# AN12284 LPC55xx CoreMark on Cortex-M33 Porting Guide

Rev. 1 — December 2019

**Application Note** 

# **1** Introduction

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

LPC55xx is an Arm<sup>®</sup> Cortex<sup>®</sup> -M33 based microcontroller for embedded applications. These devices include:

- An Arm Cortex-M33 coprocessor
- CASPER Crypto/FFT engine
- · PowerQuad hardware accelerator for DSP functions
- Up to 320 KB of on-chip SRAM, up to 640 KB on-chip flash
- PRINCE module for on-the-fly flash encryption/decryption
- High-speed and full-speed USB host and device interface with crystalless operation for full-speed, SDIO/MMC
- Five general-purpose timers, one SCTimer/PWM, one RTC/alarm timer
- One 24-bit Multi-Rate Timer (MRT)
- A Windowed Watchdog Timer (WWDT)
- Nine flexible serial communication peripherals (which can be configured as a USART, SPI, high-speed SPI, I2C, or I2S interface)
- Programmable Logic Unit (PLU)
- · One 16-bit 1.0 Msamples/sec ADC, comparator, and temperature sensor

The Cortex-M33 offers 18.2 % performance increase in the same process technology compared to the high-embedded performance bars established by Cortex-M4 processors, while improving power efficiency. Cortex-M33 official CoreMark is 4.02 CoreMark/MHz, Cortex-M4 official CoreMark is 3.40 CoreMark/MHz.

This application note describes how to port CoreMark code to LPC55xx, which involves setting up software and hardware including memory partitioning, compiler setting, and board setup. It also describes how to measure CoreMark scores on the Cortex-M33 and the result including CoreMark scores and power consumption in µA/MHz. Separate CoreMark projects for different software development tools (Keil MDK, IAR EWARM, and MCUXpresso IDE) are also included herewith for reference.

# 2 Integration of CoreMark library to SDK2.0 framework

The software package associated with this application note contains SDK2.0 based project framework. It allows developers to drop in the CoreMark library sources and quickly get up and running with benchmarking the LPC55xx. To get started, go to: https://www.eembc.org/coremark, Click the **Download** link as shown in Fig. 1 and follow the instructions on the page.

<ul><li>2.1 Porting CoreMark library into CoreMark framework</li><li>2.2 Optimizing the CoreMark framework</li></ul>	2 18
3 Measuring CoreMark on board	25
3.1 LPC55S69Xpresso board	25
3.2 Board Setup	25
3.3 Run CoreMark code	28
4 Result	29
5 Conclusion	. 32
6 Reference	. 32
7 Revision history	32

1 Introduction.....1

SDK2.0 framework.....1

2 Integration of CoreMark library to

Contents



	C https://www.	.eembc.org/coremark/						\$	
		IBEDDED MICRO NCHMARK CON!	iprocessor Sortium						
	Member & Lic	ensee Request	Members •	Licensees •	Benchmarks	Newsletter	Press	Library •	About
	About - FAC About - FAC	eMark® is a benchm	Submit Score - Goog	gle Group the performance of	microcontrollers (MC	Us) and central proc	cessing units (	CPUs) used in emt:	pedded
gure 1. EEMBC C	oreMark	download	link web	opage					

After reviewing the license terms, go through the readme and documentation file. The readme provides step-by-step instructions on unpacking and building the distribution. It also helps in 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 for each IDE. One executes the CoreMark application from internal flash and other executes the CoreMark application from internal SRAMX

The CoreMark projects are:

1. run\_in\_flash\_xxmhz - Cortex-M33 executes CoreMark application from internal flash.

2. run\_in\_ramx\_xxmhz - Cortex-M33 executes CoreMark application from internal RAM.

The locations of CoreMark projects are:

Keil MDK IDE :

- lpc5500\_coremark\_mdk\coremark.uvprojx.eww

IAR Workbench IDE:

- lpc5500\_coremark\_iar\coremark.eww

Each of executes settings have four frequency settings : 12 MHz, 48 MHz, 96 MHz and 150 MHz.

Depending on the toolchain, the workspace should look as shown in below figures. The CoreMark framework requires the addition of the CoreMark files from EEMBC.

#### 2.1.1 CoreMark framework for Keil MDK / IAR EWARM / MCUXpresso IDE

The run\_in\_xxxx\_xxmhz project must be set as active before the CoreMark files can be added.

#### Integration of CoreMark library to SDK2.0 framework

W	_	Second Sec.	abl	ement\SW\lpc55
File Edit View Project Flash	Debug Peripherals To	ools SVCS V	Vindow Help	
🗋 🗋 🚰 🛃 🎒 👗 🛍 🛍 🧐	$[\phi_{-}] \not \leftrightarrow \Rightarrow [-\phi_{-}] \not \oplus \not \oplus \not \oplus$	18日 年 月	//= //👷 🖄 startu	o_lpc55s69_cn 💌
🛛 🤌 🖾 🛍 🥔 🔜 🛛 🗱 🛛 run_in_	flash_12mhz 🔽 🔊 🛔	🗟 🔶 🐡 🙆	1	
Project run_in_	ramx_12mhz			
Project: coremark	ramx_48mhz			
i	flash_12mhz = flash_48mhz = flash_96mhz =			



#### Integration of CoreMark library to SDK2.0 framework

X lpc5500_coremark_	mcux - Welcome page - N	1CUXpresso	IDE	
File Edit Navigate Se	arch Project ConfigTool	s Run Fre	ertos Window Hel	lp
	▼ 🗟   🖉 🌣   🕥 ▼   🖻			₹ \$\$  0> 00 1
🔁 Proj 🛛 🔀 Peri 🖁	🗄 Regi 🞋 Faults 🖾 Sy		📄 readme.txt 🛛 🎱 🛚	Velcome 🖾
	🖻 🔄 🖶 🍫	🗙 🔻 🖂	⇔ ⇒ ■ 🔗 file:///0	C:/nxp/MCUXpressoIDE_10.3.0_2187_prc3/ide
E-C Ipc5500_coremark	ZDUN IN ELACH 12MH75. EW	<b>▲</b>	•	
Includes Go	o Into			
E CMSIS	nen in New Window		-	
🗄 😂 board Sh	now in Local Terminal	1		
🗄 🔁 componer		Ctrdu C		To install an SDK, simply drag and drop an SDK (sip Netholder) into Name
device 📋 Co	opy	Ctrl+V		E         #50x_2x_1PC3persor54114         22.5
E 🔁 libs 💥 De	elete	Delete		
	ource	1	-	DE 000,24,1404,450 225 0, -
🖻 😕 startup 🛛 🕅	ove			
E PLIN THE	ename	F2		
	nport			Welcome
	kport		N	MCUXpresso IDE provides an easy-to-use Eclip
E BI	uild Project		- (	Cortex®-M cores, including LPC and Kinetis mi
	ean Project		n	nulticore debugging, and integrated configuration
	efresh	F5		Your installation of MCUXpresso IDE is ready to
	ose Project		r	equired.
🕀 🗁 uAMHz_IN	ose Unrelated Projects		Docur	mentation
🕀 🗁 uAMHz_IN 🛛 🔒	uild Configurations		Set Active	✓ 1 RUN_IN_FLASH_12MHz
	uild Targets	I	Manage	2 RUN_IN_FLASH_48MHz o IDE
	dex		Build All	3 RUN_IN_FLASH_96MHz le
Va	alidate		Clean All	4 RUN_IN_SRAMX_12MHZ
() Quic (x)= Glob., Rt	un As	I	Build Selected	- 6 RUN IN SRAMX 96MHz
De	ebug As		i Installed S 🔲 Pr	7 uAMHz_IN_FLASH_12MHz
	one as estore from Local History			8 uAMHz_IN_FLASH_48MHz
Project: lpc La	aunch Configurations	1	consoles to display a	9 uAMHz_IN_FLASH_96MHz
T Create or imp	tilities	I	•	10 UAMHZ_IN_SRAMX_12MHZ
SE SE	DK Management	I	•	12 JAMHZ IN SRAMX 96MHZ
New pro To	pols	I	•	
	CUXpresso Config Tools		•	
	an C/C++ Coue Analysis			
■ Build your pro     Co     Co	ompare With	1	•	
👧 🔨 Build 🛛 Co	onfigure	1	•	
Sc 🖉 🖉 Clean	ource	I	•	
- Debug your pi Pr	operties	Alt+Enter		
Figure 4. MCUXpresso	o project configuration	select		

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

👃 barebones	2018/9/28 17:17	File folder	
🔒 cygwin	2018/9/28 17:17	File folder	
🝌 docs	2018/9/28 17:17	File folder	
🝌 linux	2018/9/28 17:17	File folder	
🝌 linux64	2018/9/28 17:17	File folder	
🝌 simple	2018/9/28 17:17	File folder	
🕖 core_list_join.c	2018/5/31 10:42	C File	15 KB
🗾 core_main.c	2018/5/31 10:42	C File	13 KB
🕖 core_matrix.c	2018/5/31 10:42	C File	8 KB
🕖 core_state.c	2018/5/31 10:42	C File	7 KB
🕖 core_util.c	2018/5/31 10:42	C File	5 KB
🕺 coremark.h	2018/5/31 10:42	C Header Source F	5 KB
JICENSE.md	2018/5/31 10:42	Markdown Source	19 KB
🔋 Makefile	2018/5/31 10:42	File	4 KB
README md	2018/5/31 10:42	Markdown Source	19 KB

#### Figure 5. CoreMark files to copy

-For Keil MDK place these files in the project directory

lpc5500\_coremark\_mdk\source

-For IAR Embedded Workbench place these files in the project directory

lpc5500\_coremark\_iar\source

-For MCUXpresso place these files in the project directory.

lpc5500\_coremark\_mcux\source

The files ee\_printf.c, core\_portme.c and core\_portme.h(under port\_lpc5500 folder)need to be copied to the following folder locations.

-For Keil IDE place the files in "lpc5500\_coremark\_mdk\source"

Add the files into the Keil MDK project framework to the respective groups source by double clicking on the groups.

File Edit View Project Flash Debug Peripherals Tools SVCS Window	Help	
□ 22 및 21   3   10   10   10   10   10   10   10	🖄 startup_lpc55s69_cn 🔻 🚦	<u>،</u> د 🌒 🕲 🗠 🕼
🛛 🖉 🕮 🥔 🤐 🛱 run_in_flash_12mhz 🔽 🔊 🛔 着 🛸 🗇 幽		
Project 🛛 🗣 📧		
Project: coremark		
🖻 🐖 run_in_flash_12mhz		
🖶 📮 board		
🗉 📮 CMSIS		
🗉 📮 component/lists		
🗉 📮 component/serial_manager		
e Component/uart		
device		
🖶 💻 doc		
drivers		
source		
Kan Add Files to Group 'source'		
Look in: 📜 source 🗨 🗲 🖻	* Ⅲ▼	
Name	Date modified	Туре
📜 docs	2018/9/29 11:46	File folder
📕 port_lpc5500	2018/9/29 17:30	File folder
📄 core_list_join.c	2018/5/31 10:42	C File
core_main.c	2018/9/19 11:28	C File
Core_matrix.c	2018/9/18 15:58	C File
core_state.c	2018/5/31 10:42	C File
Core_util.c	2018/5/31 10:42	C File
Figure 6. Adding files in Keil MDK		

-For IAR Embedded workbench place the files in "lpc5500\_coremark\_iar\source"

Add the files into the IAR project framework to the respective groups source by double clicking on the groups.

Workspace	<b>→</b> ‡ ×
run_in_flash_12mzh	▼
Files	🌣 • 🌣
□       □       coremark - run_in_flash_12mzh         □       □       board         □       □       CMSIS         □       □       component         □       □       device         □       □       doc         □       □       drivers         □       □       libs         □       □       linkcfg         □       □       source	
Add Files - source	oremark_iar 🕨 source 🕨
Name	Date modified
⊢⊞ i i i i i i i i i i i i i i i i i i i	2018/9/29 11:46
	2018/9/29 15:27
core_list_join.c	2018/9/29 14:46
core_main.c	2018/9/29 14:09
coreme 🦲 core_matrix.c	2018/9/29 15:04
Build ore_state.c	2018/5/31 10:42
core_util.c	2018/5/31 10:42
Mee coremark.h	2018/9/29 11:50
-For MCUXpresso place the files in "lpc550	00_coremark_mcux\source"
ld the files into the MCUXpresso project framework to the respe	ctive groups source by click the "refresh" selection
a the meeting the meetipledee project numerion on to helpe	

🔀 lpc5500_core	mark_mcux - Welco	ome page - MC	UXpress	so IDE		
File Edit Naviga	ite Search Project	ConfigTools	Run F	reeRTOS	Window	Help
📑 🗕 🖬 👘 🛛 📚	- 🔨 - 🗟 🖓 ५	> 🕘 🔻 📃	`≈   ।⊳		3.9.	e   🗟 📌   🕩 🗓
🔁 Proj 🛛 🔀 Р	eri 🚻 Regi 🎋 F	aults 🖾 Sy		📄 readı	me.txt	🎱 Welcome 🖾
		🔄   🖶 🍫   🗙	• •	$\langle \Rightarrow \Rightarrow \rangle$	🔳 🔗 🚺	e:///C:/nxp/MCUXp
E 😂 lpc5500_co	remark <run_in_fla< th=""><th>SH_12MHz&gt;</th><th><u> </u></th><th></th><th></th><th>_</th></run_in_fla<>	SH_12MHz>	<u> </u>			_
⊡ 🛞 Project S	Settings					
	ent					
						-
🕀 😕 drivers						X
🗄 😕 libs						
B 🔁 source	New		•			IDE
⊞ <u>&gt;</u> d(	Go Into					
			L			
	Open in New Window		.			
⊡ ⊡ α	Show in Local Termin	al				MCUXpresso II
🗐 🔈 💁 🖽	Сору	Ctrl+C				Cortex®-M core
	Paste	Ctrl+V				editing, compili multicore debut
🗄 🖓 📜 🗄	Delete	Delete				
⊞ <mark>.</mark> S€	Source		•			Your installation required
	Move				_	required.
M	Rename	F2			D	ocumentation
E 🖓 start 🖄	Import					For information
🕀 😕 utilitie 🖆	Export					please consult
	Pefresh	ES				« neip -> Mici
<u> </u>	Kenesh	15				Further product
() Quic (×)=	Index		•			೫ Help -> Help
	Build Targets		•	(a) Incha		
	Resource Configuration	ons	· ·	U Insta	neu 5	🔄 Properdes 🖳 🖸 🕻
			_			
dding files in MCU	Xpresso project					

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 source folder for all three tool chains and add the core\_portme.c file in the project framework under the source group.

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

File Edit View Project Flash Debug	Peripherals	Tools	SVCS V	Vindow	Help
🗋 😂 🔙 🗿 🐰 🖬 🛍 🖉 🔶 (+ -	19 m (n)	R R	建建 /	<i>≣ 11</i> ≩	🖄 start
🧼 🕮 🕮 🧼 🗮 🗰 run_in_flash_12m	nhz 💌 🞊 🛛	📥 🔁 🤇	🔶 🐡 🍅		
Project	<b>4</b>				
🖃 🏄 Project: coremark					
🖃 💭 run_in_flash_12mhz					
😥 💭 🛄 board					
E GMSIS					
component/lists					
component/serial_manage	er				
e component/uart					
device					
doc	_				
source					
core_list_join.c					
core_main.c					
core_matrix.c					
core_state.c					
core_util.c					
core_portme.c					
ee_printt.c					
utilities					
E Project Books {} Functions 0,	Templates				
/DK project workspace after adding Corel	Mark files				
server project workopade after adding objer					

forkspace     un_in_flash_12mzh     illes		Tools Window Hei	≥ip < Q > <b>\$</b> ►	e < 😳 >
an_in_flash_12mzh       ▼         Files       ▼         ■ coremark - run_in_flash_12mzh       ✓         = Board       ✓         = Board       ✓         = CMSIS       ✓         = Gevice       ✓         = Gore_monent       ✓         = Gore_monent       ✓         = Gore_list_join.c       ✓         = Gore_matrix.c       ✓         = Core_portme.c       ✓         = Core_portme.h       ✓         = Core_state.c       ✓         = Core_matrix.c       ✓         = Core_portme.h       ✓         = Core_till.c       ✓         = Core_till.c       ✓         = Core_till.c       ✓         = Startup       ✓         = W utilities       ✓	Workspace		▼ д :	×
Files Image: Coremark - run_in_flash_12mzh   Image: Dot in the second	run_in_flash_12mzh			-
Coremark - run_in_flash_12mzh  Component  Component  Component  Component  Component  Core_main.c  Core_main.c  Core_matix.c  Core_portme.c  Core_portme.c  Core_portme.c  Core_portme.h  Core_portme.h  Core_portme.h  Core_portme.h  Core_portme.h  Core_portme.c  Core_matik.c  Core_matik.c  Core_matik.c  Core_matik.c  Core_matik.c  Core_portme.h  Core	Files		۰ د	
	coremark - run_in_flash_12mzh   Board   CMSIS   Component   device   device   doc   divers   libs   libs   libs   libs   core_list_join.c   core_main.c   core_matrix.c   core_portme.c   core_portme.h   core_state.c   core_util.c   coremark.h   e e_printf.c   utilities			



A few files need to be modified to support CoreMark and are described below. In the project scatter file change the stack size as 0x2000.

```
define symbol size cstack = 0x2000;
```

To support 'printf' statements to a PC terminal, the 'core\_portme.h' file needs to be modified. Add the following line of code for ee\_printf function.

```
#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
#endif
```

In 'eeprintf.c' file, add #ifdef COREMARK\_SCORE\_TEST and the function ee\_printf(const char \*fmt,...).

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

#endif

}

This is added so that the printf code is optimized when running the  $\mu$ A/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++(AC6) tab, click 'Include path' and add the following paths that contain the header files.

Options for Target 'run_in_flash_12mhz'       2         Device   Target   Output   Listing   User       C/C++ (AC6)   Asm   Linker   Debug   Utilities	Z
Preproces       Folder Setup         Define       Setup Compiler Include Paths:         Undefine       board         Source       CMSIS         Component/serial_manager       component/serial_manager         Optimizatie       device         doc       drivers         string       Strup         Unclude       source/port_lpc5500         OK       Cancel	
OK Cancel Defaults Help	
Figure 12. Keil MDK compiler include paths	

In IAR under Project->Options-> C/C++ Compiler, click "Preprocessor" and add the following paths that contains the header files.

Category:	Factory Settings
General Ontions	Multi-file Compilation
Static Analysis	Discard Unused Publics
Runtime Checking	Disgnastica MISDA C-2004 MISDA C-1008 Encodings Extra Options
C/C++ Compiler	Language 1 Language 2 Code Optimizations Output List Preprocessor
Assembler	Language I Language Z Code Optimizations Output List
Output Converter	Ignore standard include directories
Custom Build	
Build Actions	Additional include directories: (one per line)
Linker	\$PR0J_DIR\$/strc
Debugger	
Simulator	\$PROJ_DIR\$/source
CADI	\$PROJ_DIR\$/source/port_Ipc5500
CMSIS DAP	Preinclude file:
GDB Server	
I-jet/JTAGjet	
J-Link/J-Trace	Defined symbols: (one per line)
TI Stellaris	DEBUG CPLL L DC55560 IBD100 cm33 c
Nu-Link	RUN IN 12MHZ
PE micro	RUN_IN_FLASH
ST-LINK	
Third-Party Driver	
TI MSP-FET	
11 XDS	

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

In MCUXpresso under "Properties for xxxx"->C/C++ Build-> Settings->, click "Includes" and add the following paths that contains the header files.



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 run\_in\_ram\_xxmhz 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:

 $. \label{linear} . \label{linear} \label{linear}$ 

The linker script setting for run\_in\_ramx\_xxmhz project is shown in Fig 15

#### Integration of CoreMark library to SDK2.0 framework

	t Output Listing Lloor C/C++	(ACG) App Linker Dobug Liti	ition	
evice   Targe		(AC6)   Asin Linker   Debug   Ou	liles	_
🔲 Use Mem	ory Layout from Target Dialog	X/O Base		
Make	RW Sections Position Independe	nt R/O Base	0x0000000	_
Make	RO Sections Position Independent	nt P/W Boss	0x20040000	_
Don't	Search Standard Libraries	N/W Dase		
Repo	rt 'might fail' Conditions as Errors	disable Warnings		
Scatter File	LPC55S69_cm33_core0_ramx.s	cf	<b>.</b>	Edit
Scatter File	LPC55S69_cm33_core0_ramx.s	cf	<b>•</b>	Edit
Scatter File Misc controls	LPC55S69_cm33_core0_ramx.s	cf	<b>.</b>	Edit
Scatter File Misc controls Linker control string	LPC55S69_cm33_core0_ramx.s libs/keil_lib_power.lib remove cpu=Cortex-M33 *.o library_type=microlibstricts libs/keil_lib_power.libremove	cf catter "LPC55S69_cm33_core0_ram» -summary_stderrinfo summarysize		Edit

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, as Fig 18 shows, above #include in all five files.

These CoreMark files are labeled as their own IAR EWARM linker "section". The provided .icf linker file in .\lpc5500\_coremark\_iar \LPC55S69\_cm33\_core0\_ramx.icf

then places this section, which is called "critical\_text" into SRAMX. To do this, add the following line of code in icf file, as shown in Fig 16.

if (isdefinedsymbol(USE_DLIB_PERTHREAD))	)
<pre>1 /* Required in a multi-threaded applicat</pre>	tion */
initialize by copy with packing = none	section DLIB PERTHREAD 1:
}	(
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	<pre>{ last block HEAP };</pre>
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,
	iw object core_utii.o,
initialize by conv	/ object core portme o
initialize by copy	object core main.o.
	object core list join.o.
	object core matrix.o.
	object core state.o,
	object core util.o,
	······································

Figure 16. IAR EWARM allocate Code to SRAM area

Category:	Eactory Setting
calogoly.	
General Options	
Static Analysis	
Runtime Checking	#define Diagnostics Checksum Encodings Extra Option
C/C++ Compiler	Config Library Input Optimizations Advanced Output List
Assembler	I inker configuration file
Output Converter	
Custom Build	
Build Actions	\$PROJ_DIR\$/LPC55S69_cm33_core0_flash.icf
Linker	
Debugger	Edit
Simulator	
CADI	Configuration file symbol definitions: (one per line)
CMSIS DAP	
GDB Server	
I-jet/JTAGjet	
J-Link/J-Trace	
TI Stellaris	
Nu-Link	
PE micro	
ST-LINK	
Third-Party Driver	
TI MSP-FET	
TI XDS	

<pre>#if RUN_IN_RAMX</pre>	
<pre>#pragma default_function_attributes = @ .critical_code _ #endif</pre>	41
ب	Ŧ
Added below code in the files end	
#if RUN_IN_RAMX	
<pre>#pragma default_function_attributes =</pre>	
- #endif	
Figure 18. IAR EWARM #pragma command	

For MCUXpresso to execute CoreMark in Internal SRAM, just selected the linker file as "lpc5500\_coremark\_RUN\_IN\_SRAMX.ld" in "Managed Linker Script", as shown in Fig 19..



## 2.2 Optimizing the CoreMark framework

There are many factors that affect the CoreMark and  $\mu$ A/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 v5.28

IAR EWARM 8.40.2

MCUXpresso 11.0.1\_2563

## 2.2.1 Memory considerations

Due to the inherent architecture of SRAM and flash, CoreMark executes faster when running out of SRAM. The LPC55xx internal memory uses a multilayer AHB matrix system that provides a separate instruction and data bus for Cortex-M33 and SRAMX bank. See Fig 20. SRAM0 to SRAM4 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  $\mu$ A/MHz test, it is possible to save power by disabling the prefetch ability of flash. The LPC55xx user manual contains more information on configuring the flash memory, such as the minimum number 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.



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  $\mu$ A/MHz test. If this macro is defined, the CoreMark score test runs. If this macro is commented out, the  $\mu$ A/MHz test runs. Use this macro to switch between the two benchmarks.

## 2.2.2 IDE Optimization Setting

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++(AC6) tab, the optimization level needs to be set as "-mcpu=Cortex-m33 --target=arm-arm-none-eabi -Omax -g -mthumb - mfpu=fpv5-sp-d16 -mfloat-abi=hard -fno-common -ffp-mode=fast" in Misc Ctonrols.

Device   Target   Output   Listing   User	C/C++ (AC6) Asm Linker Debug Utilities
Preprocessor Symbols Define: DEBUG, CPU_LPC55S69, Undefine:	JBD100_cm33_core0, RUN_IN_12MHZ, RUN_IN_FLASH,COREMARK_SCORE_TEST
Language / Code Generation Execute-only Code Optimization: -03 Link-Time Optimization Split Load and Store Multiple One ELF Section per Function	Warnings:       All Warnings       Language C:       c99         Turn Warnings into Errors       Language C++:       c++11         Plain Char is Signed       Short enums/wchar         Read-Only Position Independent       use RTTI         Read-Write Position Independent       No Auto Includes
Include Paths Misc Controls Compiler control string Nisc -xctarget=arm-arm-none -xc -std=c99target=arm -fno-rtti -ftuo -funsigned-ch -D_MICROLIB -mlittle-er	iponent/lists;component/serial_manager;component/uart;device;doc;drivers;src;stz e-eabi -mfpu=fpv5-sp-d16 -mfloat-abi=hard -Wno-pedantic -Wno-padded -Wno-unuse -arm-none-eabi -mcpu=cortex-m33 -mfpu=fpv5-sp-d16 -mfloat-abi=hard -c iar -fshort-enums -fshort-wchar ndian -O3 -ffunction-sections -Weverything -Wno-packed -Wno-reserved-id-macro
	OK Cancel Defaults Help

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.

Options for Target 'run_in_ramx_96mhz'
Device   Target   Output   Listing   User C/C++ (AC6)   Asm   Linker   Debug   Utilities
Preprocessor Symbols Define: DEBUG, CPU_LPC55S69JBD100_cm33_core0, RUN_IN_96MHZ, RUN_IN_RAMX Undefine:
Language / Code Generation       Warnings: All Warnings Language C: c99         Execute-only Code       Warnings: All Warnings To Errors         Optimization: -00       Turn Warnings into Errors         Link-Time Optimization       Plain Char is Signed         Split Load and Store Multiple       Read-Only Position Independent         One ELF Section per Function       Read-Write Position Independent
Include Paths       board;source;CMSIS;component/lists;component/serial_manager;component/uart;device;doc;drivers;src;stz          Misc Controls       -xctarget=arm-arm-none-eabi -mfpu=fpv5-sp-d16 -mfloat-abi=hard -Wno-pedantic -Wno-padded -Wno-unuse         Compiler control string       -xc -std=c99target=arm-arm-none-eabi -mcpu=cortex-m33 -mfpu=fpv5-sp-d16 -mfloat-abi=hard -c -fno-rtti -funsigned-char -fshort-enums -fshort-wchar -D_MICROLIB -mlittle-endian -gdwarf-3 -O0 -ffunction-sections -Weverything -Wno-packed -Wno-reserved-
OK Cancel Defaults Help
Figure 22. 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



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

Category:	Factory Settings
General Options	Multi-file Compilation
Static Analysis	Discard Unused Publics
Runtime Checking	Diagnostica MISDA C:2004 MISDA C:1998 Encodings Extra Options
C/C++ Compiler	Language 1 Language 2 Code Optimizations Output List Proprocesse
Assembler	Canguage 1 Language 2 Code Opamizations Output List Preprocesso
Output Converter	Level inabled transformations:
Custom Build	None     Common subexpression elimination
Build Actions	Loop unrolling
Linker	Function inlining
Debugger	Code motion
Simulator	High Static clustering
CADI	Speed
CMSIS DAP	Vectorization
GDB Server	
I-jet/JTAGjet	
J-Link/J-Trace	
TI Stellaris	
Nu-Link	
PE micro	
SI-LINK	
Third-Party Driver	
II MSP-FET	
TIVDC	

### 2.2.2.3 MCUXpresso Optimization

There are two compiler optimizations that can be done to improve CoreMark score. Set the optimization level to "-O3" so please select "Optimize most(-O3)" from the drop down menu.



When benchmarking the power consumption of the MCU, the optimization level should be set to "None(-O0)"



# 3 Measuring CoreMark on board

## 3.1 LPC55S69Xpresso board

The LPC55S69Xpresso board supports a VCOM serial port connection via **P6**. To observe debug messages from the board set the terminal program to the appropriate COM port and use the setting '115200-8-N-1-none'. To make the debug messages easier to read, the new line receive setting should be set to auto.

## 3.2 Board Setup

The LPC55S69 Rev A1 development board is used for benchmarking



The board ships with CMSIS-DAP debug firmware programmed. Visit the following FAQ for more information on CMSIS\_DAP debug firmware: https://www.nxp.com/downloads/en/software/lpc\_driver\_setup.exe For debugging and terminal debug messages, connect a USB cable to P6 USB connector. Board schematics are available on www.nxp.com.

#### 3.2.1 µA/MHz measurement setup

To measure the LPC5500 power consumption, remove R92, install header at P13, and connect ammeter across P13 as shown in Figure 28.

NOTE

The current data on EVK maybe little higher than datasheet, due to the EVK have more other components may cost more power.



If we need measurement the MCU core current, we need rework the board by removing the R92. Then we can measure the current through P13 by multimeter.

While performing the  $\mu$ A/MHz benchmark, use P6 USB connector to provide power to the board. After the  $\mu$ A/MHz benchmark project has been downloaded, power cycling the board by removing the USB cable, and reinsert to make sure that the debug probe is not connected.

The baud rate setting for debug messages is 115200. It can be changed in core\_potme.c file.

Line209 config.baudRate\_Bps = 115200;

Similarly, by selecting different configuration projects in workspace window, the core clock frequency can be changed. Each of the configuration may enable below defined project configuration settings:

RUN\_IN\_12MHZ

RUN\_IN\_48MHZ

RUN\_IN\_96MHZ

RUN\_IN\_150MHZ

## 3.3 Run CoreMark code

The first step to get CoreMark result is to connect the connector P6 of the board with PC. Then the PC recognizes the LPC-Link2 debugger with a Simulate Serial Port as shown in Fig 29.

If PC cannot find the serial port driver, download the LPCScrypt from below link, and install on your PC.

https://www.nxp.com/support/developer-resources/software-development-tools/lpc-developer-resources-/lpc-microcontroller-utilities/lpcscrypt-v2.0.0:LPCSCRYPT?tab=Design\_Tools\_Tab

Device Manager
File Action View Help
Monitors   Monitors  Metwork adapters  Bluetooth Device (Personal Area Network) #2  Bluetooth Device (RFCOMM Protocol TDI) #2  Dintel(R) Dual Band Wireless-AC 8260  Dintel(R) Ethernet Connection 1219-LM  Dintel(R) Ethernet Connection 1219-LM  Dintel(R) Ethernet Adapter Manager  Microsoft Virtual WiFi Miniport Adapter  VirtualBox Host-Only Ethernet Adapter  VirtualBox Host-Only Ethernet Adapter #2  Ports (COM & LPT)  Figure Comment Technology - SQL (2013)  Figure Processors  Figure Proximity Devices  Figure Proximity Devices  Figure Smart card readers
Sound, video and game controllers
System devices
ure 29. LPC-Linkll UCom Port

Open a UART debug terminal (like Tera Term, putty, etc.), and configure as 115200, 8 data bits, no parity, 1 stop bit, refer Fig 30.

File Edit Setup Cont	rol Window Help	0	X	
Tera Ter	m: Serial port setup	D		<u>_</u>
Pi Bi Di Pi Si	ort: aud rate: ata: arity: top:	COM241 • 115200 • 8 bit • none • 1 bit •	OK Cancel Help	
FI	ow control: Transmit delay 0 msec	none -	sec/line	

Once the CoreMark necessary files are added into the project (by following Chapter 2.1 instructions), compile the project and download to the LPC5500Xpresso board.

Click reset button, the CoreMark benchmark prints on the terminal after a few seconds, like Fig 31 in Chapter 4.

# 4 Result

Figure 31 shows the CoreMark benchmark result when running LPC5500 at 96 MHz core frequency in IAR. The CoreMark benchmark score is the number of iterations per second. The CoreMark/MHz score executing from internal flash for this run is 372.786580/96 MHz = 3.883 CoreMark/MHz.

File Edit Setup	Control Window Help	
SystemCoreClo	k: 9600000	A
ystem Running	g on RAM-X	
2K performance	run parameters for coremark.	
CoreMark Size	: 666	
otal ticks	: 10730	
otal time (se	ecs): 10.730000	
[terations/Sec	: 372.786580	
[terations	: 4000	
Compiler vers	on : IAR 8.30.1	
Compiler flag	: High, Speed, No size constraints	
Memory location	on : STACK	
seedcrc	: 0xe9f5	
0]crclist	: 0xe714	
0]crcmatrix	: 0x1fd7	
0]crcstate	: 0x8e3a	
[0]crcfinal	: 0x65c5	
Correct operat	ion validated. See readme.txt for run a	nd reporting rules.
CoreMark 1.0	372.786580 / IAR 8.30.1 High, Speed, N	o size constraints / ST
ACK		
ONE		

#### Figure 31. CoreMark result

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

#### Table 1. LPC55S69Xpresso board CoreMark/MHz Score

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

NOTE

Test under 96 MHz

For  $\mu$ A/MHz, following tables show typical results when running on the LPCXpresso55S69 board with VDD = 3.3 V at room temperature. Fig 24 compares the three IDEs in terms of power consumption.

NOTE The current data on EVK maybe little higher than datasheet, due to the EVK have more other components may cost more power.

NOTE

The average current in 150MHz will higher than other modes, the reason is 150Mhz will enable PLL, the PLL cost more power.

#### Table 2. Keil MDK µA/MHz score

Frequency	Avg. Power Consumption (mA, SRAM X)	μΑ/MHz Score (SRAM X)	Avg. Power Consumption (mA, Flash)	μA/MHz Score (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

#### Table 3. IAR EWARM µA/MHz score

Frequency	Avg. Power Consumption (mA, SRAM X)	μΑ/MHz Score (SRAM X)	Avg. Power Consumption (mA, Flash)	μΑ/MHz Score (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

#### Table 4. MCUXpresso $\mu$ A/MHz score

Frequency	Avg. Power Consumption (mA, SRAM X)	μΑ/ΜΗz Score (SRAM X)	Avg. Power Consumption (mA, Flash)	μΑ/MHz Score (Flash)
12 MHz	1.33	110.84	1.19	115.84
48 MHz	2.40	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



# **5** Conclusion

Three types of CoreMark benchmarking on the LPC55xx are presented in this document with different IDEs (Keil, IAR, MCUXpresso):

CoreMark score, power consumption, and µA/MHz.

It also describes how to optimize the benchmark results when running the benchmark out of internal SRAM and flash.

The CoreMark results are measured on LPCXpresso55S69. The best CoreMark number is 4.021, achieved by using KEIL MDK(Arm Compiler 6.12) and running CoreMark from SRAM X. The best CoreMark power consumption in  $\mu$ A/MHz is 37.30, achieved by running CoreMark from flash when core frequency is 96 MHz.

# 6 Reference

- 1. CoreMark Benchmarking for ARM Cortex Processors, ARM
- 2. AN11811 LPC5411x CoreMark Cortex-M4 Porting Guide,NXP
- 3. UM11126\_LPC55xx/LPC55Sxx User Manual ,NXP

# 7 Revision history

Revision history				
Rev.	Date	Substantial changes		
0	25 January, 2019	Initial reversion		
1	December, 2019	Updated CoreMark scores on silicon '1B' with SDK2.6.3 and ad MHz CoreMark		

#### How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

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/

While NXP has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer's applications and products, and NXP accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX, EMBRACE, GREENCHIP, HITAG, I2C BUS, ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, AltiVec, C-5, CodeTEST, CodeWarrior, ColdFire, ColdFire+, C-Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, UMEMS, EdgeScale, EdgeLock, eIQ, and Immersive3D are trademarks of NXP B.V. All other product or service names are the property of their respective owners. AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, µVision, Versatile are trademarks or registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

#### <sup>©</sup> NXP B.V. 2019.

#### All rights reserved.

For more information, please visit: http://www.nxp.com For sales office addresses, please send an email to: salesaddresses@nxp.com

> Date of release: December 2019 Document identifier: AN12284

# arm