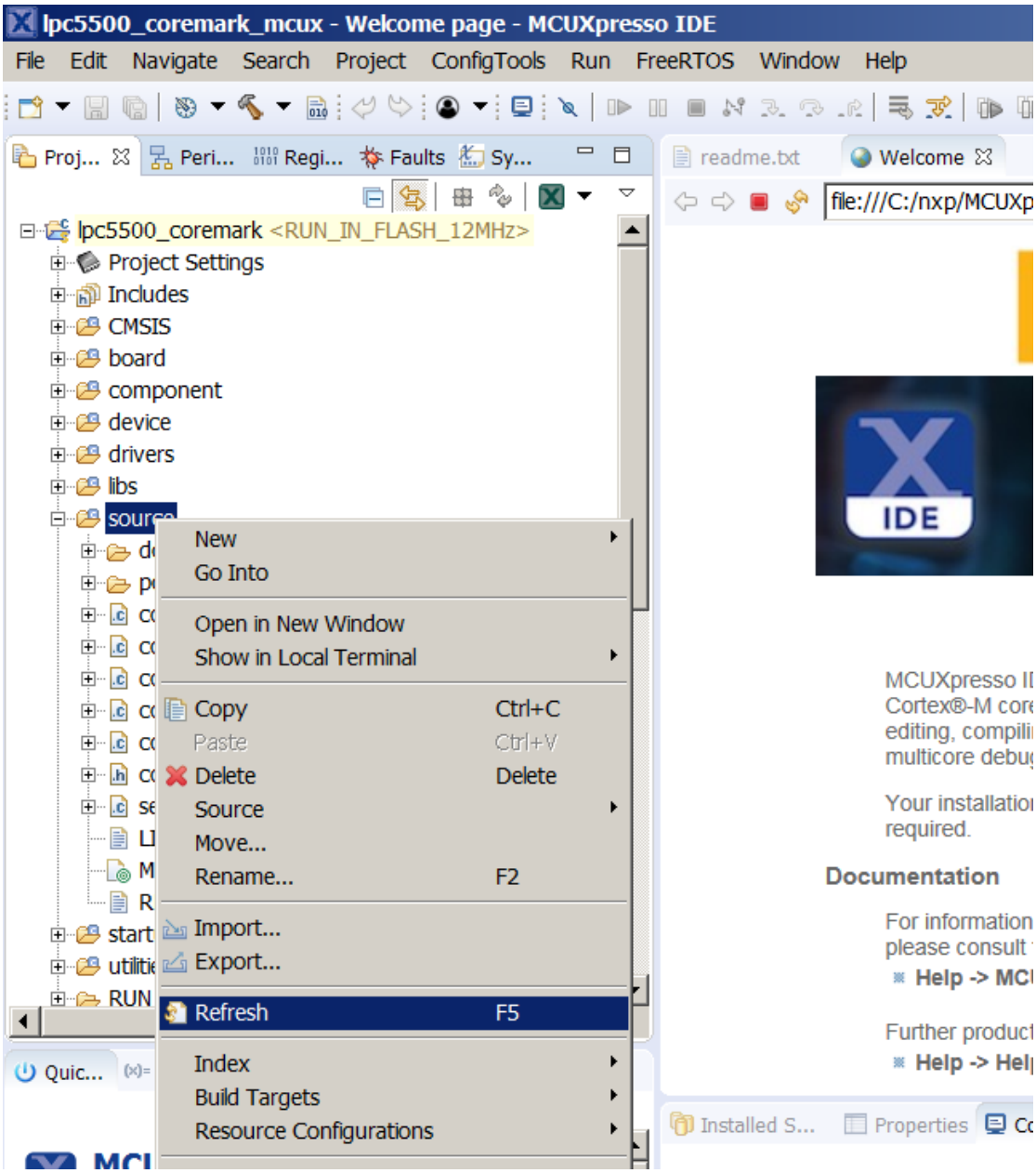

图 8. 在 MCUXpresso 项目中添加文件

请使用应用笔记附带的 core_portme.c 和 core_portme.h 文件，而不要使用 EEMBC CoreMark 软件包中的文件。为了方便起见，这些文件已准备好所需的端口更改。

将这些文件复制到所有三个工具链的源文件夹中，并在源框架下的项目框架中添加 core_portme.c 文件。

添加完所有文件后，工作空间应如 图 9，图 10，及 图 11 所示：

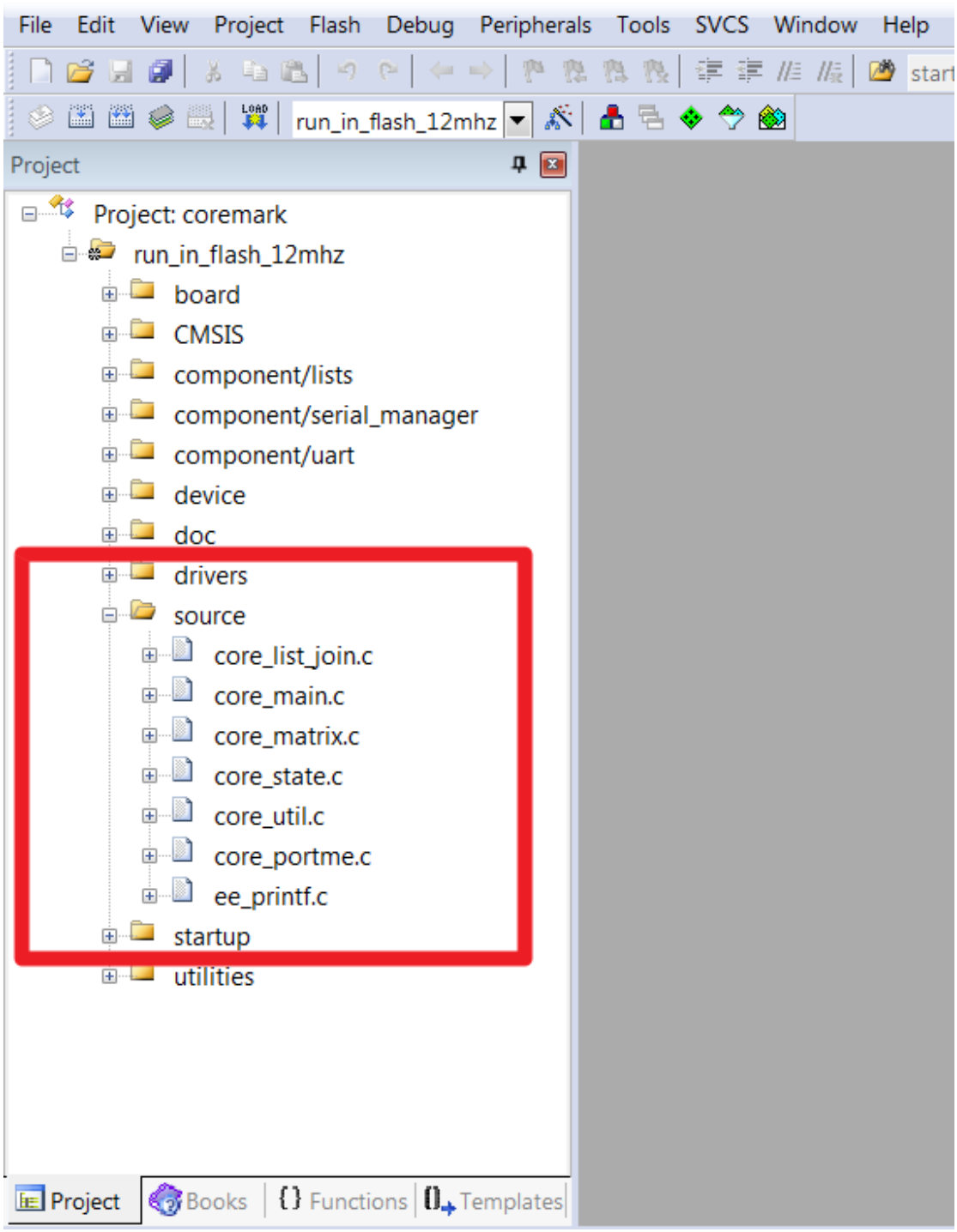

图 9. 添加 CoreMark 文件后的 Keil MDK 项目工作区

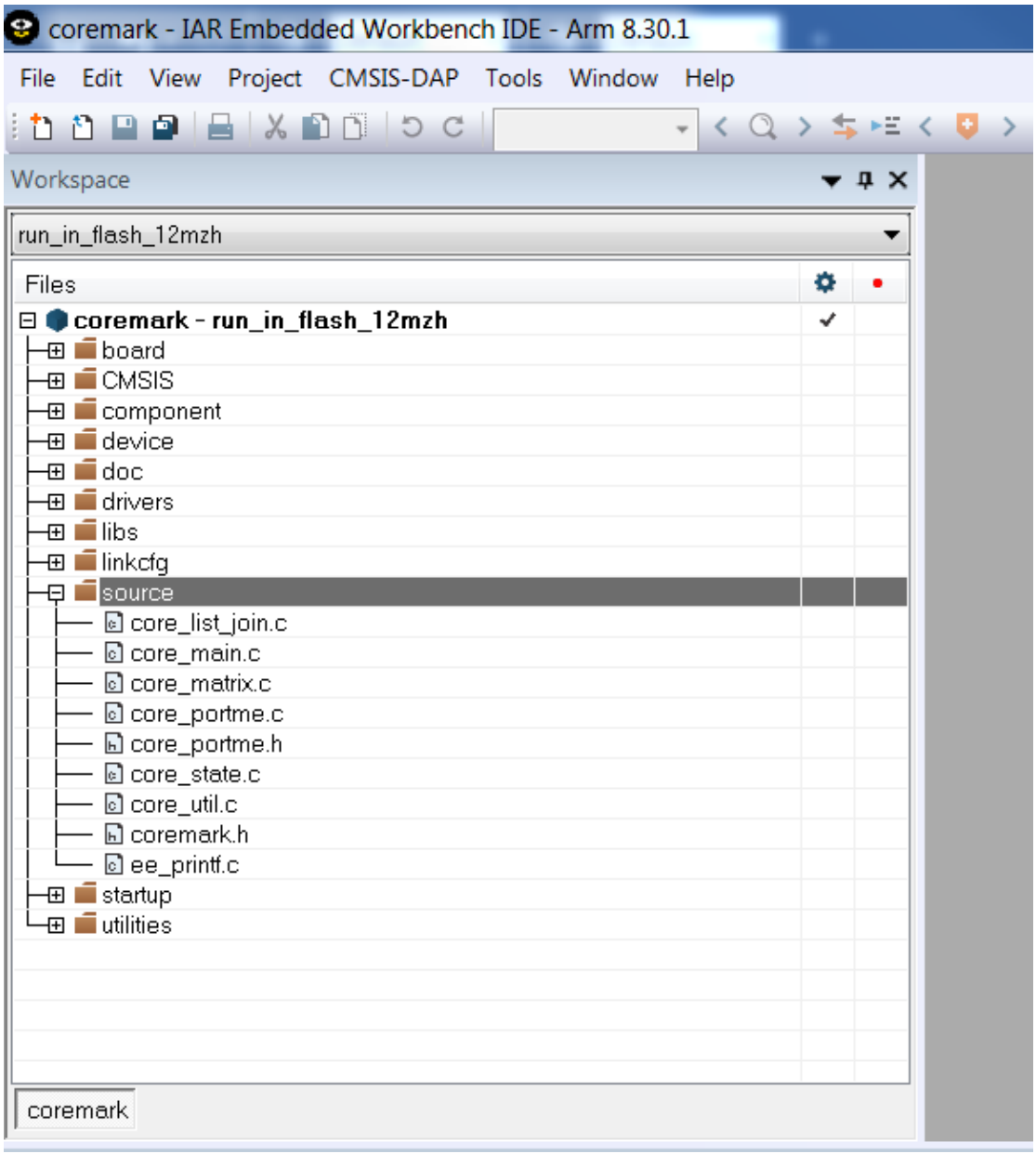

图 10. 添加 CoreMark 文件之后的 IAR EWARM 工作区

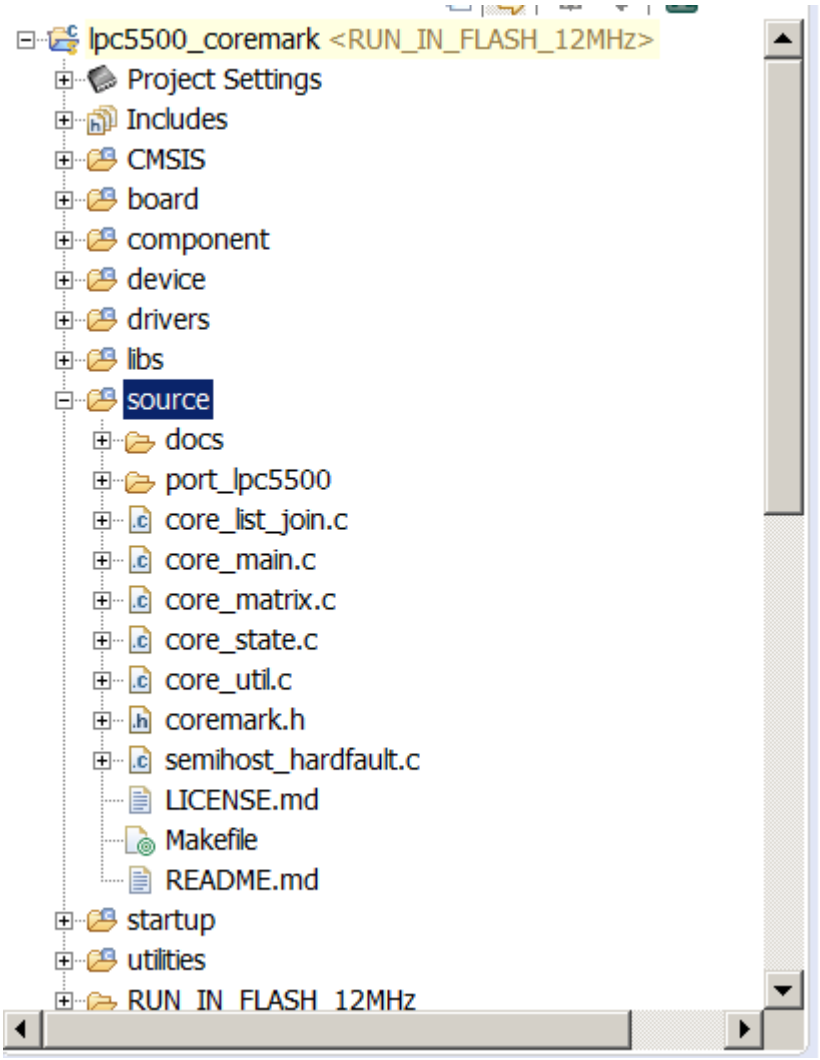

图 11. 添加 CoreMark 文件后的 MCUXpresso 项目

4. 需要修改一些文件以支持 CoreMark，并在下面进行描述。在项目分散文件中，将堆栈大小更改为 0x2000。

```
define symbol __size_cstack__ = 0x2000;
```

为了支持 PC 终端上的 printf 语句，需要修改 core_portme.h 文件。为 ee_printf 函数添加以下代码行。

```
#if HAS_PRINTF
#else
#ifdef COREMARK_SCORE_TEST
#define ee_printf printf
#else
extern int ee_printf_template(const char *fmt, ...);
#define ee_printf ee_printf_template
#endif
#endif
#endif
```

在 eeprintf.c 文件中，添加#ifdef COREMARK_SCORE_TEST 和函数 ee_printf(const char*fmt, …)。

```
#ifndef COREMARK_SCORE_TEST
int ee_printf_template(const char *fmt, ...)
{
return 0;
}
#endif
```

添加这个宏定义是为了在运行 µA/MHz 测试时优化 printf 代码。在“core_portme.h”中，有一个#define COREMARK_SCORE_TEST，用于指示应用程序是否正在执行 CoreMark 得分测试。

5. 将路径添加到项目中使用的头文件中：
   - 在 Keil MDK 中的 Project->C/C++(AC6)选项卡下，单击 Setup Compiler Include Paths，然后添加包含头文件的以下路径。

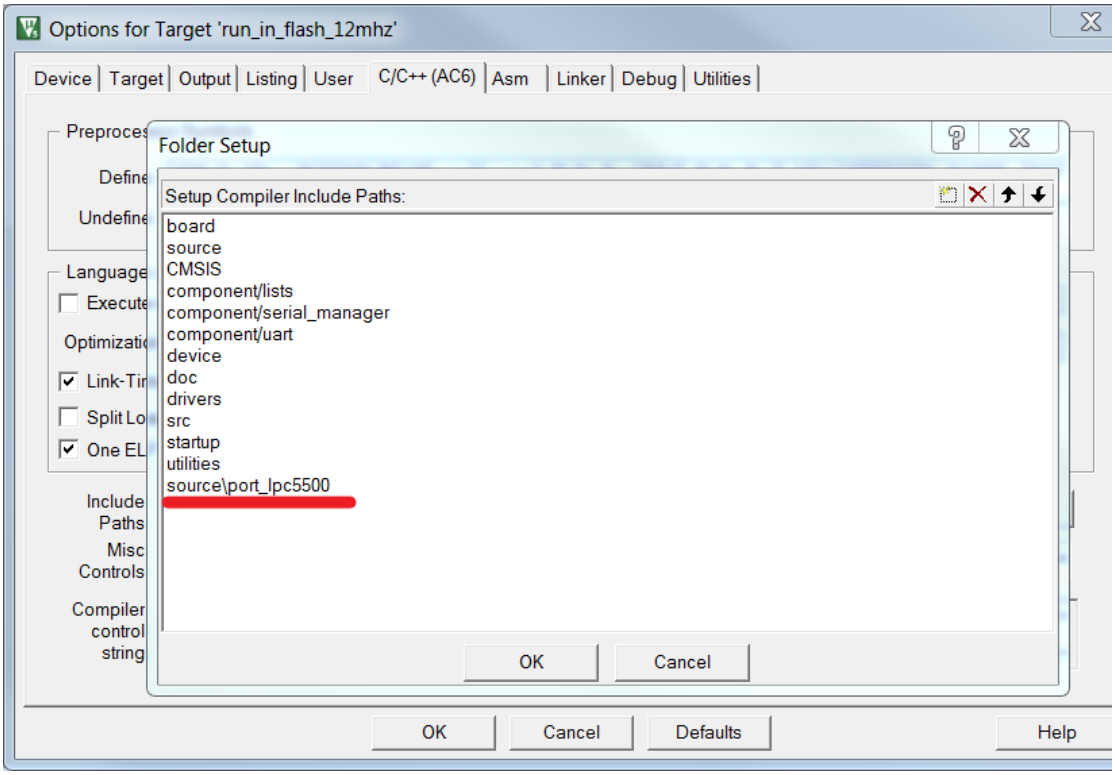

图 12. Keil MDK 编译器包含路径

   - 在 IAR 中，在 Project->Options->C/C++Compiler 下，单击 Preprocessor，然后添加包含头文件的以下路径。

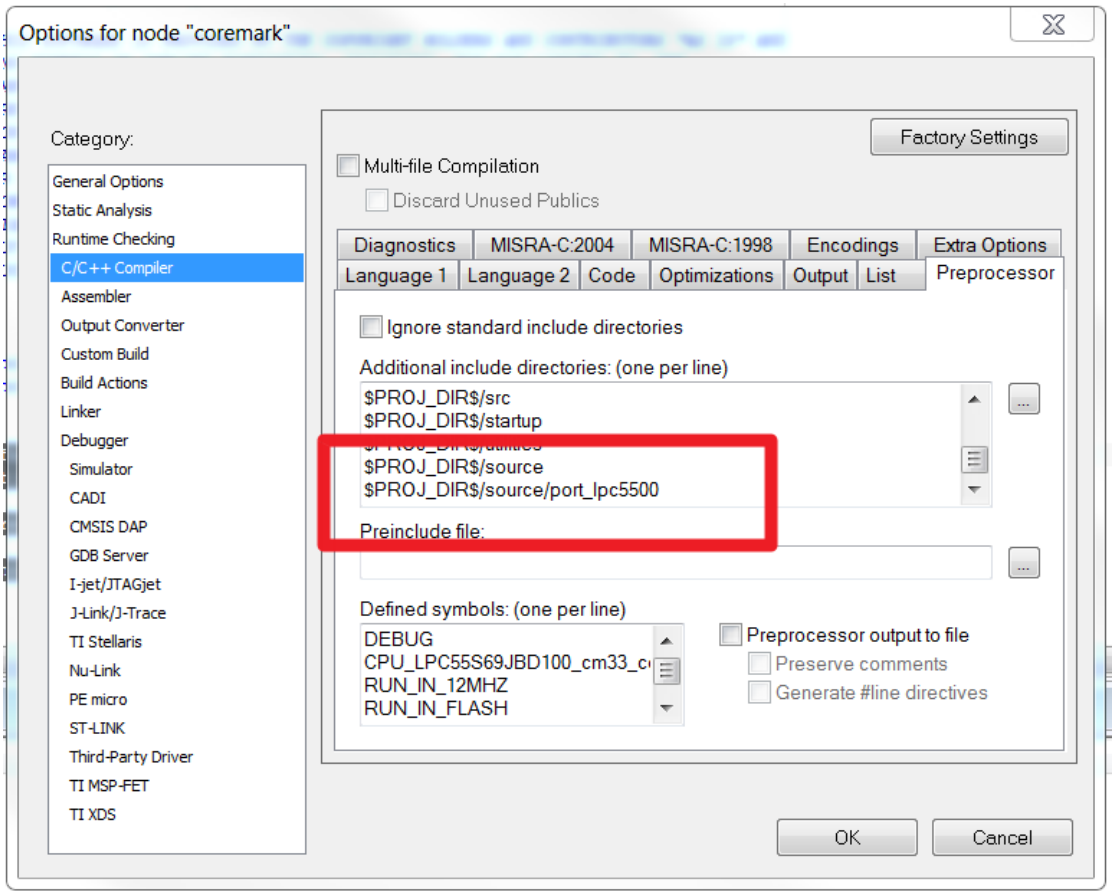

图 13. IAR EWARM 编译器包含路径

- 在 MCUXpresso 的 Properties for xxxx->C/C++Build->Settings 下，单击 Includes 并添加以下包含头文件的路径。

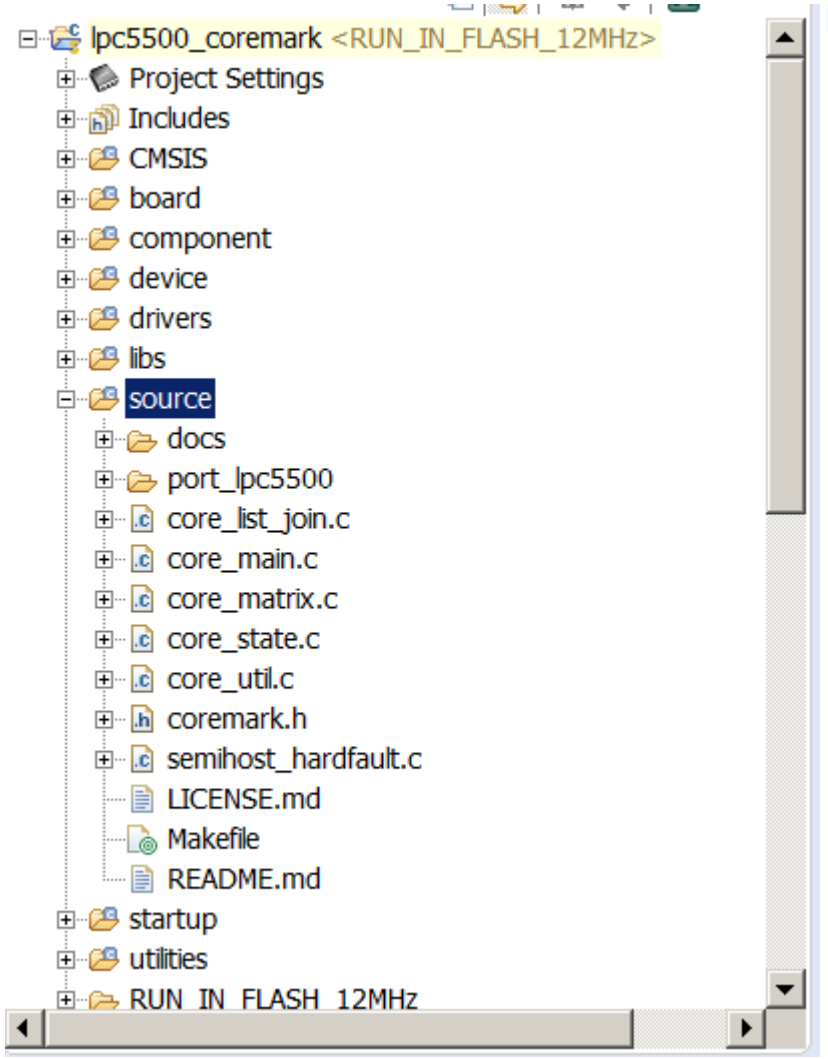

图 14. MCUXpresso 编译器包含路径

6. 现在，CoreMark 文件已被成功添加到 CoreMark 项目框架中。

2.1.2 从内部 SRAM 执行 CoreMark 框架

项目 run_in_ram_xxmhz 从 32 KB SARMX 内存区域执行 CoreMark 应用程序。

使用链接程序脚本重定位文件 core_list_join.c，core_main.c，core_matrix.c，core_state.c 和 core_util.c 以从 SARAMX 执行。

对于 Keil MDK，链接描述文件位于：

.\lpc5500_coremark_mdk\LPC55S69_cm33_core0_ramx.scf

图 15 显示了 run_in_rmax_xxmhz 项目的链接器脚本设置。

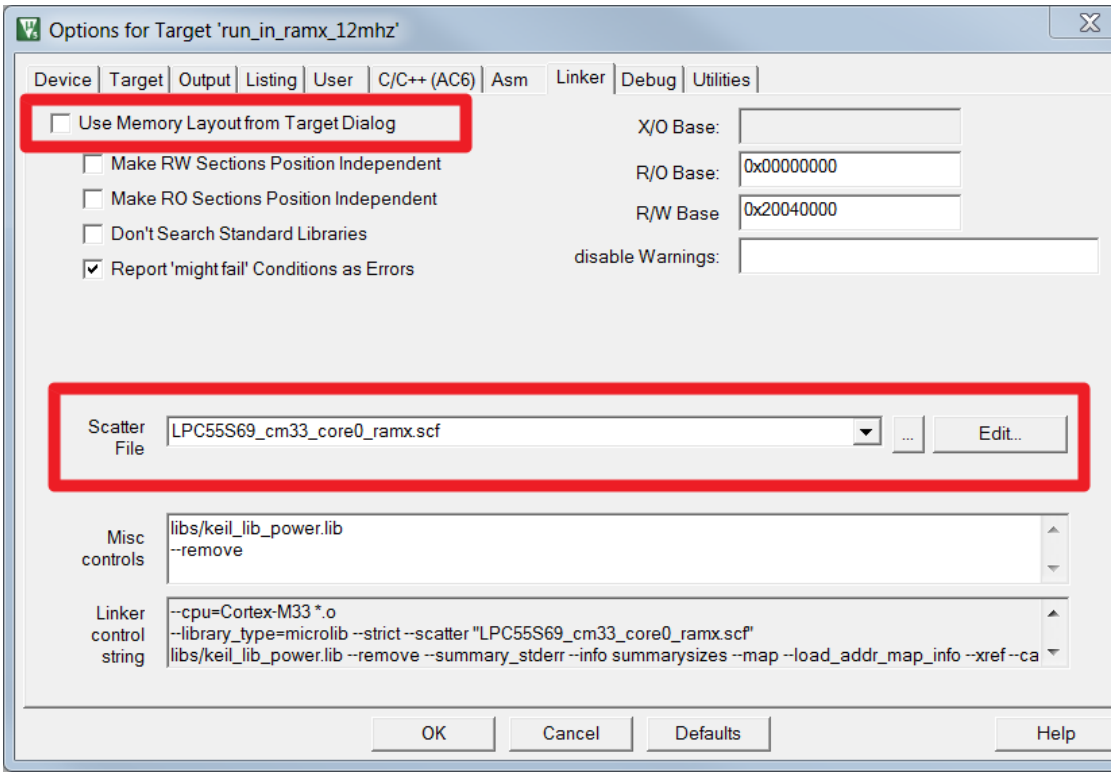

图 15. Keil IDE 中的链接描述

为了使 IAR EWARM IDE 在内部 SRAM 中执行 CoreMark，需要在文件 core_main.c，core_util，c，core_state.c，core_matrix.c和 core_list_join.c 中添加一行代码，如 图 18 所示，在所有五个文件中的#include 的上方。

这些 CoreMark 文件被标记为他们自己的 IAR EWARM 链接器 section。.\lpc5500_coremark_iar\LPC55S69_cm333_core0_ramx.icf 中提供的.icf 链接器文件。

然后将该部分（称为“ critical_text”）放入 SRAMX 内。 为了实现这个功能，需要在 icf 文件中添加以下代码行，如 图 16 所示。

```
initialize by copy { readwrite, section .textrw };
do not initialize  { section .noinit };

if (isdefinedsymbol(__USE_DLIB_PERTHREAD))
{
  /* Required in a multi-threaded application */
  initialize by copy with packing = none { section __DLIB_PERTHREAD };
}

place at address mem: m_interrupts_start    { readonly section .intvec };
place in TEXT_region                        { readonly };
place in DATA_region                        { block RW };
place in DATA_region                        { block ZI };
place in DATA_region                        { last block HEAP };
place in CSTACK_region                      { block CSTACK };

place in XCODE_region                       { section .critical_code };
initialize by copy                          { section .critical_code };
place in XCODE_region                       { rw object core_portme.o,
                                              rw object core_main.o,
                                              rw object core_list_join.o,
                                              rw object core_matrix.o,
                                              rw object core_state.o,
                                              rw object core_util.o,
                                            };
initialize by copy                          { object core_portme.o,
                                              object core_main.o,
                                              object core_list_join.o,
                                              object core_matrix.o,
                                              object core_state.o,
                                              object core_util.o,
                                            };
```

图 16. IAR EWARM 将代码分配给 SRAM 区域

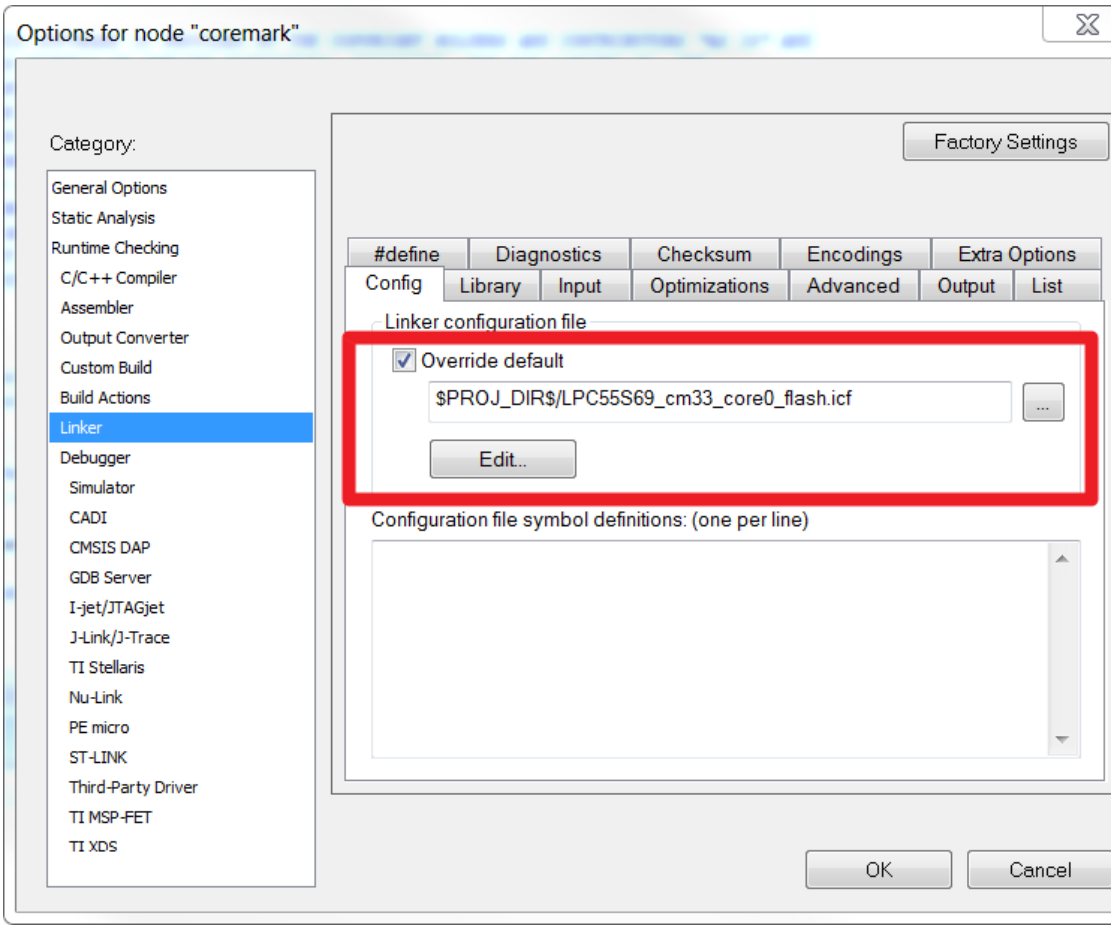

图 17. IAR EWARM 中的链接描述

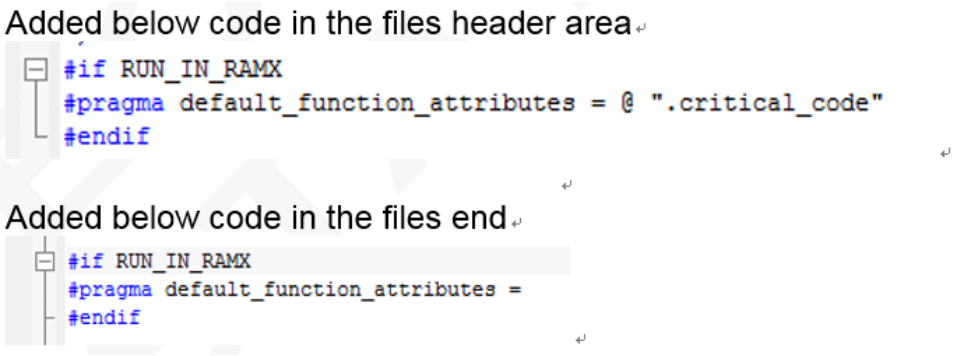

图 18. IAR EWARM # 编译指示

为了使 MCUXpresso 在内部 SRAM 中执行 CoreMark，只需在 Managed Linker Script 中选择名为lpc5500_coremark_RUN_IN_SARMX.ld 的链接器文件，如 图 19 所示。

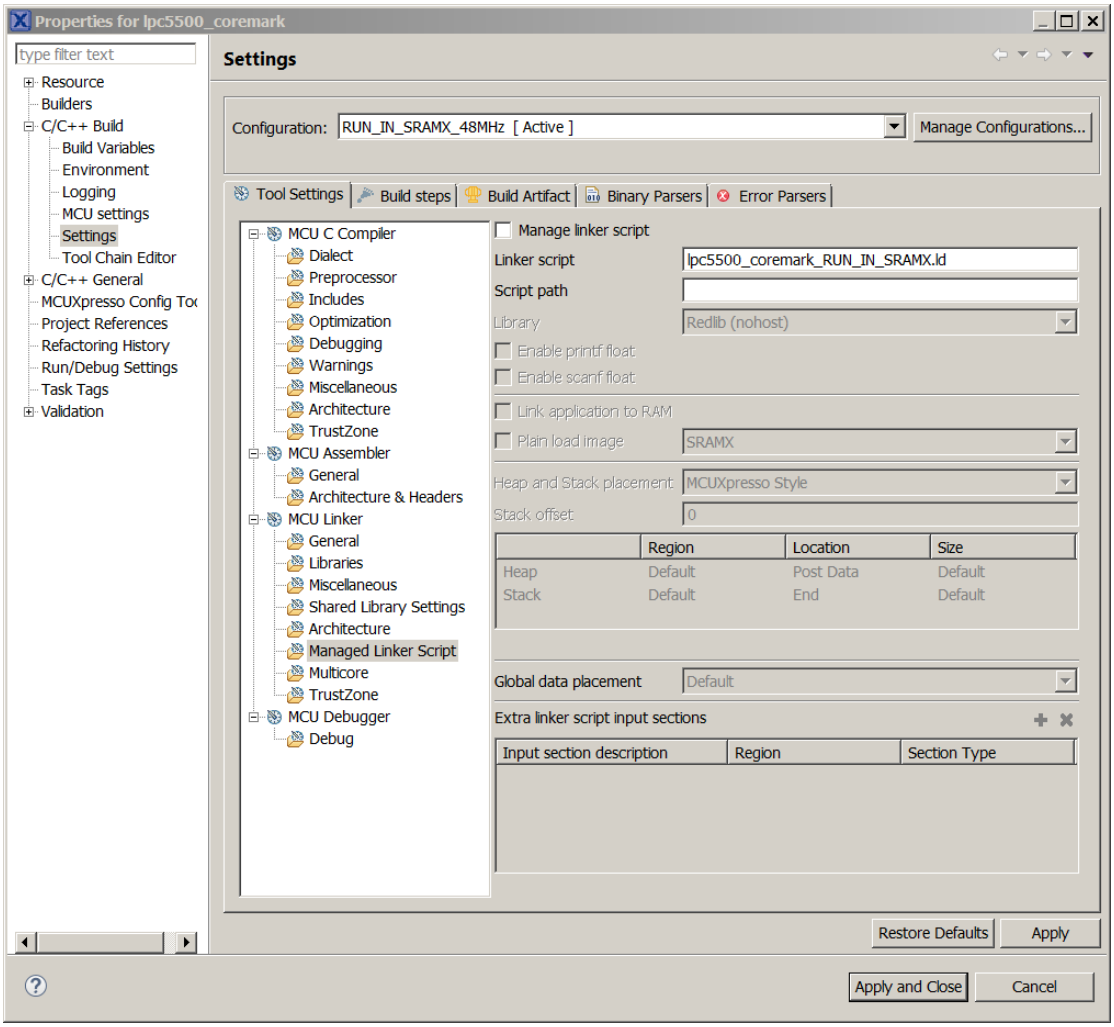

图 19. MCUXpresso 将代码分配给 SRAM 区域

2.2 优化 CoreMark 框架

有许多因素可以影响到 CoreMark 和 µA/MHz。其中一些因素是依赖 IDE 的优化，而其它因素则利用 MCU 架构来提高性能。本笔记的目的是能够从所有三个 IDE 中获得最佳分数，重要的是用户要了解 CoreMark 测试的结果会随着这些 IDE 版本的设计可能会有所变化。以下 IDE 版本是适用于本应用笔记的 IDE 版本：

- Keil MDK v5.28
- IAR EWARM 8.40.2
- MCUXpresso 11.0.1_2563

2.2.1 内存注意事项

由于 SRAM 和闪存的特定体系结构，CoreMark 运行在 SRAM 时执行速度更快。LPC55xx 内部存储器使用多层 AHB 矩阵系统，可为 Cortex-M33 和 SRAMX bank 提供独立的指令和数据总线。参见 图 20。SRAM0 至 SRAM4 在系统总线上。将 CoreMark代码和数据放在不同的 SRAM 中，多存储区分块可最大程度地减少总线竞争，改善指令和数据并行性。

重要的是根据 MCU 频率来优化闪存等待周期，以优化 CoreMark 最终的得分。相反，在执行 µA/ MHz 测试时，可以通过禁用闪存的预取功能来节省功耗。LPC55xx 用户手册包含有关配置闪存的更多信息，例如在给定核心频率下 Flash 允许等待的最小周期数。

本笔记提供的 CoreMark 框架项目包括实现各种内存优化的基于 SARM 和闪存都是单独项目。

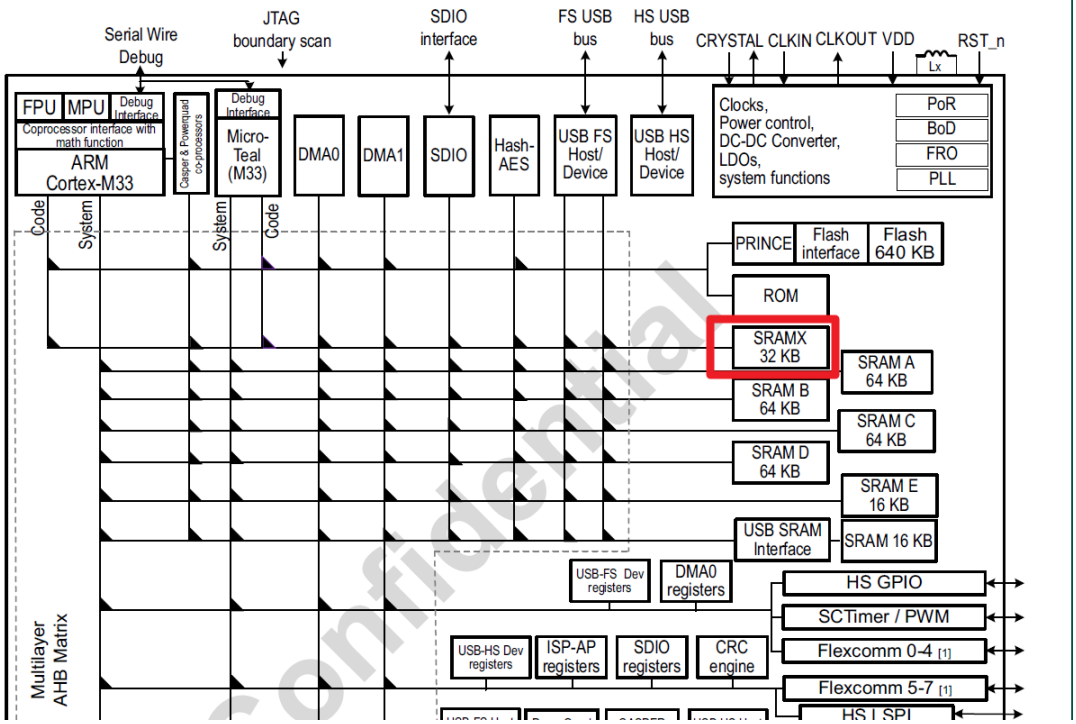

图 20. LPC55Sxx AHB 矩阵

在 SRAM 和闪存项目中，在 core_portme.h 中定义了一个 COREMARK_SCORE_TEST 宏，该宏指示该项目是否配置为执行 CoreMark 基准测试或 µA/MHz 测试。如果定义了此宏，则将运行 CoreMark 得分测试。如果该宏被注释掉，则将运行 µA/ MHz 测试。使用此宏可以在两个项目配置之间切换。

2.2.2 IDE 优化设置

以下优化基于编译器，因此取决于 IDE。这些优化适用于基于 SRAM 和基于闪存的项目。

2.2.2.1 Keil 优化

有两种编译器优化可以提高 CoreMark。在 Project->Options 中的 C/C++（AC6）选项卡下，在 Misc Ctonrols 中需要将优化级别设置为“-mcpu=Cortex-m33 --target=arm-arm-none-eabi -Omax -g -mthumb - mfpu=fpv5-sp-d16 -mfloat-abi=hard -fno-common -ffp-mode=fast”。

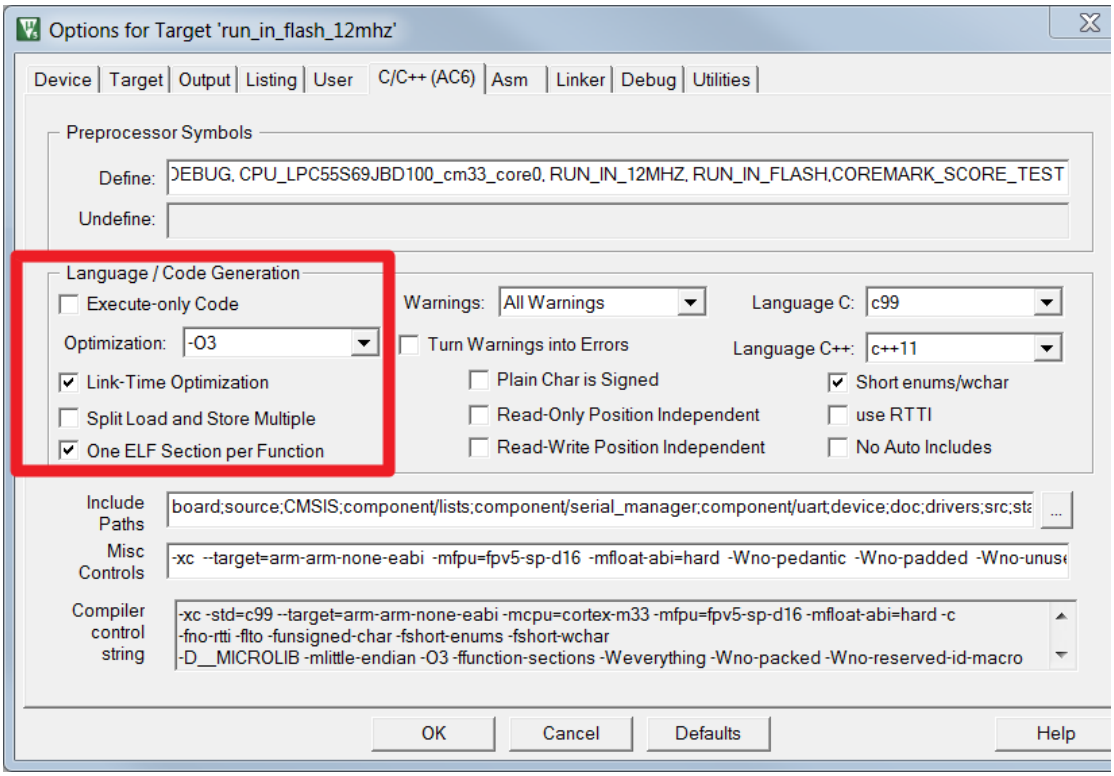

图 21. Keil MDK CoreMark 得分优化

在对 MCU 的功耗进行基准测试时，必须将优化设置调整为 0 级（-O0），并且必须取消选中 Link-Time Optimization。

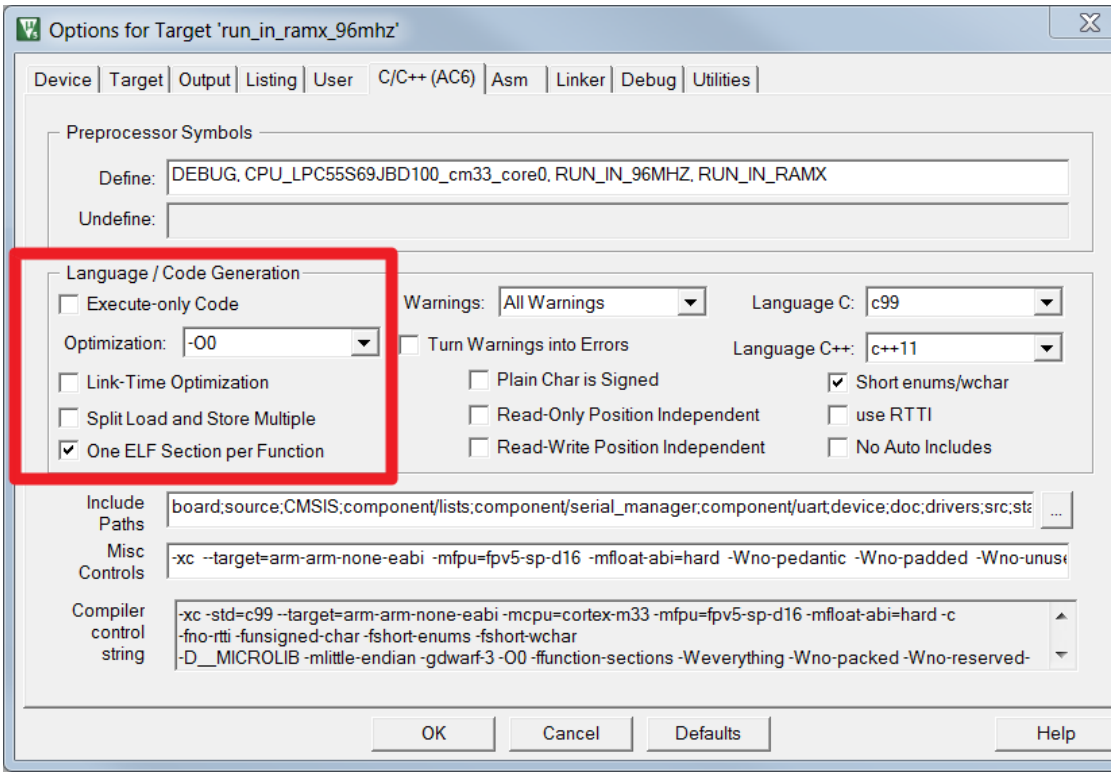

图 22. Keil MDK µA/MHz 优化

2.2.2.2 IAR 优化

有两种编译器优化可以提高 CoreMark。将优化级别设置为 High，从下拉菜单中选择 Speed，然后选中 No size constraints 复选框。

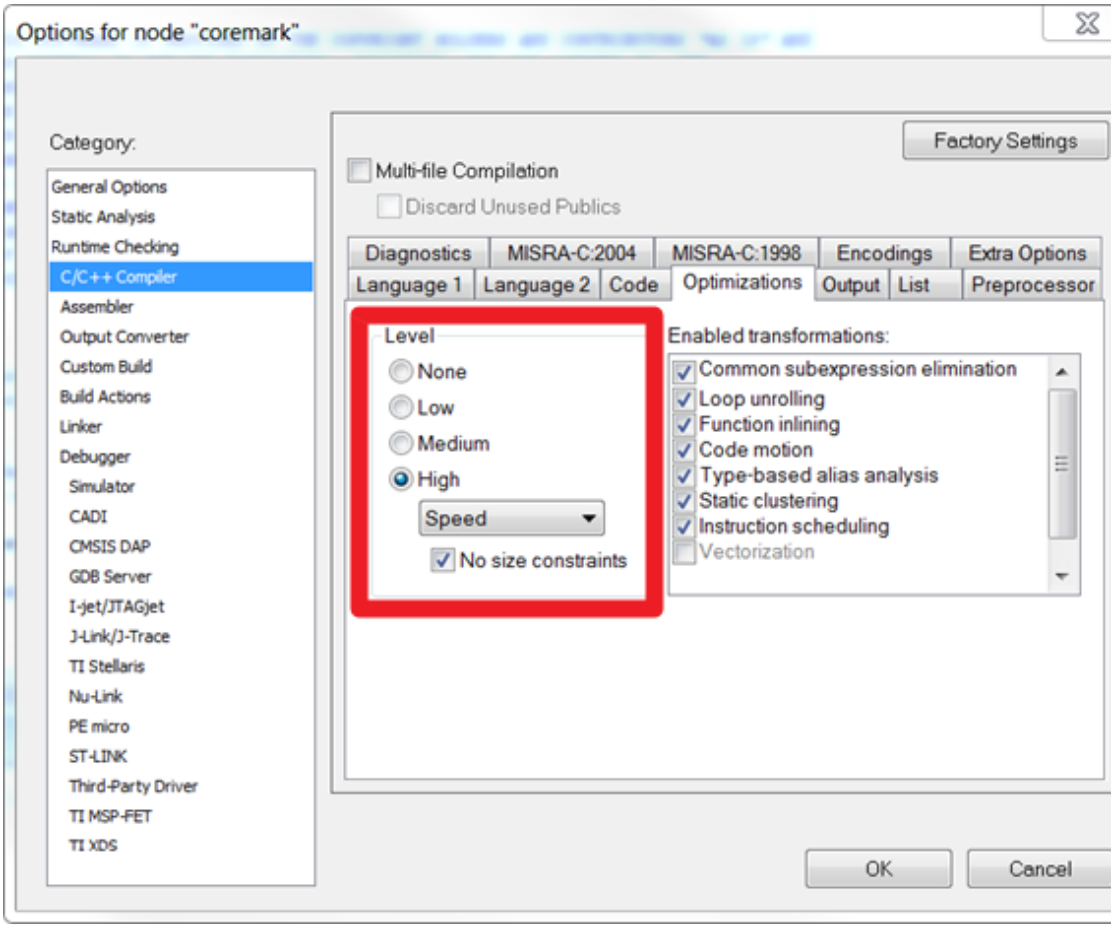

图 23. IAR EWARM CoreMark 得分优化

在对 MCU 的功耗进行基准测试时，应将优化级别设置为 None。

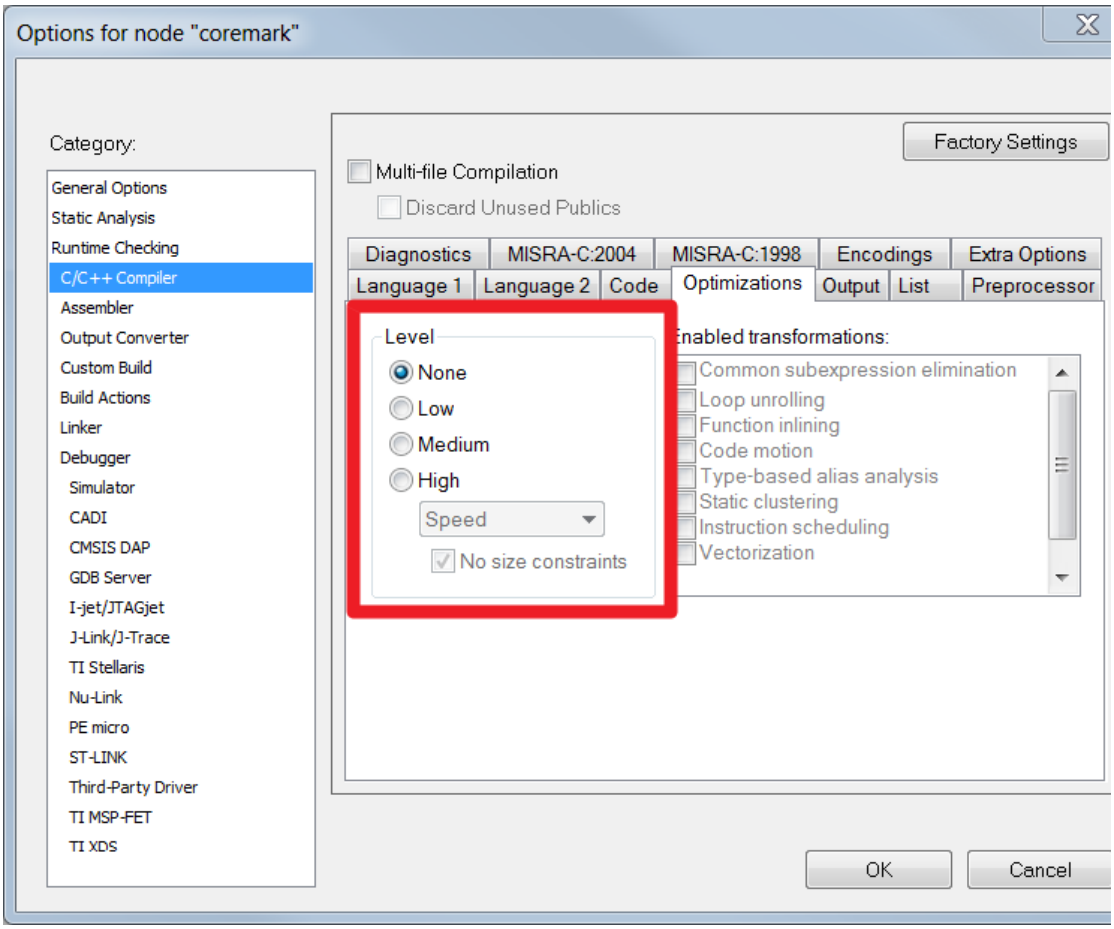

图 24. IAR EWARM μA/MHz 优化

2.2.2.3 MCUXpresso 优化

将优化级别设置为-O3，请从下拉菜单中选择 Optimize most (-O3)。

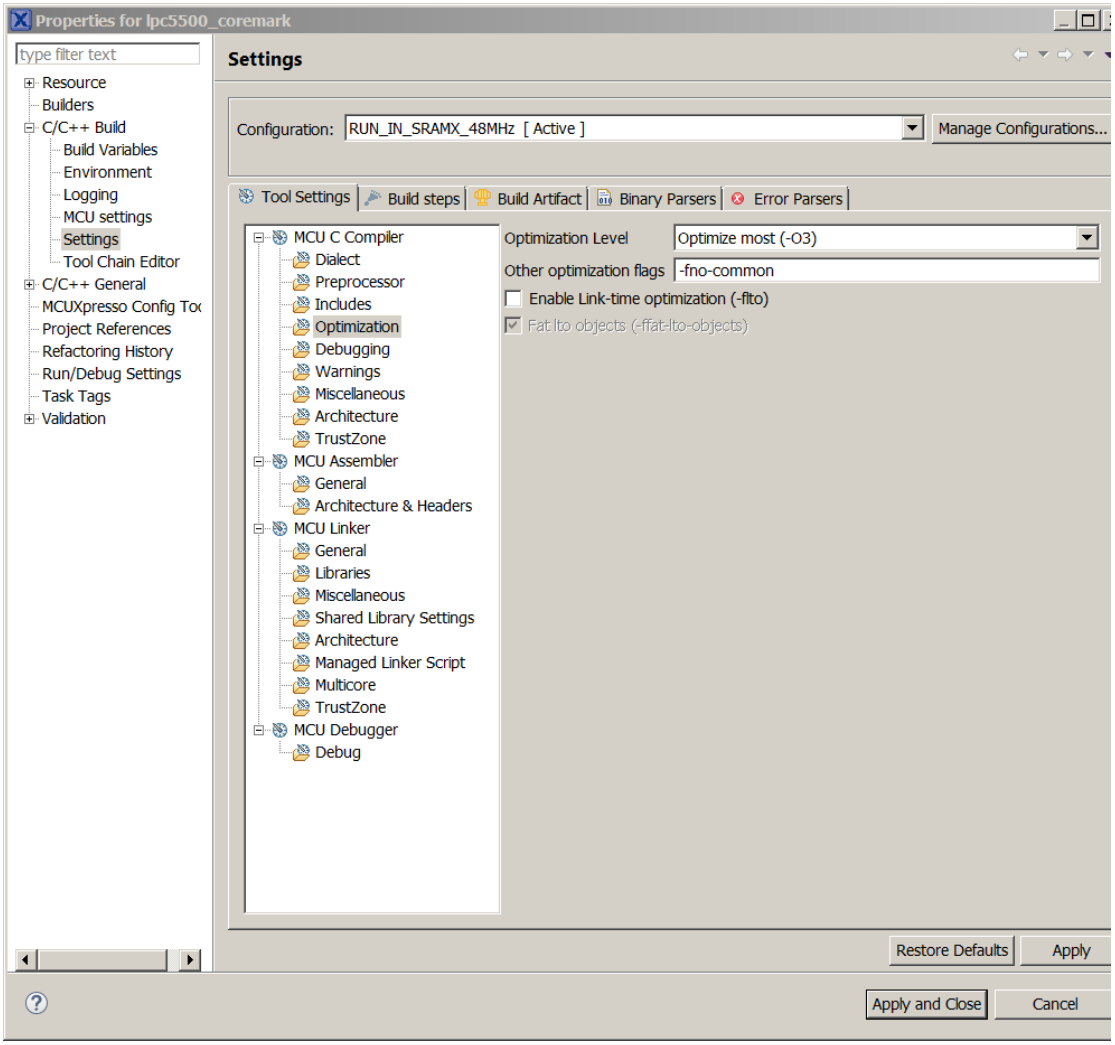

图 25. MCUXpresso CoreMark 得分优化

在对 MCU 的功耗进行基准测试时，优化级别应设置为 None(-O0)。

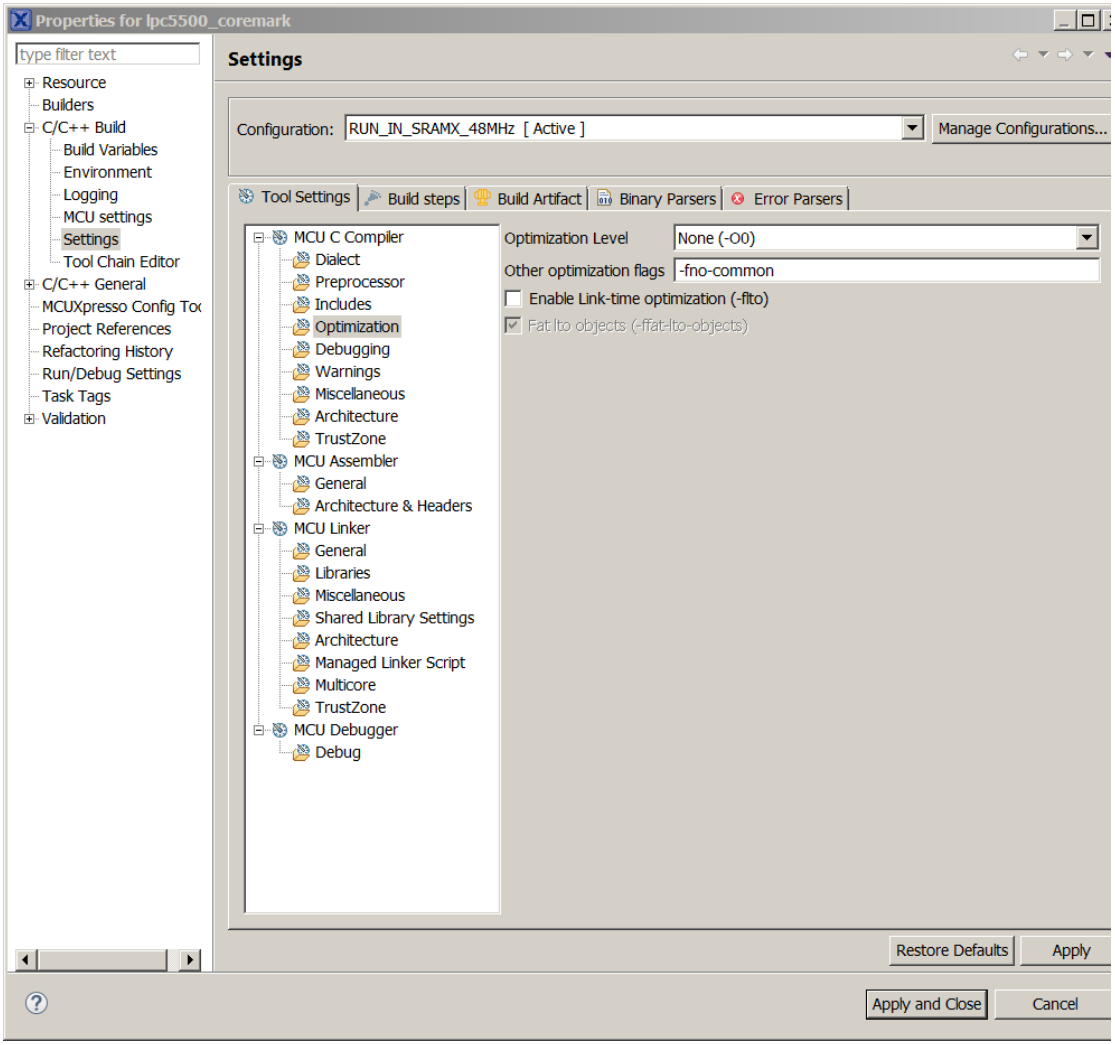

图 26. MCUXpresso µA/MHz 优化

3 在板子上测量 CoreMark

3.1 LPC55S69Xpresso 板

LPC55S69Xpresso 板通过 P6 支持 VCOM 串行端口连接。I 来观察来自板上的调试消息，请将串口终端设置为对应的 COM 端口，并使用设置 115200-8-N-1-none。为了使调试信息更易于阅读，新行接收设置应设置为自动。

3.2 电路板设置

LPC55S69 Rev A1 开发板用于 CoreMark 基准测试。

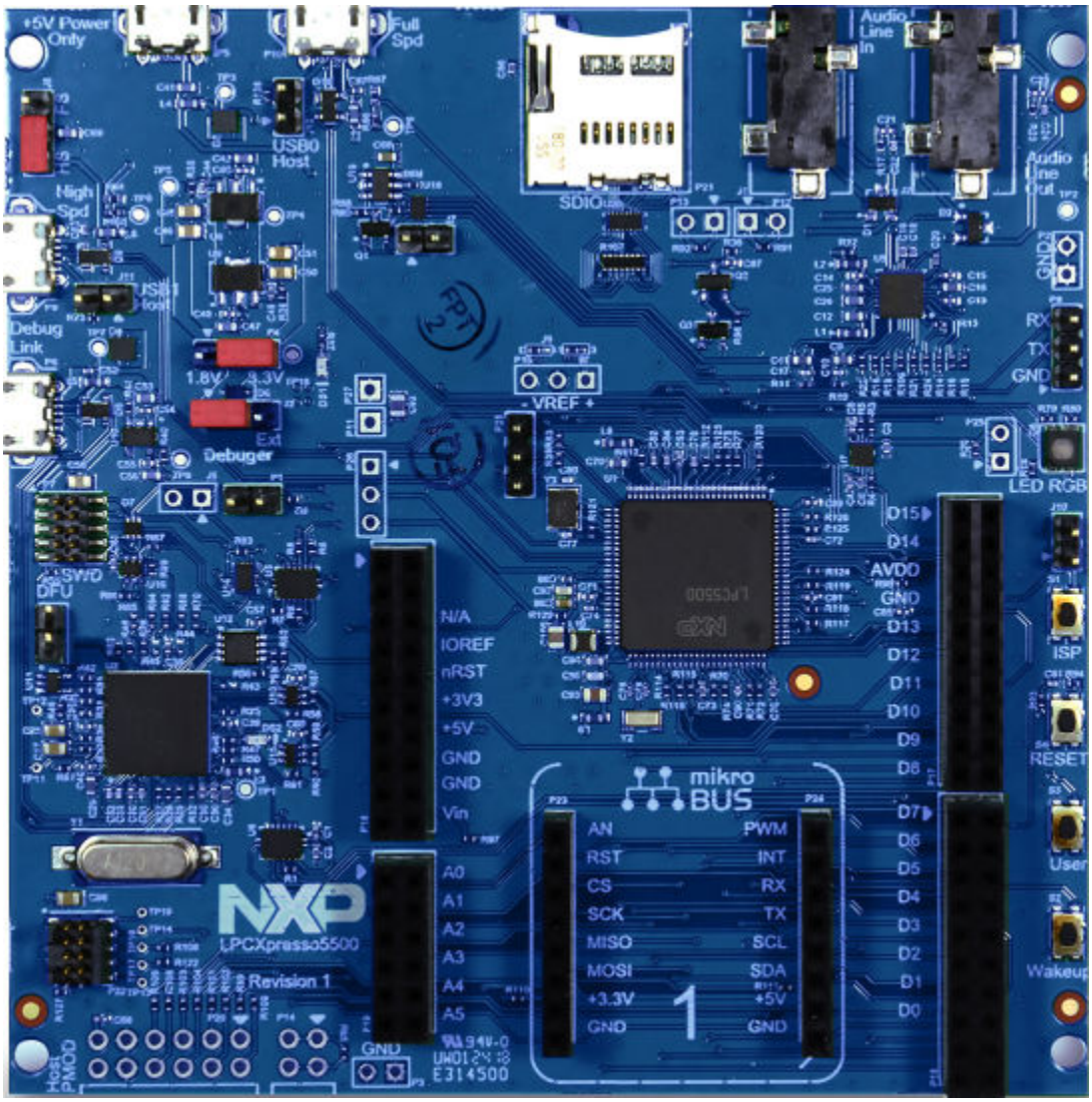

图 27. LPC55S69Xpresso 板

该评估板附带编程的 CMSIS-DAP 调试器。有关 CMSIS_DAP 调试固件的更多信息，请访问 LPC driver setup 以解答常见问题。有关调试和终端调试的说明，请将 USB 电缆连接到 P6 USB 连接器。电路板原理图可从 www.nxp.com 获得。

3.2.1 μA/MHz 测量设置

要测量 LPC5500 的动态功耗，请卸下 R92，在 P13 上安装的跳线帽，然后在 P13 上连接电流表，如 图 28 所示。

注意

EVK 上的当前数据可能略高于数据手册，因为 EVK 上具有很多其他的器件，会消耗更多电流。

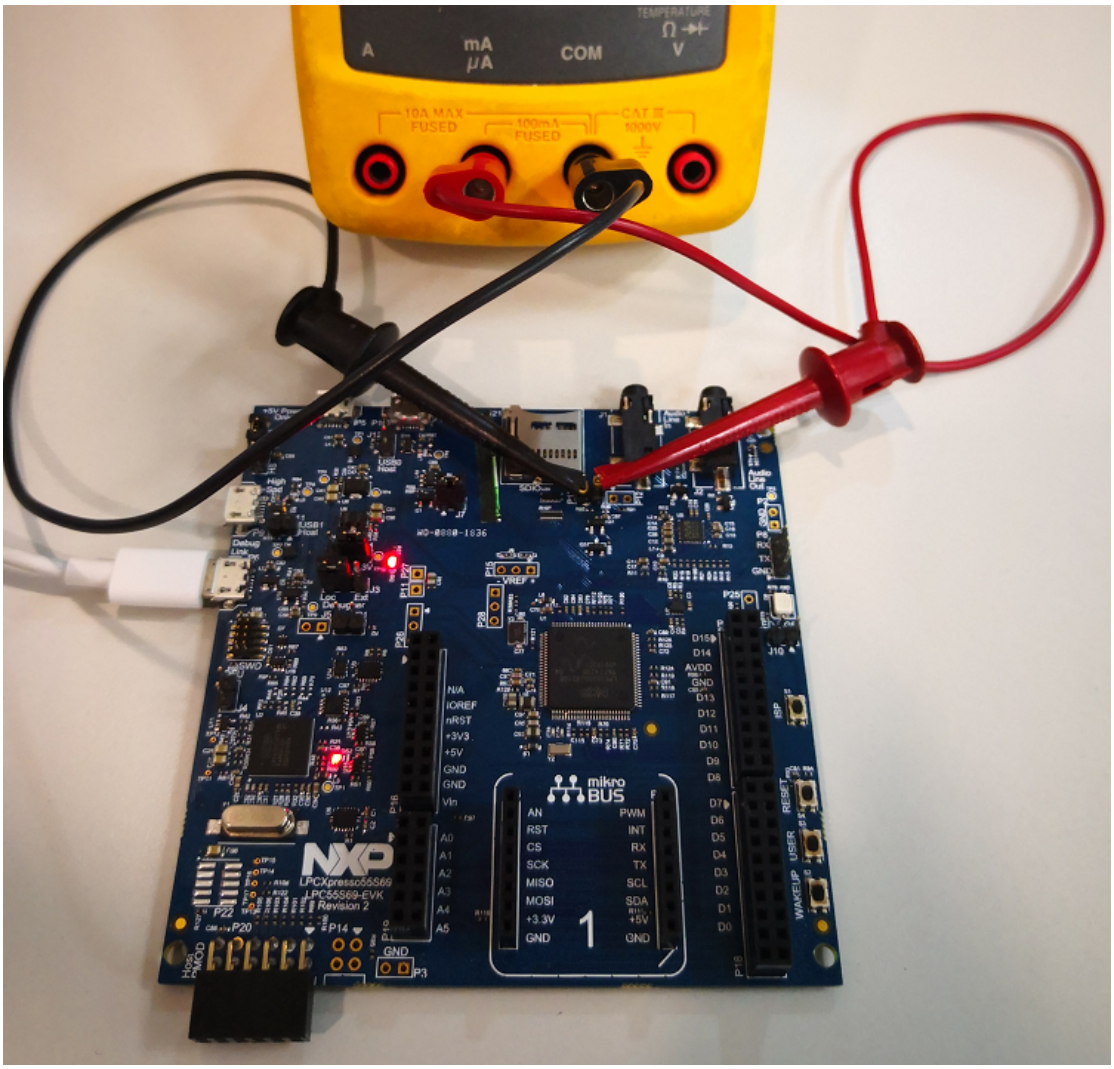

图 28. A/MHz 测量设置

如果需要测量 MCU 核心电流，则需要通过拆下 R92。然后我们可以用万用表测量流经 P13 的电流。

在执行 μA/MHz 基准测试时，请使用 P6 USB 连接器为电路板供电。在下载 μA/MHz 测试项目之后，通过拔下 USB 电缆重启电路板，然后重新插入以确保未连接调试探针。

调试接口的波特率设置为 115200。可以在 core_potme.c 文件中进行更改。

Line209 config.baudRate_Bps = 115200;

同样，通过在工作区窗口中选择不同的配置项目，可以更改核心时钟频率。每个配置都可以启用以下定义的项目配置：

- RUN_IN_12MHZ
- RUN_IN_48MHZ
- RUN_IN_96MHZ
- RUN_IN_150MHZ

3.3 运行 CoreMark 代码

获得 CoreMark 结果的第一步是将开发板的连接器 P6 与 PC 连接。然后，PC 就可以识别到 LPC-Link2 调试器，如 图 29 所示。

如果 PC 找不到串行端口驱动程序，请从下面的链接下载 LPCScrypt，然后安装在 PC 上。

http://www.nxp.com/support/developer-resources/software-development-tools/lpc-developer-resources-/lpc-microcontroller-utilities/lpcscrypt-v2.0.0:LPCSCRYPT?tab=Design_Tools_Tab

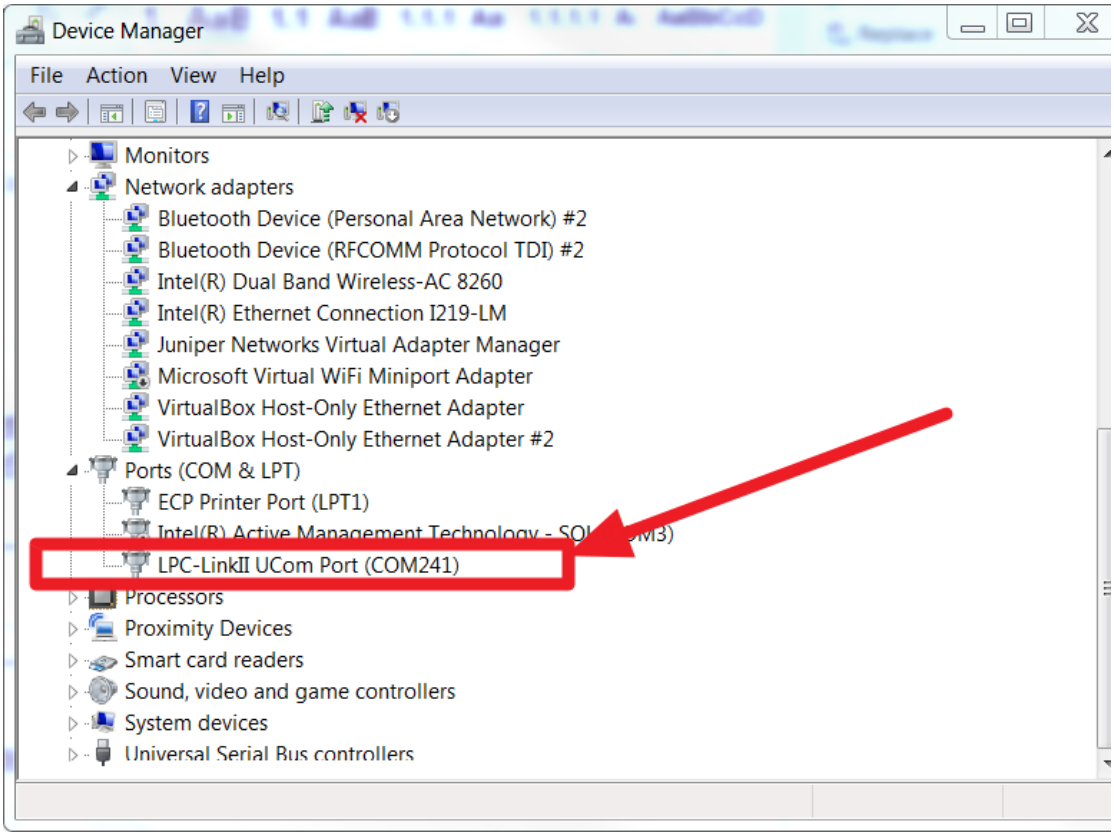

图 29. LPC-LinkII UCom 端口

打开一个 UART 调试终端（如 Tera Term， Putty 等），并将其配置为 115200、八个数据位，无奇偶校验，一个停止位，请参见 图 30。

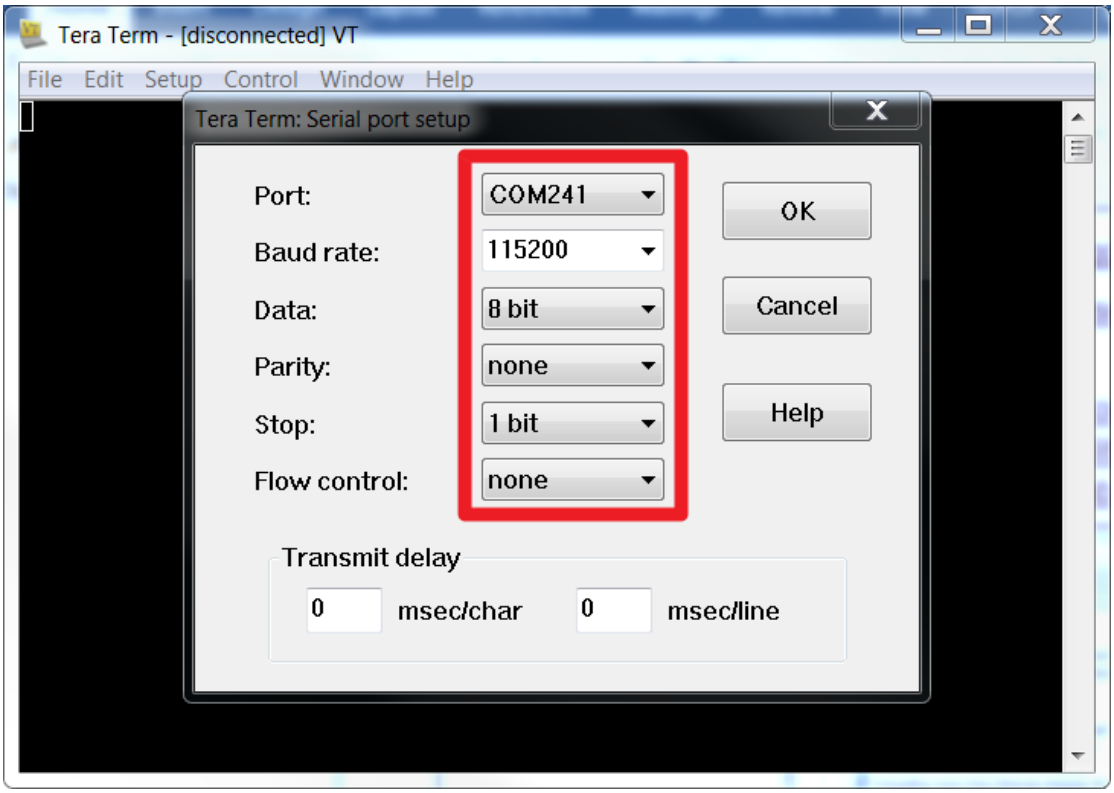

图 30. UART 调试终端配置

一旦将 CoreMark 必要文件添加到项目中（按照将 CoreMark 库集成到 SDK2.0 框架中的说明），编译该项目并下载到 LPC5500Xpresso 板上。

单击复位按钮，几秒钟后 CoreMark 测试的结果将在终端上显示出来，如 图 31 所示。

4 结果

图 31 显示了在 IAR 中以 96MHz 核心频率在 LPC5500 时的 CoreMark 基准测试结果。CoreMark 基准得分时每秒的迭代次数。从片上闪存运行的 CoreMark/MHz 得分为 372.786580/96MHz=3.883CoreMark/MHz。

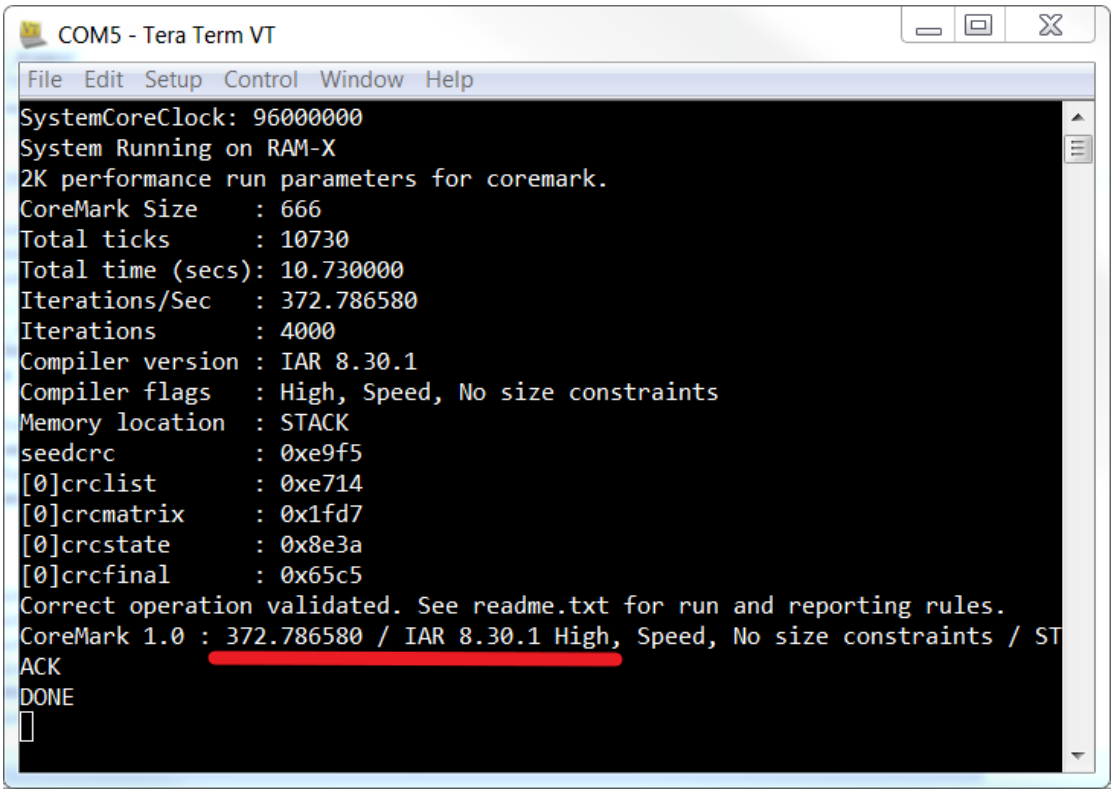

图 31. CoreMark 结果

表 1 显示了以 96 MHz 核心频率从片上闪存和 SRAM 运行时，在 Keil MDK，IAR EWARM 和 MCUXpresso IDE 上进行 CoreMark 测试的得分。

表 1. LPC55S69Xpresso 板 CoreMark/MHz 得分

IDE	CoreMark/MHz Score (SRAMX)	CoreMark/MHz Score (Flash)
KEIL MDK	4.021	2.333
IAR EWARM	3.887	2.435
MCUXpresso	2.843	2.016

注意

在 96 MHz 下测试。

对于 μA/MHz，表 2，表 3，和 表 4 显示了在室温状态下，在 VDD=3.3 V 时 LPCXpresso55S69 开发板上运行的结果。 图 32 比较了三种 IDE 编译后不同的电流消耗。

注意

EVK 上的当前数据可能略高于数据表，因为 EVK 具有很多的器件，可能会消耗更多功率。

150 MHz 的平均电流将高于其他模式，原因是 150 MHz 使能了 PLL，而 PLL 本身会消耗更多的电流。

表 2. Keil MDK µA/MHz 得分

频率	平均功耗（mA，SRAM X）	µA/MHz 得分（SRAM X）	平均功耗（mA，Flash）	µA/MHz 得分（Flash）
12 MHz	1.34	111.67	1.35	112.50
48 MHz	2.68	55.84	2.72	56.67
96 MHz	3.89	40.53	3.95	41.14
150 MHz	7.24	48.27	6.30	42.00

表 3. IAR EWARM µA/MHz 得分

频率	平均功耗（mA，SRAM X）	µA/MHz 得分（SRAM X）	平均功耗（mA，Flash）	µA/MHz 得分（Flash）
12 MHz	1.48	123.34	1.29	107.50
48 MHz	2.63	54.80	3.32	69.17
96 MHz	3.96	41.25	4.28	44.59
150 MHz	7.63	50.87	7.56	50.40

表 4. MCUXpresso µA/MHz 得分

频率	平均功耗（mA，SRAM X）	µA/MHz 得分（SRAM X）	平均功耗（mA，Flash）	µA/MHz 得分（Flash）
12 MHz	1.33	110.84	1.19	115.84
48 MHz	2.30	50.00	2.41	50.03
96 MHz	3.64	37.92	3.58	37.30
150 MHz	7.19	47.94	6.57	43.80

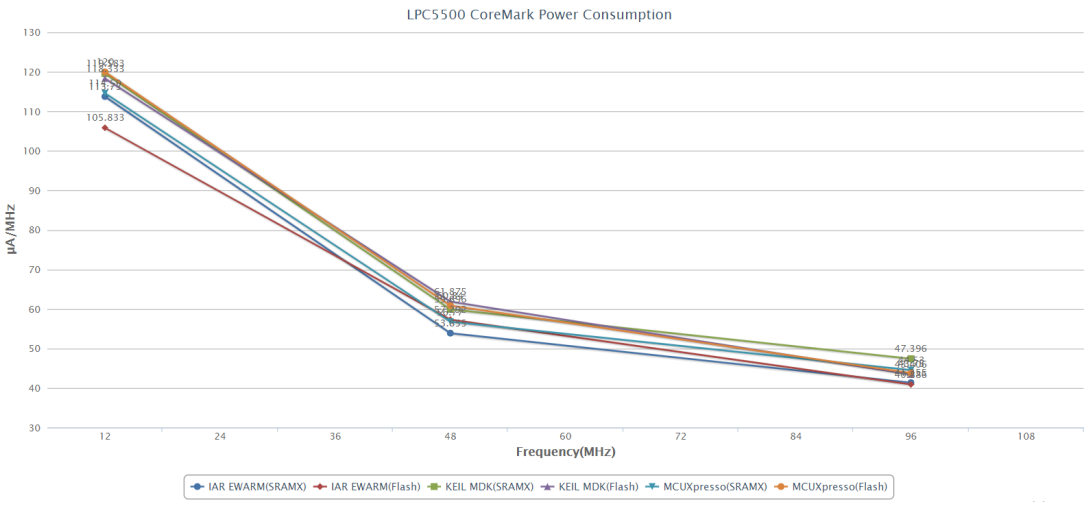

图 32. µA/MHz IDE 比较

5 结论

本笔记介绍了 LPC55xx 上的三种类型的 CoreMark 基准测试，支持不同的 IDE 开发环境（Keil，IAR，MCUXpresso）：

CoreMark 得分，功耗和 µA/MHz。

列出了内部 SRAM 和闪存中运行 CoreMark 的测试结果。

CoreMark 结果是在 LPCXpresso55S69 上测量的。在使用 KEIL MDK（Arm 编译器 6.12）并从 SRAM X 运行，CoreMark 可获得的最佳 CoreMark 值为 4.021。在内核频率为 96MHz 时，从闪存运行 CoreMark 程序可获得的最佳 CoreMark 功耗（µA/MHz）为 37.30。

6 参考资料

1. CoreMark Benchmarking for ARM Cortex Processors - 应用笔记 350
2. *LPC5411x CoreMark Cortex-M4 Porting Guide, NXP* (document AN11811)
3. *LPC55xx/LPC55Sxx User Manual* (document UM11126)

7 修订记录

版本号	日期	说明
0	2019 年 1 月 25 日	初始版本
1	2019 年 12 月	更新了“1B”版本芯片的 CoreMark 分数（使用 SDK2.6.3）

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

Limited warranty and liability — Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

Right to make changes - NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Security — Customer understands that all NXP products may be subject to unidentified or documented vulnerabilities. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer’s applications and products. Customer’s responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer’s applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately. Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

For more information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 2019 年十二月

Document identifier: AN12284

arm