AN11811

LPC5411x CoreMark Cortex-M4 Porting Guide

Rev. 1.0 — 21 March 2016 Application note

Document information

Info	Content
Keywords	LPC54114, CoreMark, Cortex-M4 Porting Guide, Keil MDK, IAR EWARM, LPCXpresso, Power Consumption
Abstract	This application note describes how to adapt the provided framework projects to run CoreMark benchmark tests on the Cortex-M4 core of the LPC5411x family of microcontrollers. Different project configuration options for best CoreMark number and µA/MHz are discussed in the document.

Revision history

Rev	Date	Description
1.0	20160321	Initial revision

Contact information

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

1. Introduction

CoreMark, developed by EEMBC, is a simple, yet sophisticated benchmark that is designed specifically to test the functionality of a processor core. Running CoreMark produces a single-number score allowing users to make quick comparisons between processors.

ARM released Application Note 350 “CoreMark Benchmarking for ARM Cortex Processors”. The application note can be found here:

http://infocenter.arm.com/help/topic/com.arm.doc.dai0350a/DAI0350A_coremark_benchmarking.pdf. This is also a useful resource to review and get familiar with the benchmark.

The following sections of this application note describes the process of setting up the LPC5411x device and software including memory partitioning, build options and compiler flags. It also shows how to measure CoreMark scores with the Cortex-M4 and presents the Cortex-M4 µA/MHz results that can be achieved. Separate CoreMark frameworks executing CoreMark code from Flash and SRAM are provided for Keil MDK, IAR EWARM and LPCXpresso IDE.

2. Integration of CoreMark library to LPCOpen framework

The software package associated with this application note contains LPCOpen based project framework that allows developers to drop in the CoreMark library sources and quickly get up and running with benchmarking the LPC5411x.

To get started go to: https://www.eembc.org/coremark. Click the download link as shown in Fig 1 and follow the instructions on that page.

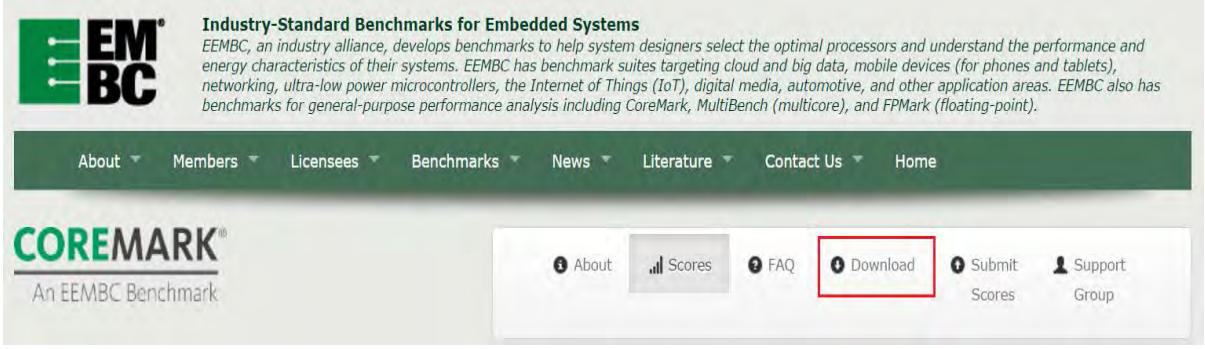

Fig 1. EEMBC CoreMark download link

After reviewing the license terms, look through the readme and documentation file. The readme gives step by step instructions on unpacking and building the distribution. This will also help with getting familiar with the CoreMark terminology used throughout the application note.

2.1 Porting CoreMark library into CoreMark framework

There are two variants of CoreMark projects in this application note for each IDE. One variant executes the CoreMark application from internal flash and other variant executes the CoreMark application from internal SRAM.

The various CoreMark projects are:

1. m4_coremark_flash – Cortex-M4 executes CoreMark application from internal Flash.

2. m4_coremark_ram – Cortex-M4 executes CoreMark application from internal SRAM.

The CoreMark projects are found in the following locations:

- Keil MDK IDE :
  - .\ lpc5411x\prj_lpcxpresso_54114\keil\m4_coremark_flash.uvmpw
  - .\ lpc5411x\prj_lpcxpresso_54114\keil\m4_coremark_ram.uvmpw
- IAR Workbench IDE:
  - .\ lpc5411x\prj_lpcxpresso_54114\iar folder\m4_coremark_flash.eww
  - .\ lpc5411x\prj_lpcxpresso_54114\iar folder\m4_coremark_ram.eww
- LPCXpresso IDE :
  - LPC5411x_lpcxpresso_m4_coremark_flash
  - LPC5411x_lpcxpresso_m4_coremark_ram

Depending on the toolchain, the workspace should look as shown in following figures. The m4_coremark_flash project workspace is shown as an example. The CoreMark framework requires the addition of the CoreMark files from EEMBC.

2.1.1 CoreMark framework to execute from internal Flash

The m4_coremark_flash project must be set as active before the CoreMark files can be added.

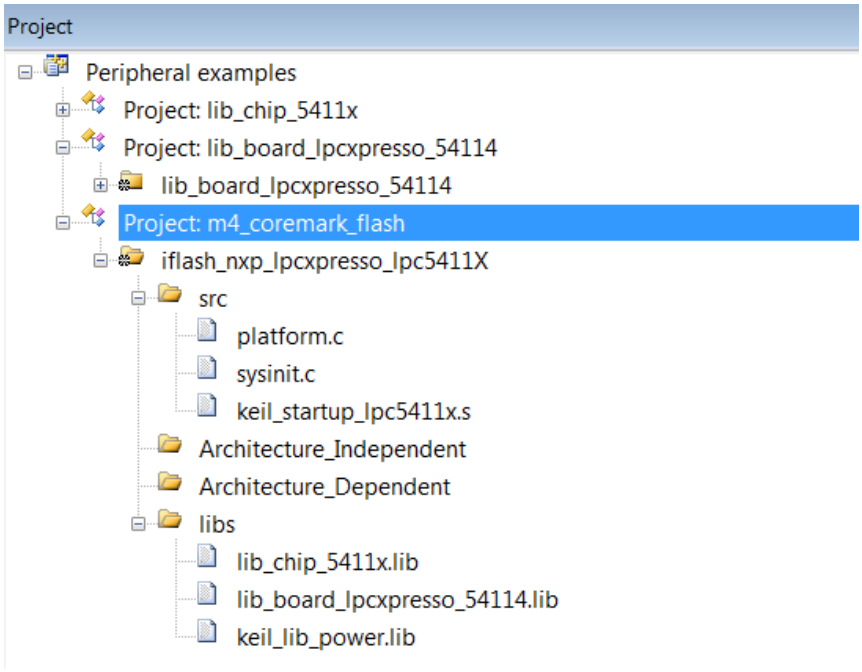

Fig 2. Keil MDK CoreMark framework

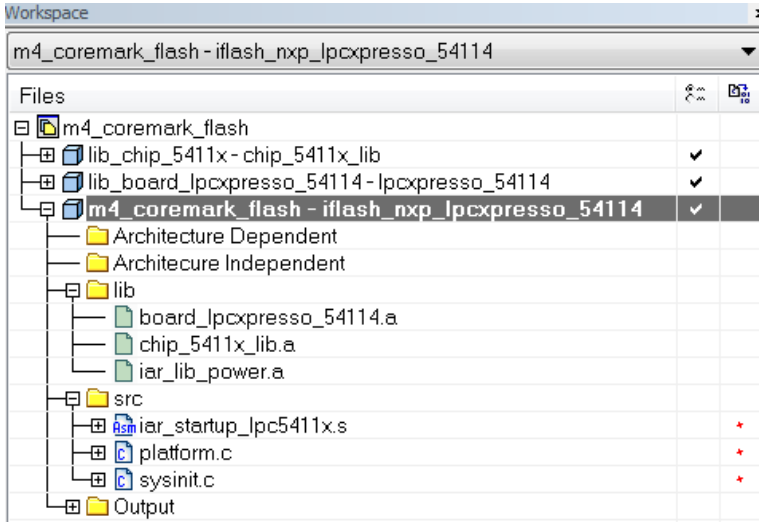

Fig 3. IAR EWARM workspace

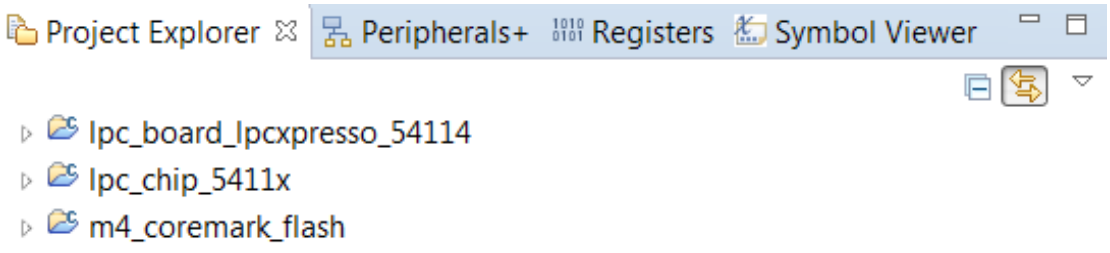

Fig 4. LPCXpresso IDE workspace

Copy the following files from the CoreMark package downloaded from EEMBC.

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

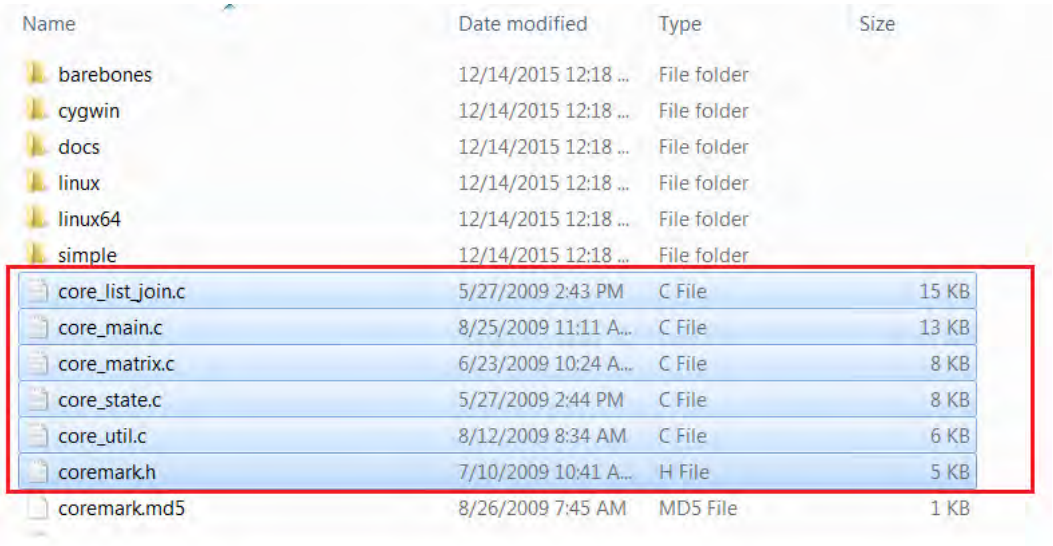

Fig 5. CoreMark files to copy

For Keil MDK place these files in the project directory

.\lpc5411x\examples_5411x\m4_keil_coremark_flash\Architecture_Independent

For IAR Embedded Workbench place these files in the project directory

.\lpc5411x\examples_5411x\m4_iar_coremark_flash\Architecture_Independent

For LPCXpresso IDE place these files in the project directory

.\LPC5411x_lpcxpresso_m4_coremark_flash\m4_coremark_flash\example\Architecture_Independent

The files ee_printf.c and cvt.c from the \barebones directory of the EEMBC CoreMark package need to be copied to the following folder locations.

For Keil IDE place the files in .\lpc5411x\examples_5411x\m4_keil_coremark_flash\keil_lpc5411x\Architecture_Dependent

For IAR Embedded workbench place the files in .\lpc5411x\examples_5411x\m4_iar_coremark_flash\iar_lpc5411x\Architecture_Dependent

For LPCXpresso IDE place the files in .\LPC5411x_lpcxpresso_m4_coremark_flash\m4_coremark_flash\example\Architecture_Dependent

Add the files into the Keil MDK project framework to their respective groups Architecture_Independent and Architecture_Dependent by double clicking on the groups.

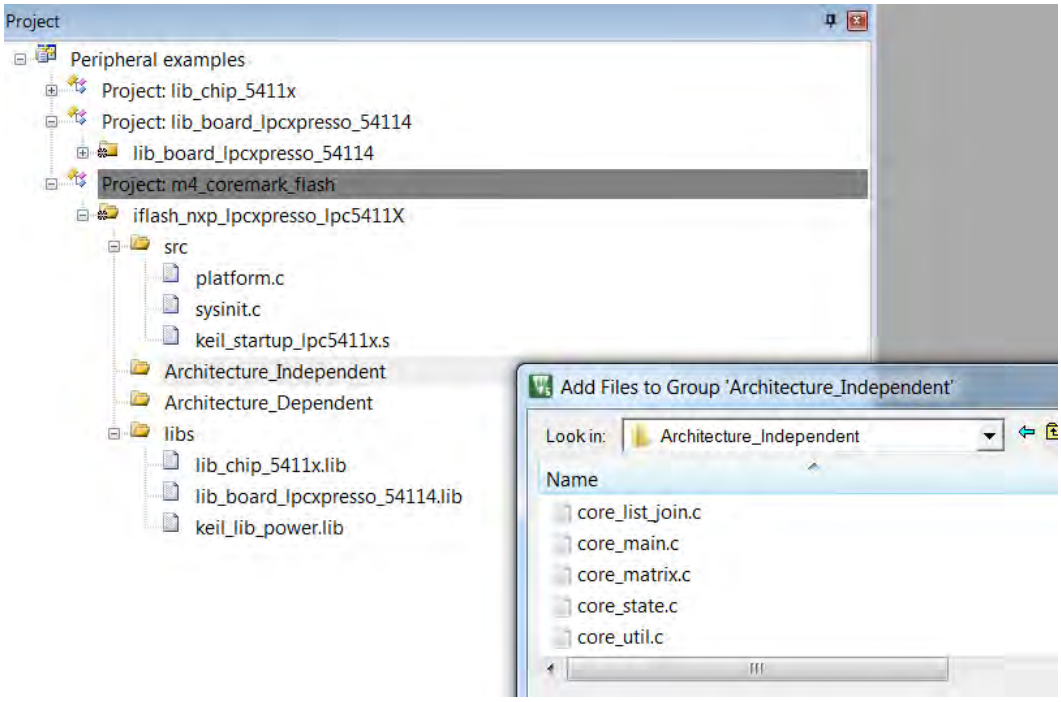

Fig 6. Adding Files in Keil MDK

For IAR Embedded workbench right click the Architecture_Independent and Architecture_Dependent folder and select Add and then “Add Files…”.

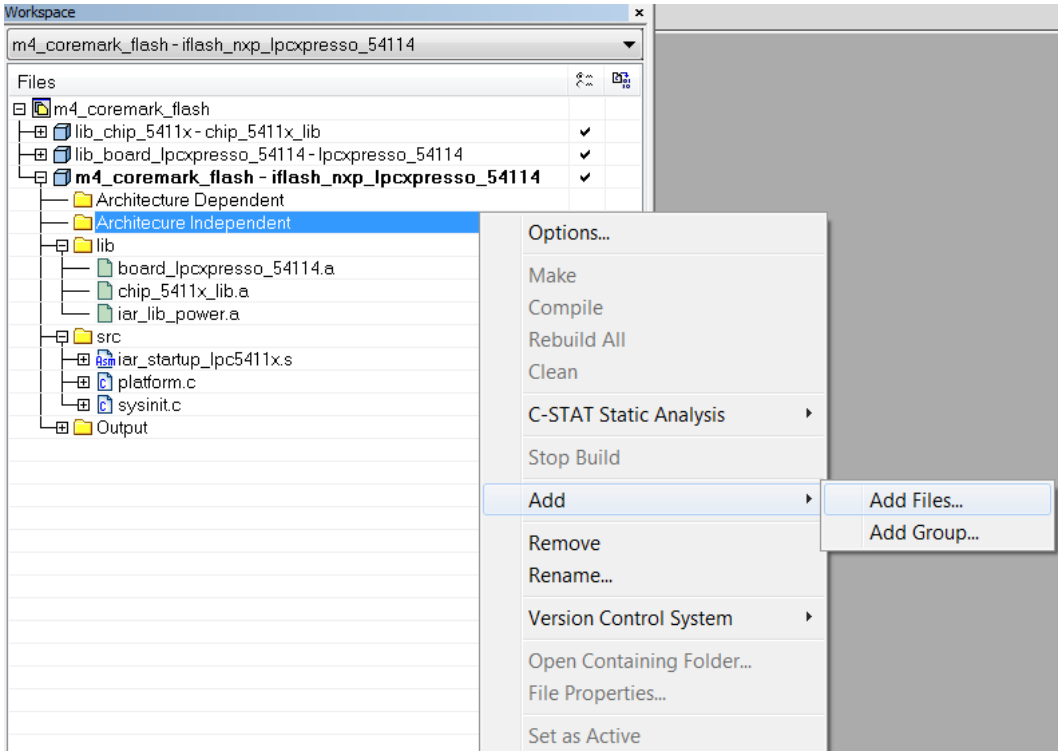

Fig 7. Adding Files in IAR EWARM workspace

For LPCXpresso IDE, right click inside the “Project Explorer” tab and select Refresh. LPCXpresso should detect the newly added files and add them to the Project Explorer automatically.

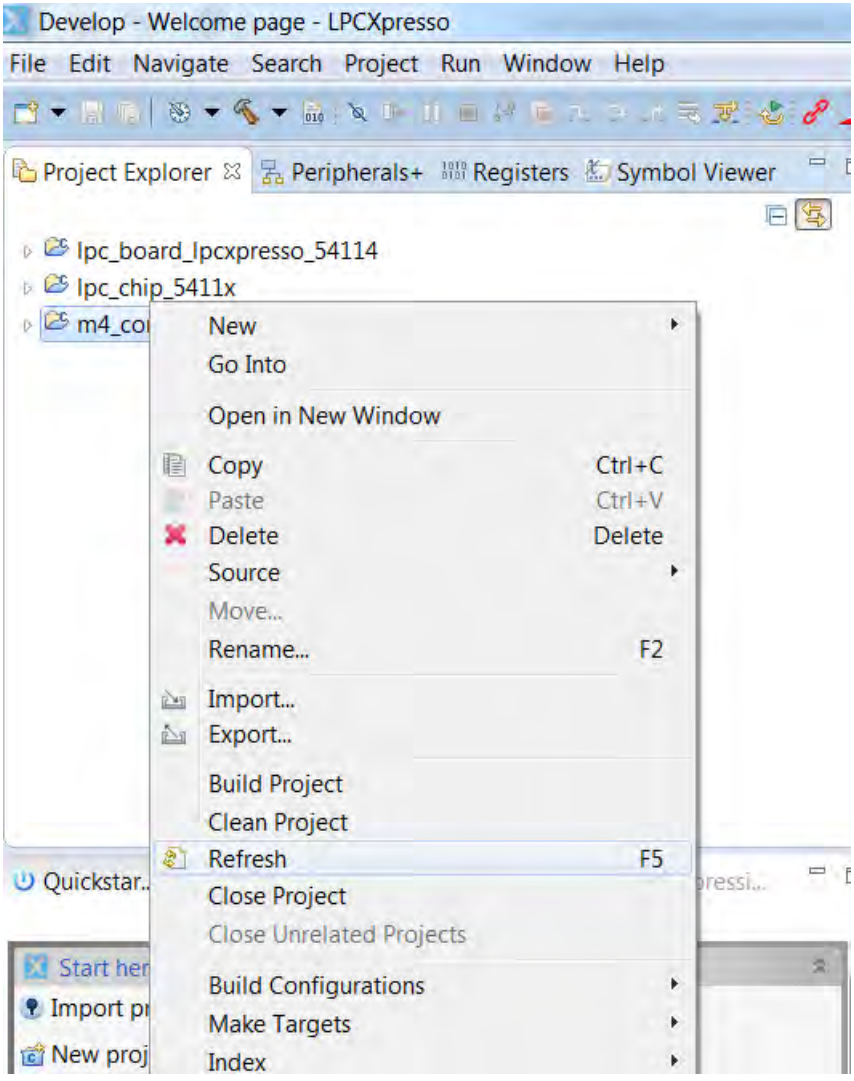

Fig 8. Refreshing LPCXpresso workspace

Use the core_portme.c and core_portme.h files provided with the application note and not the one from the EEMBC CoreMark package. For convenience these files have the required porting changes ready for use.

Copy these files to the Architecture_Dependent folder for all three tool chains and add the core_portme.c file in the project framework under the Architecture_Dependent group.

Once all the files have been added, the workspace should look as shown below:

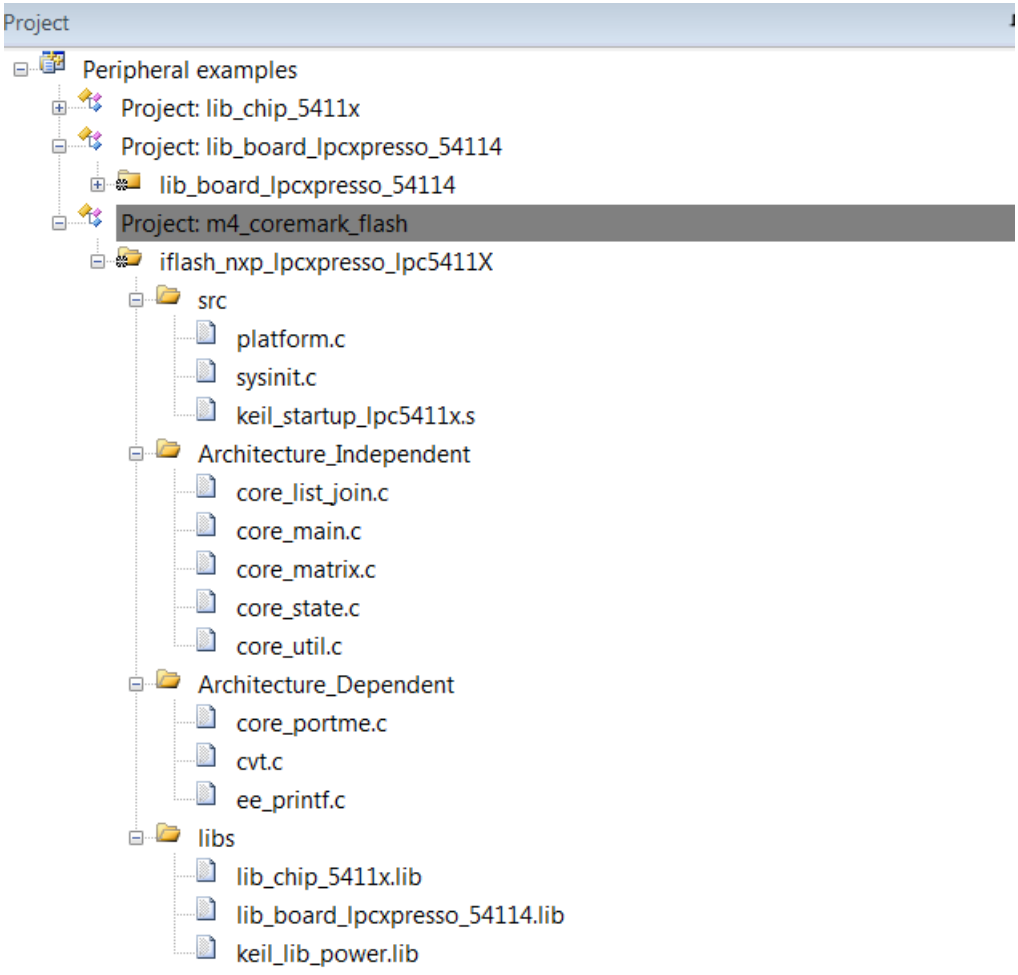

Fig 9. Keil MDK project workspace after adding CoreMark files

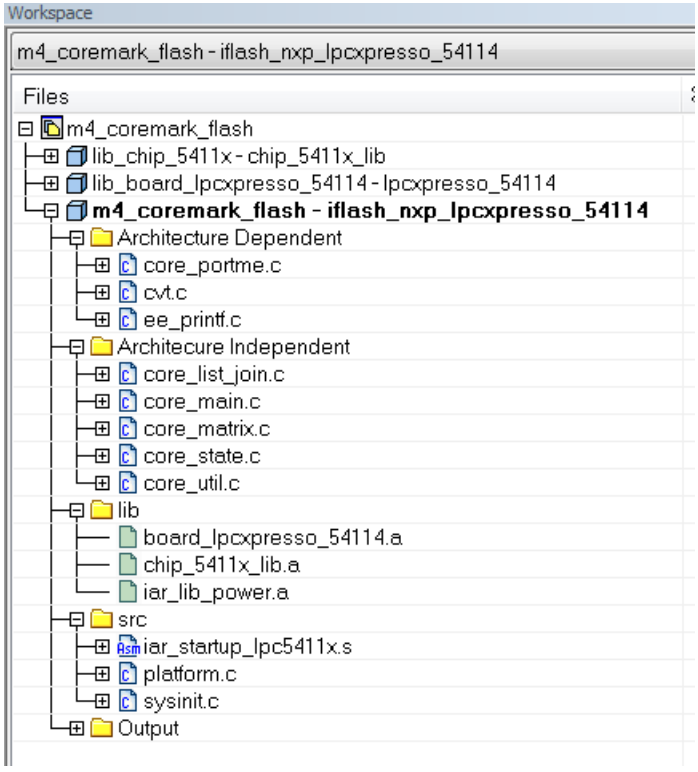

Fig 10. IAR EWARM workspace after adding CoreMark files

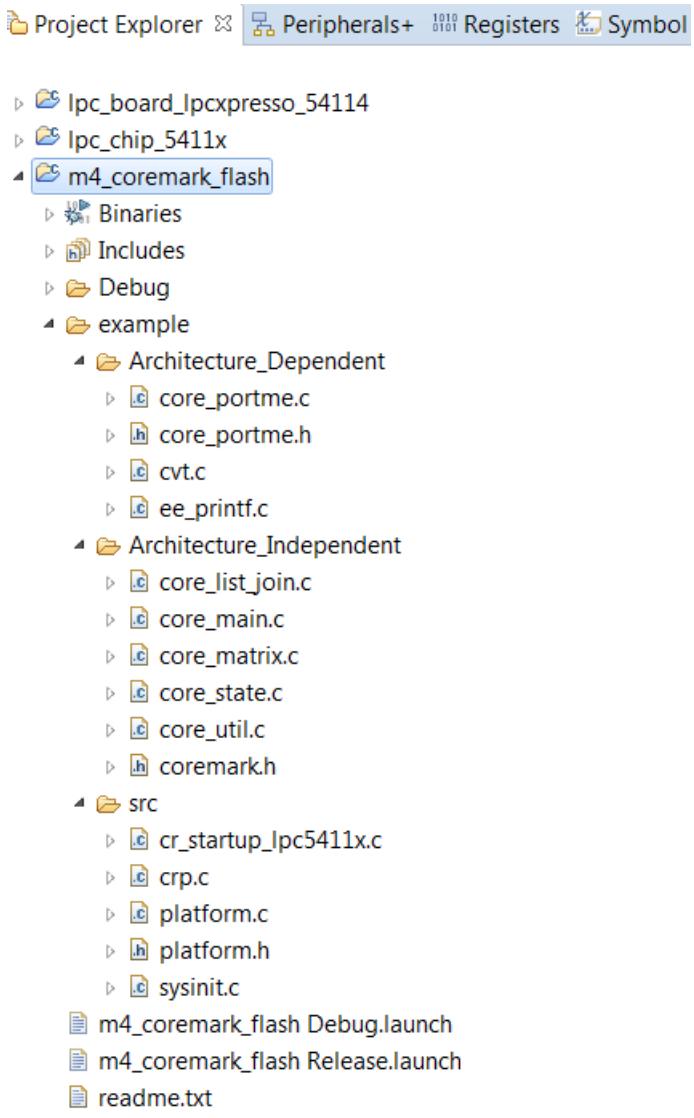

Fig 11. LPCXpresso workspace after adding CoreMark files

A few files need to be modified for executing CoreMark.

In the keil_startup_lpc5411x.s change the stack size from default 0x200 to 0x1000.

```
1    Stack_Size      EQU     0x00001000
```

To support 'printf' statements to a PC terminal, the 'eeprintf.c' file needs to be modified. Add the following line of code in *uart_send_char(char c)* function.

```
2    void uart_send_char(char c) {
3                portable_send_char(c);
4    }
```

In 'eeprintf.c' file, add an #ifdef COREMARK_SCORE_TEST in the function *ee_printf(const char *fmt,…).*

```
5    int ee_printf(const char *fmt, ...)
6    {
```

```
7       char buf[256],*p;
8       va_list args;
9       int n=0;
10      #ifdef COREMARK_SCORE_TEST
11      va_start(args, fmt);
12      ee_vsprintf(buf, fmt, args);
13      va_end(args);
14      p=buf;
15      while (*p) {
16          uart_send_char(*p);
17          n++;
18          p++;
19      }
20      #endif
21      return n;
22  }
```

This is added so that the printf code is optimized when running the µA/MHz test. In 'core_portme.h' there is a #define COREMARK_SCORE_TEST that dictates whether or not the application is executing the CoreMark score test.

In order to add the path to the header files used in the project, in Keil MDK under Project->Options-> C/C++tab, click 'Include path' and add the following paths that contain the header files.

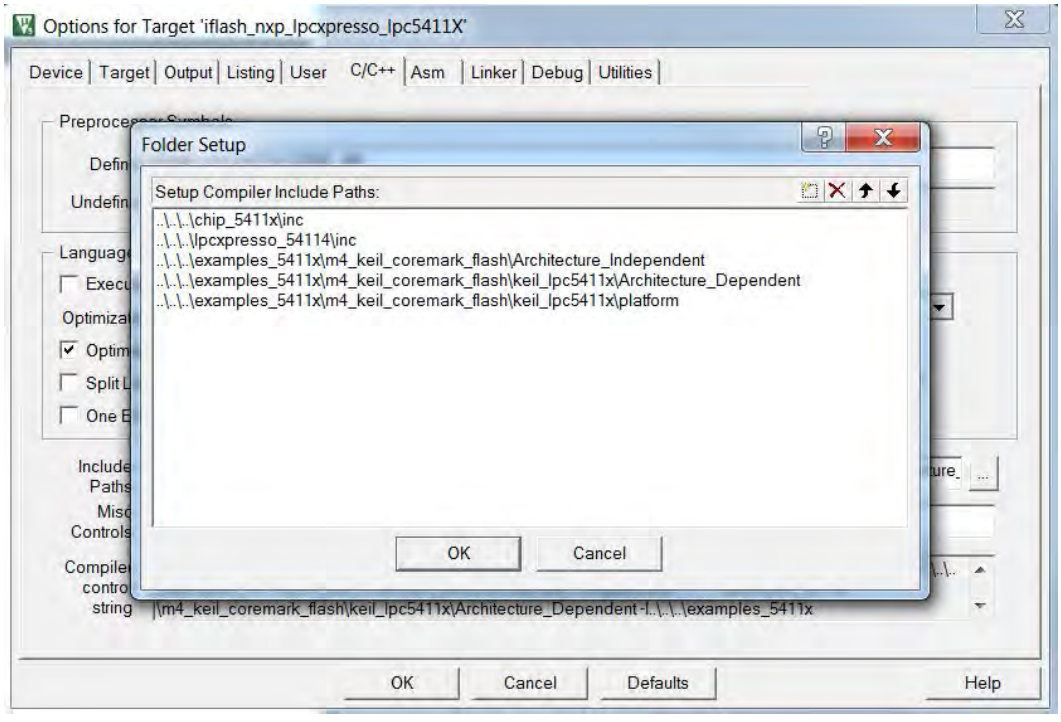

Fig 12. Keil MDK compiler include paths

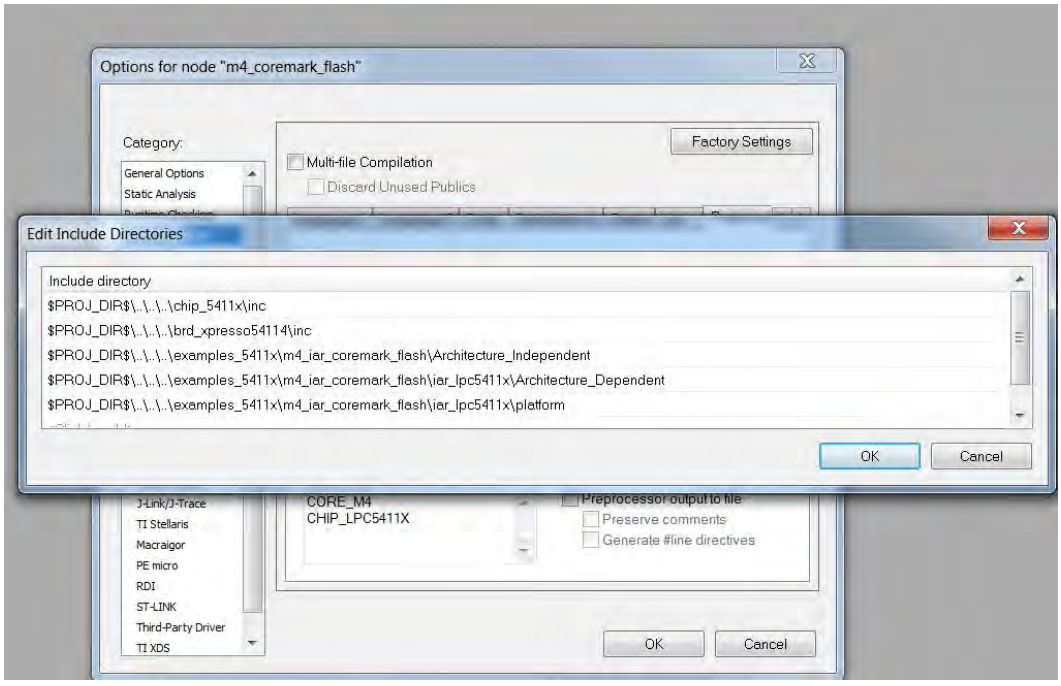

Fig 13. IAR EWARM compiler include paths

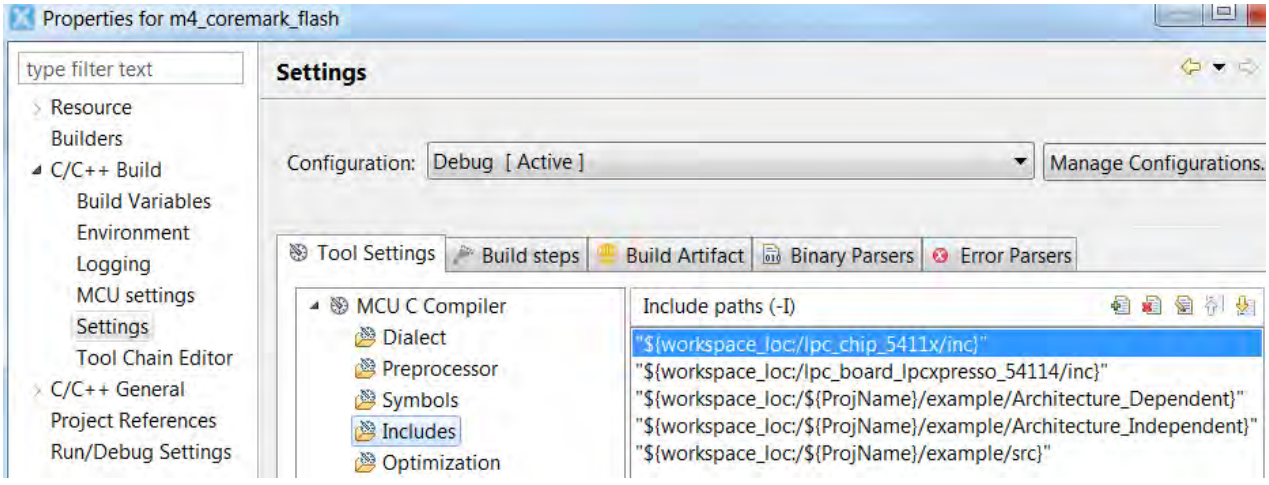

Fig 14. LPCXpresso compiler include paths

The CoreMark files have now been successfully ported into the CoreMark project framework.

2.1.2 CoreMark framework to execute from Internal SRAM

The project m4_coremark_ram executes the CoreMark application from 32 KB SRAMX memory region.

The files core_list_join.c, core_main.c, core_matrix.c, core_state.c and core_util.c are relocated to execute from SRAMX using the linker scripts.

For Keil MDK the linker script is located at:
.\lpc5411x\prj_lpcxpresso_54114\keil\support\m4_coremark_ram.sct

The linker script setting for m4_coremark_ram project is shown in Fig 15.

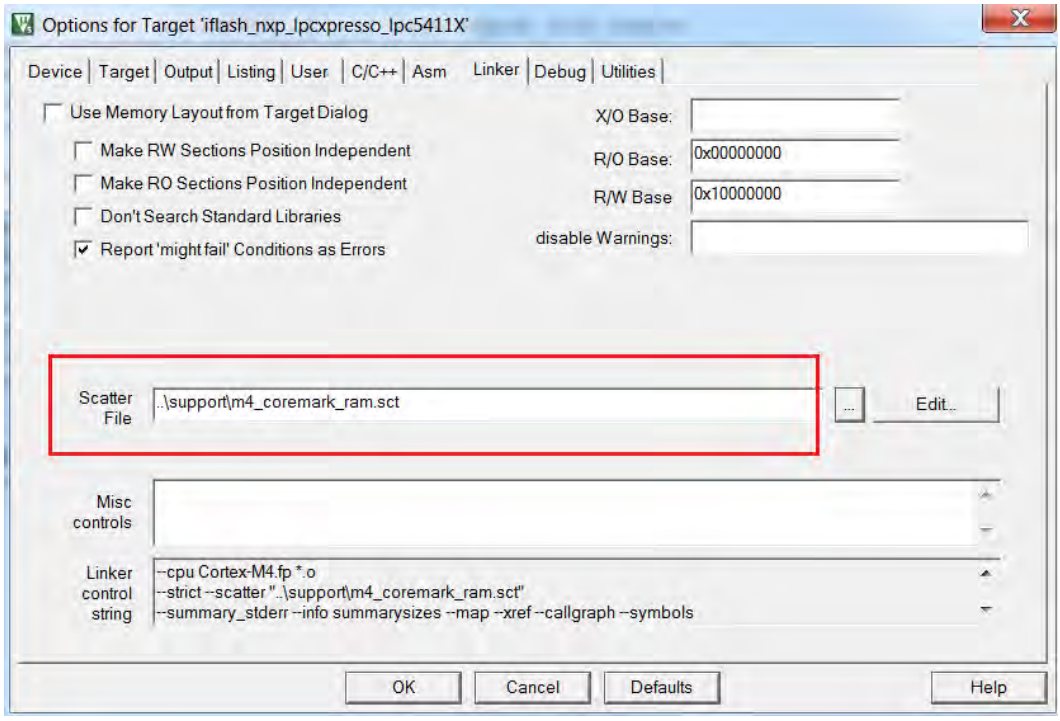

Fig 15. Linker script in Keil IDE

For IAR EWARM IDE to execute CoreMark in Internal SRAM, a line of code needs to be added to the files core_main.c, core_util,c, core_state.c, core_matrix.c and core_list_join.c.

These CoreMark files are labeled as their own IAR EWARM linker “section”. The provided .icf linker file in .\lpc5411x\prj_lpcxpresso_54114\iar\support\LPC5411x_M4_sramx_sram0.icf then places this section, which is called “critical_text” into SRAMX. To do this, add the following line of code, as shown in Fig 16, above #includes in all five files.

```
/*CoreMark source files should contain this #pragma command to be put into SRAM */
#pragma default_function_attributes = @ "critical_text"
```

Fig 16. IAR EWARM #pragma command

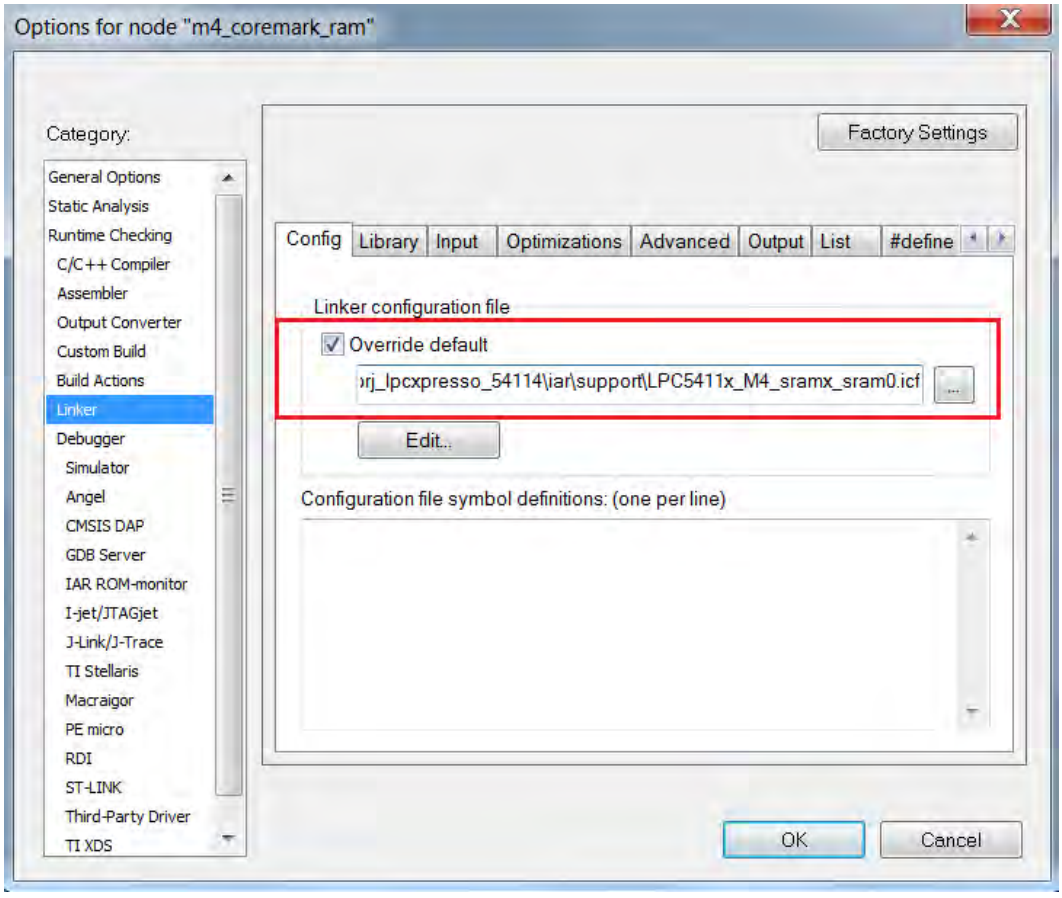

Fig 17. Linker script in IAR EWARM

In LPCXpresso, change the linker script folder path to the one located here:

\LPC5411x_lpcxpresso_m4_coremark_ram\m4_coremark_ram\linkscripts\

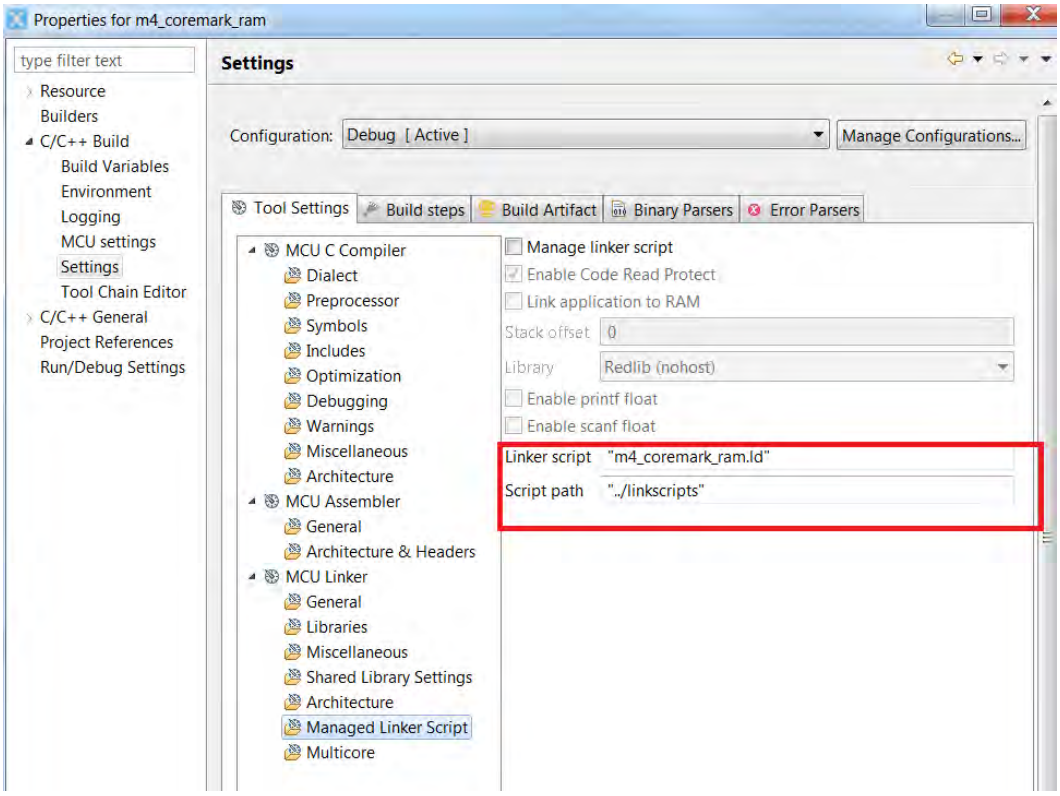

Fig 18. Linker script in LPCXpresso IDE

2.2 Optimizing the CoreMark framework

There are many factors that affect the CoreMark and µA/MHz score that can be optimized. Some of these factors are IDE dependent optimizations, while others leverage the MCU architecture for better performance. The goal is to be able to produce the best scores from all three IDEs. It is important to understand that these IDEs are constantly changing and a different version of a given IDE may add or remove features that may make these optimizations obsolete or ineffective. The following are the IDE versions that are applicable to this application note:

Keil MDK v5.17

IAR EWARM 7.60

LPCXpresso 8.1

2.2.1 Memory considerations

Due to the inherent architecture of SRAM and flash, CoreMark executes faster when running out of SRAM. The LPC5411x internal memory uses a multilayer AHB matrix system that provides a separate instruction and data bus for Cortex-M4 and SRAMX bank. See Fig 19. SRAM0, SRAM1 and SRAM2 are on System bus. Placing the CoreMark code and data in different SRAM banks minimizes bus contention and improves instruction and data parallelism.

It is important to minimize the flash wait states according to the MCU frequency to optimize the CoreMark score. In contrast, when performing the µA/MHz test, it is possible to save power by disabling the flash's prefetch ability. The LPC5411x user manual contains more information on correctly configuring the flash memory, such as the minimum amount of wait states allowed at a given core frequency.

The provided CoreMark framework projects include separate SRAM and flash based projects that implement various memory optimizations.

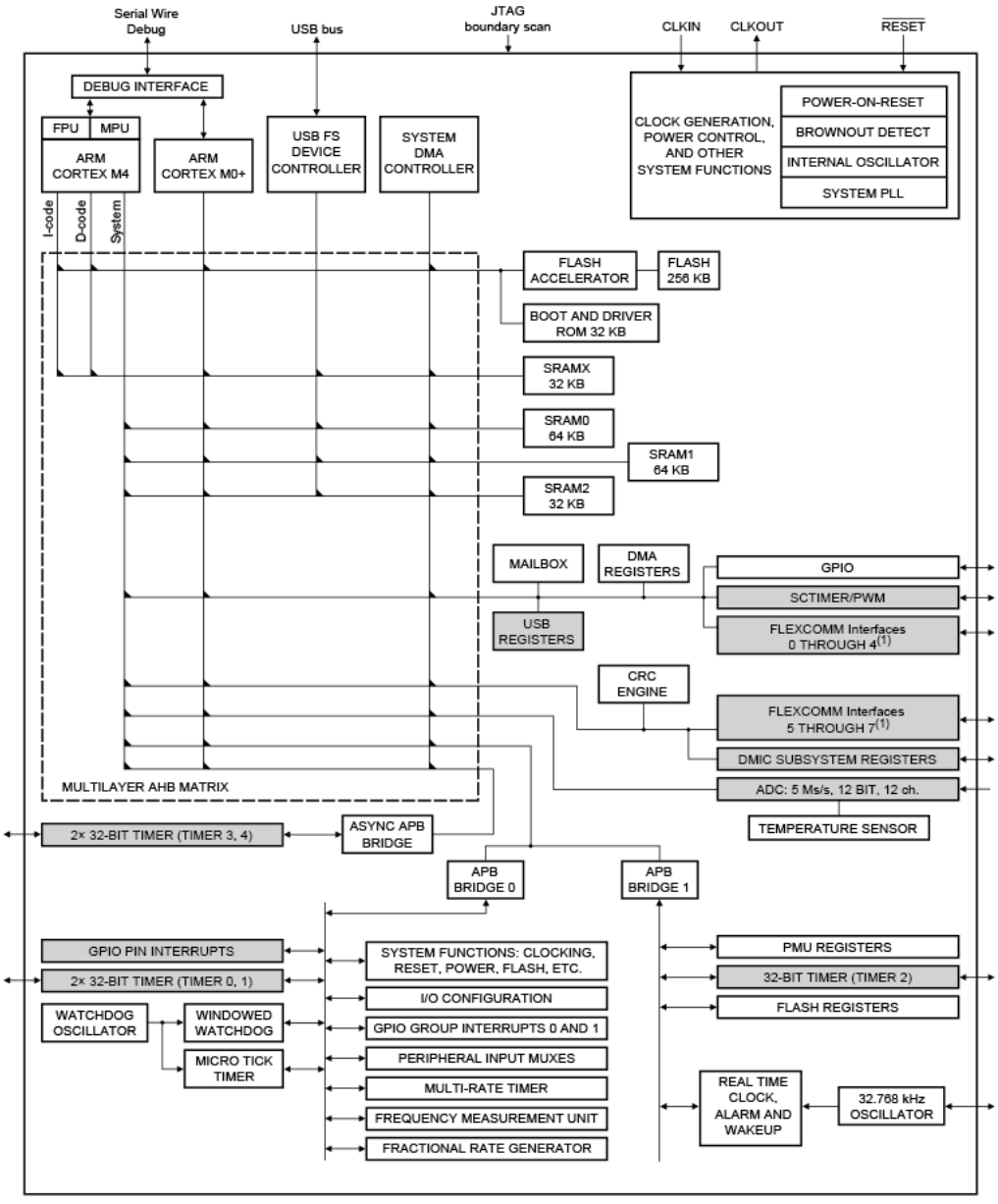

Fig 19. LPC5411x AHB matrix

In both the SRAM and flash projects, there is a COREMARK_SCORE_TEST macro defined in core_portme.h that indicates whether the project is configured to execute the CoreMark benchmark or the µA/MHz test. If this macro is defined, the CoreMark score

test will run. If this macro is commented out, the µA/MHz test will run. Use this macro to switch between the two benchmarks.

2.2.2 IDE Optimizations

The following optimizations are compiler based and therefore IDE dependent. These optimizations apply to both the SRAM and flash based projects.

2.2.2.1 Keil optimizations

There are two compiler optimizations that can be done to improve the CoreMark score. In Project->Options and under the C/C++ tab, the optimization level needs to be set to Level 3 (-O3) and “Optimize for Time” should be checked.

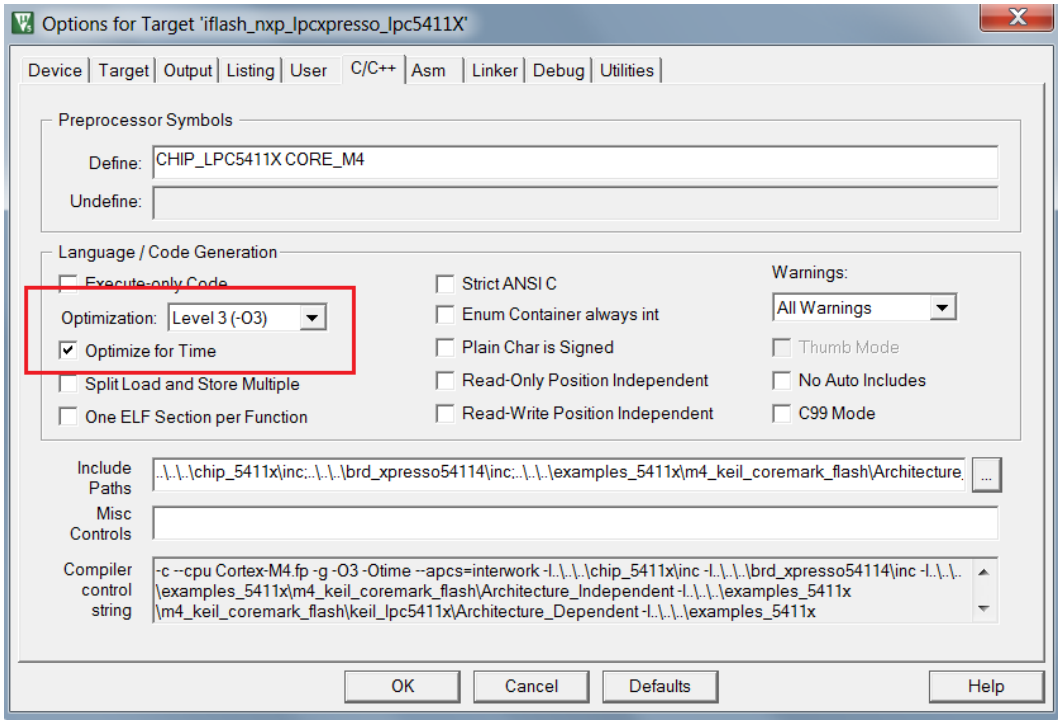

Fig 20. Keil MDK CoreMark score optimization

When benchmarking the power consumption of the MCU, the optimization setting must be set to Level 0 (-O0) and “Optimized for time” must be unchecked.

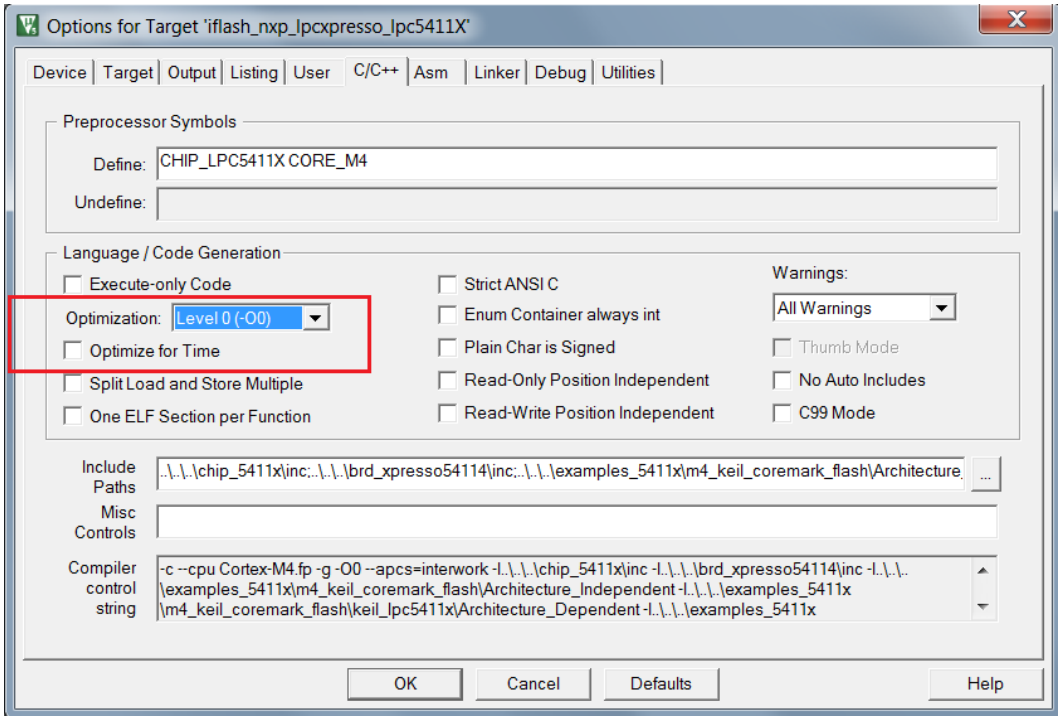

Fig 21. Keil MDK µA/MHz optimization

2.2.2.2 IAR Optimization

There are two compiler optimizations that can be done to improve CoreMark score. Set the optimization level to “High”, select “Speed” from the drop down menu and check the “No size constraints” checkbox.

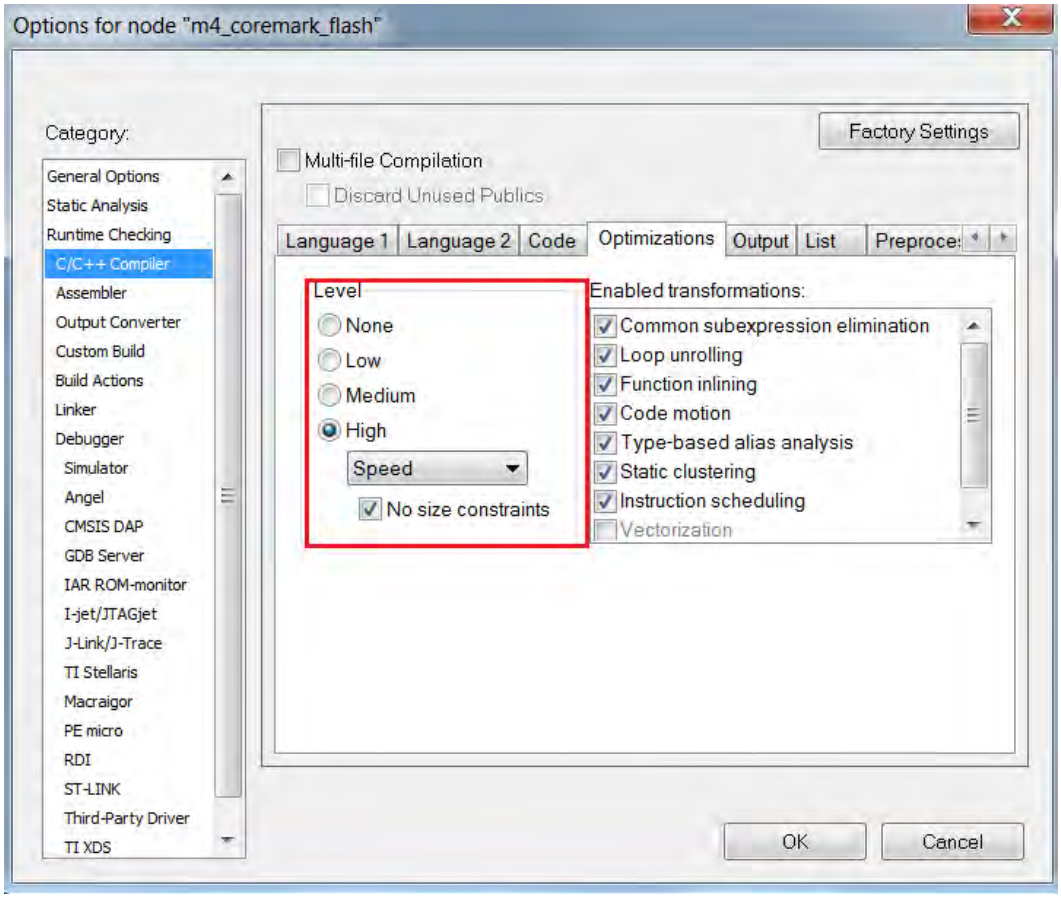

Fig 22. IAR EWARM CoreMark score optimization

When benchmarking the power consumption of the MCU, the optimization level should be set to “None”.

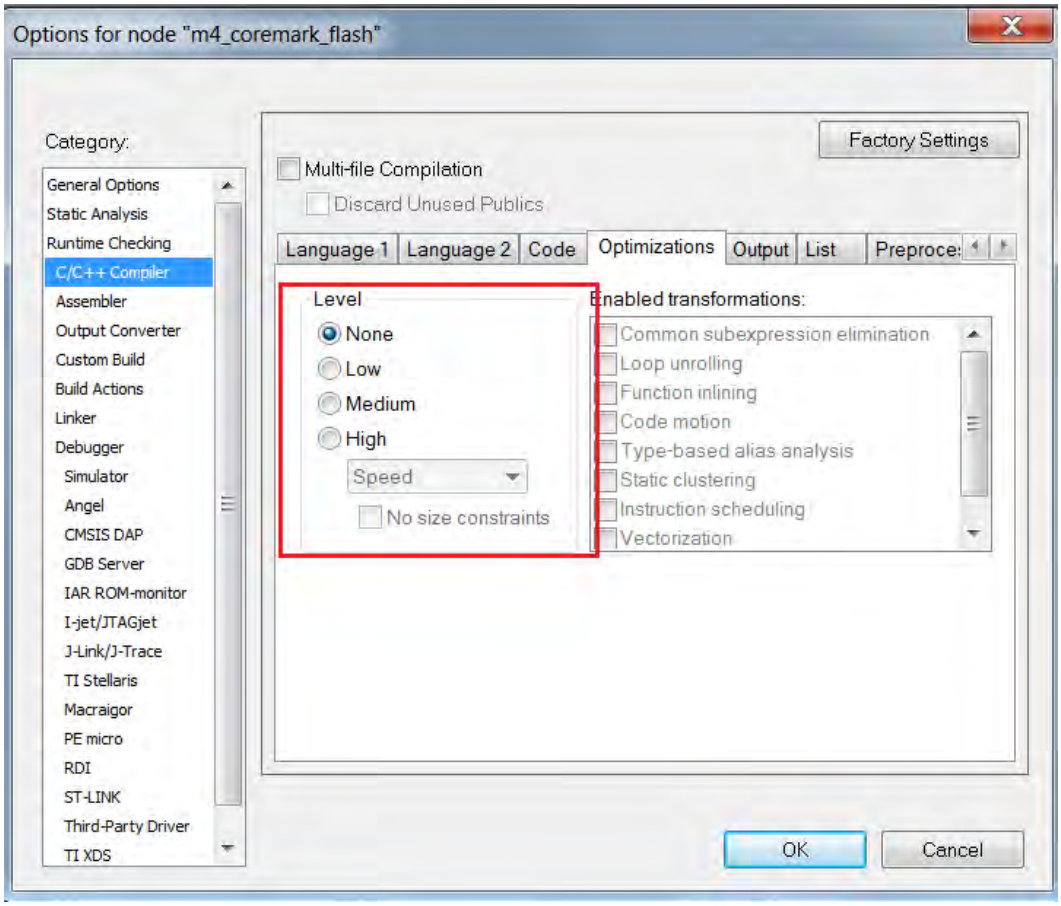

Fig 23. IAR EWARM µA/MHz optimization

2.2.2.3 LPCXpresso Optimization

There are two compiler optimizations that can be done to improve CoreMark score. Set the optimization level to “Optimize most (-O3)”.

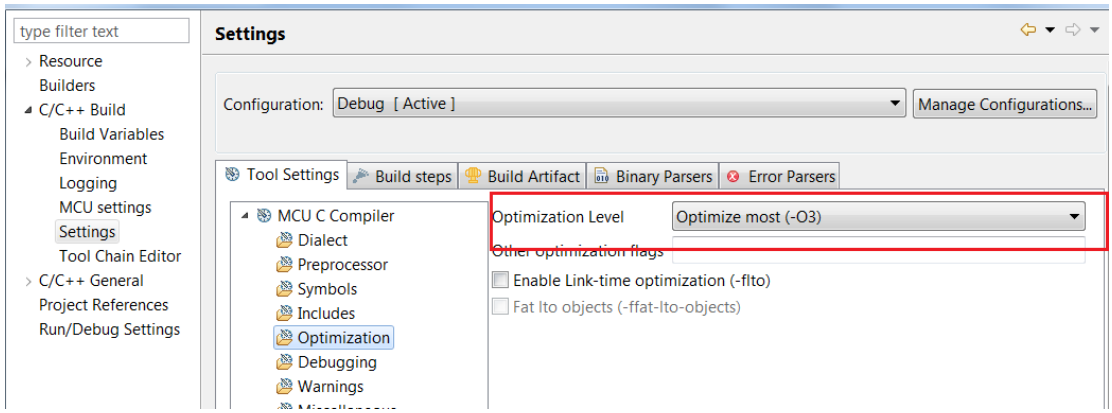

Fig 24. LPCXpresso CoreMark score optimization

When benchmarking the power consumption of the MCU, set the optimization level to “None (-O0)”.

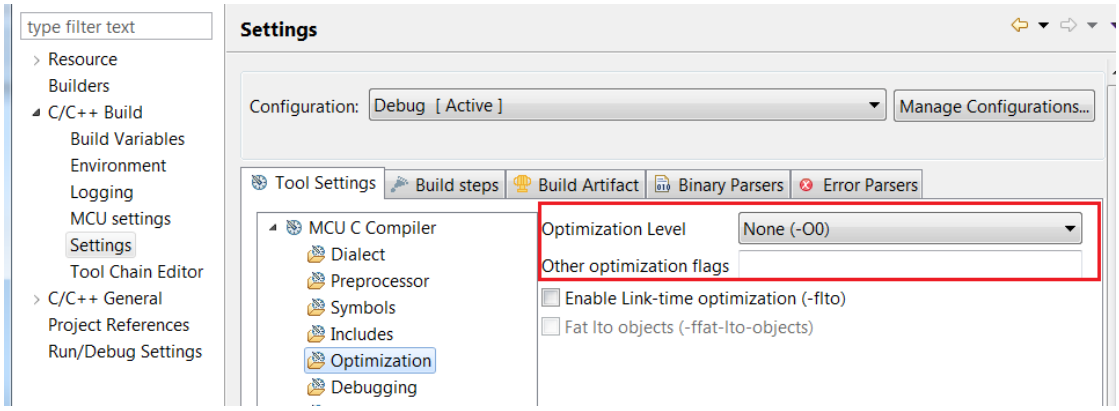

Fig 25. LPCXpresso µA/MHz optimization

3. LPCXpresso LPC54114 board setup

The LPC54114 LPCXpresso board supports a VCOM serial port connection via J6. To observe debug messages form the board set the terminal program to the appropriate COM port and use the setting ‘9600-8-N-1-none’. To make the debug messages easier to read, the new line receive setting should be set to auto.

3.1 Board setup

The LPCXpresso LPC54114 Rev A development board is used for benchmarking.

Fig 26. LPC54114 LPCXpresso board

The board ships with CMSIS-DAP debug firmware programmed. Visit the following FAQ for more information on CMSIS_DAP debug firmware: https://www.lpcware.com/content/faq/lpcxpresso/lpc-link2-debug-probe-firmware-programming

For debugging and terminal debug messages, connect a USB cable to J7 USB connector. Board schematics are available on www.lpcware.com.

3.1.1 µA/MHz measurement setup

To measure the LPC54114 power consumption, remove JS11, install header at JP4 and connect ammeter across JP4 as shown in Fig 27.

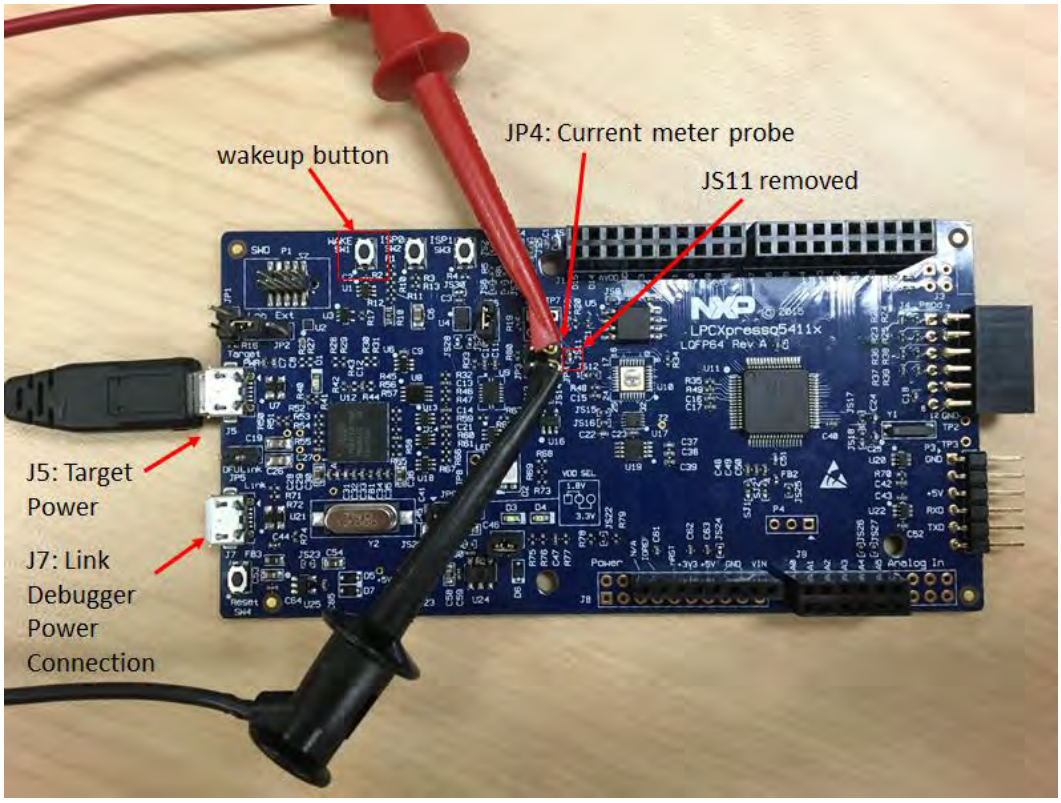

Fig 27. µA/MHz measurement setup

When performing the µA/MHz benchmark use J5 USB connector to provide power to the board. Additionally, after the µA/MHz benchmark project has been downloaded, power cycling the board by removing the USB cable and reinserting it is recommended to make sure the debug block is not left on by the debugger.

The baud rate setting for debug messages is 9600. This can be changed in platform.c file.

```
23    #define BAUD_RATE  9600
```

Similarly the core clock frequency can be changed by commenting/uncommenting the following lines in platform.c file.

```
24    #define SYS_CLK_RATE_HZ 12000000
25    #define SYS_CLK_RATE_HZ 48000000
26    #define SYS_CLK_RATE_HZ 96000000
```

4. Result

Fig 28 shows the result when running LPC5411x at 96 MHz core clock from Keil MDK. The CoreMark benchmark score is the number of iterations per second. The CoreMark/MHz score executing from internal flash for this run is 205.65/96 MHz = 2.142 CoreMark/MHz.

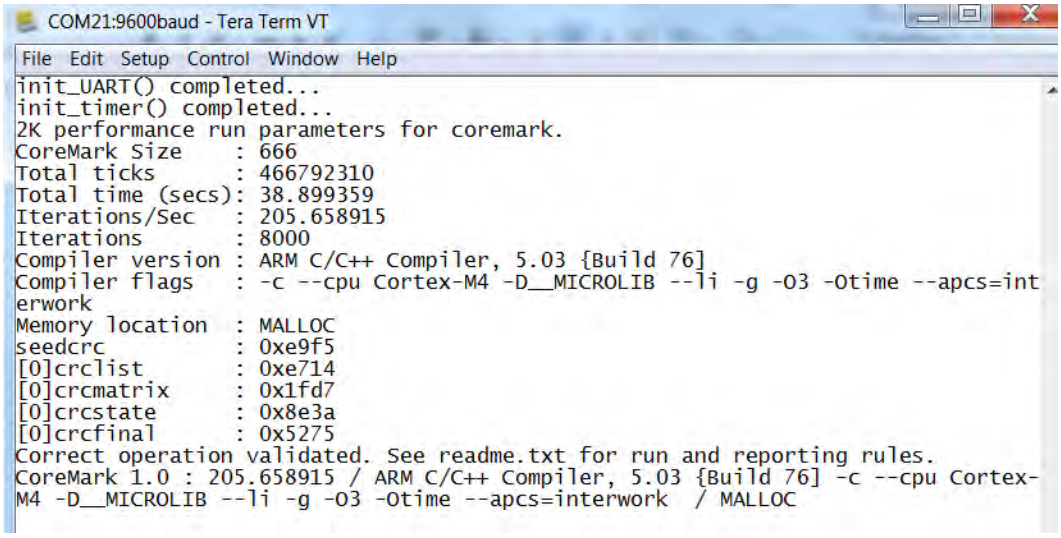

Fig 28. CoreMark result

Table 1 shows typical CoreMark score when benchmarked on Keil MDK, IAR EWARM and LPCXpresso IDE when running from internal flash and SRAM at 96 MHz.

Table 1. LPC54114 LPCXpresso board CoreMark/MHz Score

IDE	CoreMark/MHz Score (RAM)	CoreMark/MHz Score (flash)
Keil MDK	2.63	2.14
IAR EWARM	3.37	2.52
LPCXpresso	2.63	2.08

For µA/MHz, the following tables show typical results that can be expected when running on the LPC54114 LPCXpresso board with VDD = 3.3 V at room temperature. Fig 29 compares the three IDEs in terms of power consumption.

Table 2. Keil MDK µA/MHz score

Frequency	Avg. Power consumption (SRAM)	µA/MHz score(SRAM)	Avg. Power consumption (flash)	µA/MHz score (flash)
12 MHz	1.22 mA	101.66 µA/MHz	1.46 mA	121.66 µA/MHz
48 MHz	3.94 mA	82.08 µA/MHz	4.6 mA	95.83 µA/MHz
96 MHz	9.26 mA	96.45 µA/MHz	9.97 mA	103.85 µA/MHz

Table 3. IAR EWARM µA/MHz score

Frequency	Avg. Power consumption (SRAM)	µA/MHz score (SRAM)	Avg. Power consumption (flash)	µA/MHz score (flash)
12 MHz	1.22 mA	101.66 µA/MHz	1.6 mA	133.33 µA/MHz
48 MHz	3.93 mA	81.875 µA/MHz	5 mA	104.16 µA/MHz
96 MHz	9.22 mA	96.04 µA/MHz	10.37 mA	108.02 µA/MHz

Table 4. LPCXpresso µA/MHz score

Frequency	Avg. Power consumption (SRAM)	µA/MHz score (SRAM)	Avg. Power consumption (flash)	µA/MHz score (flash)
12 MHz	1.3 mA	108.33 µA/MHz	1.66 mA	138.33 µA/MHz
48 MHz	4.23 mA	88.125 µA/MHz	5.15 mA	107.29 µA/MHz
96 MHz	9.93 mA	103.43 µA/MHz	11.04 mA	115 µA/MHz

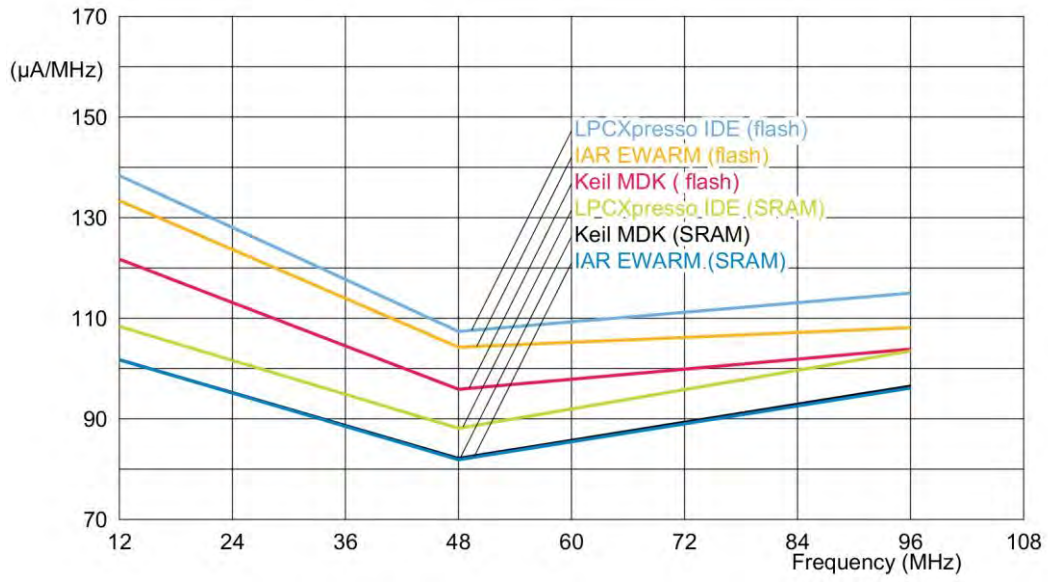

Fig 29. µA/MHz IDE comparison

5. Conclusion

In this application note, the two types of benchmarks that can be executed on the LPC5411x are presented: the CoreMark score benchmark and the µA/MHz benchmark. Details on how to optimize these benchmarks on the LPC5411x when running the benchmark out of internal SRAM and flash are discussed.

6. Legal information

6.1 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

6.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the *Terms and conditions of commercial sale* of NXP Semiconductors.

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.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Translations — A non-English (translated) version of a document is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Evaluation products — This product is provided on an "as is" and "with all faults" basis for evaluation purposes only. NXP Semiconductors, its affiliates and their suppliers expressly disclaim all warranties, whether express, implied or statutory, including but not limited to the implied warranties of non-infringement, merchantability and fitness for a particular purpose. The entire risk as to the quality, or arising out of the use or performance, of this product remains with customer.

In no event shall NXP Semiconductors, its affiliates or their suppliers be liable to customer for any special, indirect, consequential, punitive or incidental damages (including without limitation damages for loss of business, business interruption, loss of use, loss of data or information, and the like) arising out the use of or inability to use the product, whether or not based on tort (including negligence), strict liability, breach of contract, breach of warranty or any other theory, even if advised of the possibility of such damages.

Notwithstanding any damages that customer might incur for any reason whatsoever (including without limitation, all damages referenced above and all direct or general damages), the entire liability of NXP Semiconductors, its affiliates and their suppliers and customer's exclusive remedy for all of the foregoing shall be limited to actual damages incurred by customer based on reasonable reliance up to the greater of the amount actually paid by customer for the product or five dollars (US$5.00). The foregoing limitations, exclusions and disclaimers shall apply to the maximum extent permitted by applicable law, even if any remedy fails of its essential purpose.

6.3 Licenses

Purchase of NXP <xxx> components
<License statement text>

6.4 Patents

Notice is herewith given that the subject device uses one or more of the following patents and that each of these patents may have corresponding patents in other jurisdictions.

<Patent ID> — owned by <Company name>

6.5 Trademarks

Notice: All referenced brands, product names, service names and trademarks are property of their respective owners.

<Name> — is a trademark of NXP Semiconductors N.V.

7. List of figures

Fig 1. EEMBC CoreMark download link 3
Fig 2. Keil MDK CoreMark framework 5
Fig 3. IAR EWARM workspace 5
Fig 4. LPCXpresso IDE workspace 6
Fig 5. CoreMark files to copy 6
Fig 6. Adding Files in Keil MDK 7
Fig 7. Adding Files in IAR EWARM workspace 8
Fig 8. Refreshing LPCXpresso workspace 9
Fig 9. Keil MDK project workspace after adding CoreMark files 10
Fig 10. IAR EWARM workspace after adding CoreMark files 11
Fig 11. LPCXpresso workspace after adding CoreMark files 12
Fig 12. Keil MDK compiler include paths 13
Fig 13. IAR EWARM compiler include paths 14
Fig 14. LPCXpresso compiler include paths 14
Fig 15. Linker script in Keil IDE 15
Fig 16. IAR EWARM #pragma command 15
Fig 17. Linker script in IAR EWARM 16
Fig 18. Linker script in LPCXpresso IDE 17
Fig 19. LPC5411x AHB matrix 18
Fig 20. Keil MDK CoreMark score optimization 19
Fig 21. Keil MDK µA/MHz optimization 20
Fig 22. IAR EWARM CoreMark score optimization 21
Fig 23. IAR EWARM µA/MHz optimization 22
Fig 24. LPCXpresso CoreMark score optimization 23
Fig 25. LPCXpresso µA/MHz optimization 23
Fig 26. LPC54114 LPCXpresso board 24
Fig 27. µA/MHz measurement setup 25
Fig 28. CoreMark result 26
Fig 29. µA/MHz IDE comparison 27

8. List of tables

Table 1. LPC54114 LPCXpresso board CoreMark/MHz Score 26
Table 2. Keil MDK µA/MHz score 26
Table 3. IAR EWARM µA/MHz score 27
Table 4. LPCXpresso µA/MHz score 27

9. Contents

1. Introduction 3
2. Integration of CoreMark library to LPCOpen framework 3
2.1 Porting CoreMark library into CoreMark framework 4
2.1.1 CoreMark framework to execute from internal Flash 4
2.1.2 CoreMark framework to execute from Internal SRAM 14
2.2 Optimizing the CoreMark framework 17
2.2.1 Memory considerations 17
2.2.2 IDE Optimizations 19
2.2.2.1 Keil optimizations 19
2.2.2.2 IAR Optimization 20
2.2.2.3 LPCXpresso Optimization 22
3. LPCXpresso LPC54114 board setup 23
3.1 Board setup 23
3.1.1 µA/MHz measurement setup 24
4. Result 25
5. Conclusion 28
6. Legal information 29
6.1 Definitions 29
6.2 Disclaimers 29
6.3 Licenses 29
6.4 Patents 29
6.5 Trademarks 29
8. List of figures 30
9. List of tables 31
10. Contents 32

Date of release: 21 March 2016
Document identifier: AN11811