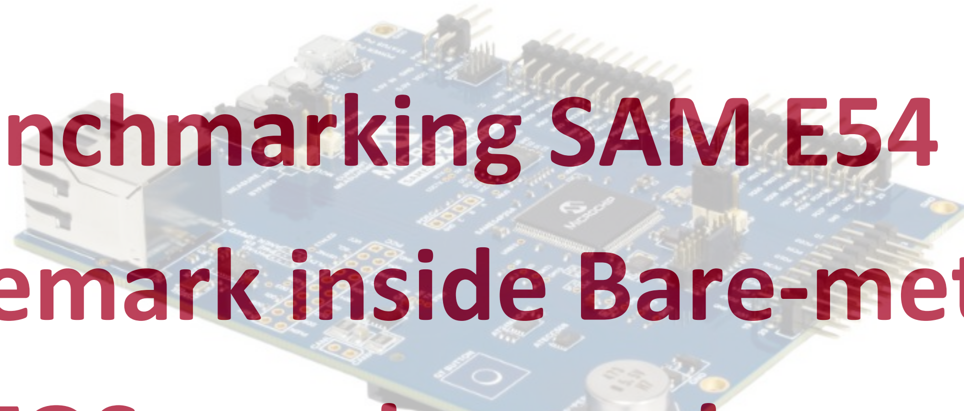
Harokopio University
Department of “Informatics and Telematics”
“Embedded Systems”



Benchmarking SAM E54 using Coremark inside Bare-metal and RTOS running environments.

01/10/2020

Plevritakis Nikolaos - Reg. Number: 18506

Introduction: Coremark [1]

- ✔ Popular, free, open source tool for benchmarking CPUs.
- ✔ Written on C programming language.
- ✔ Flexible with build configuration options (defines).
- ✔ Supports any CPU architecture from 8-bits to 64-bits data bus width.
- ✔ Needs 2000 bytes (typical) of RAM.
- ✔ FPU support.
- ✔ Find and sort list, matrix manipulation, state machine and CRC algorithms are implemented.
- ✔ Numeric score for easy comparisons.

Test-bed environments and compiler configuration

❖ Coremark is used as an application code for 2 testbed environments:

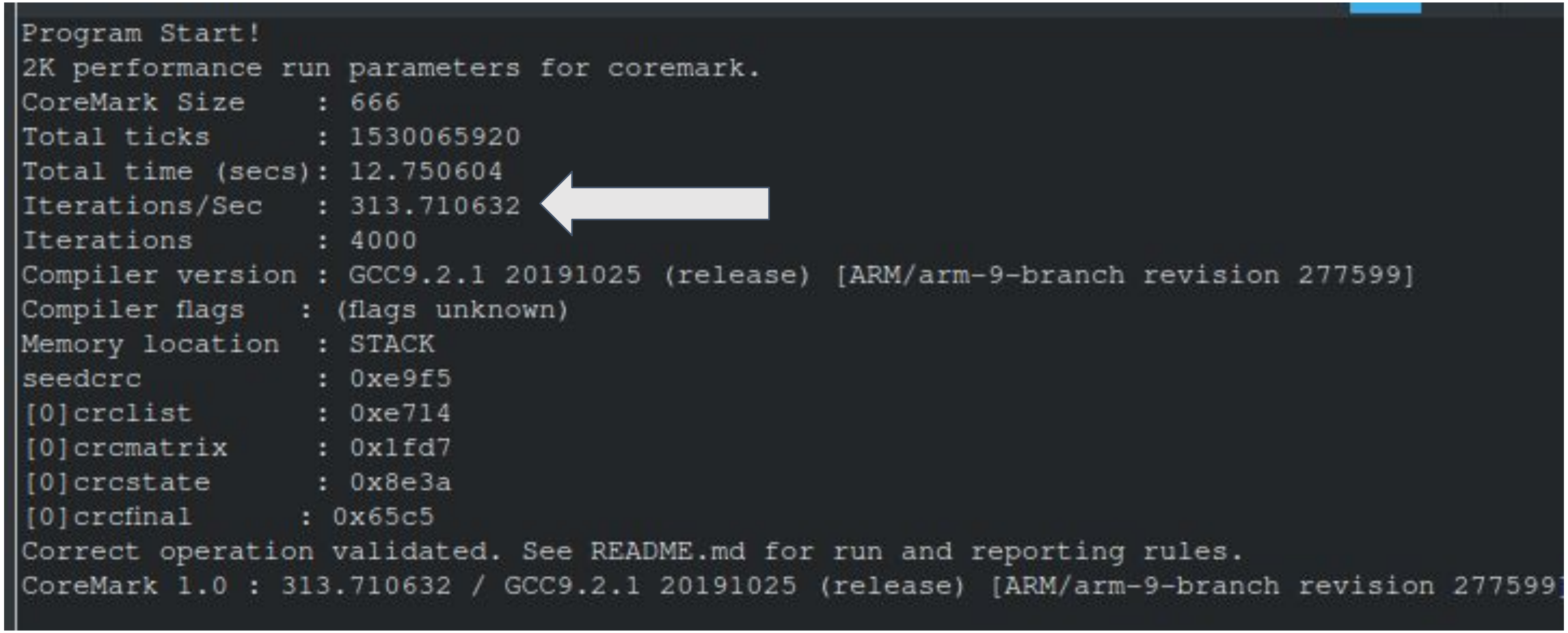
Bare-metal ➔ Simplest running environment for 1 thread only nor **non** preemptive "scheduling" using infinite main loop and non-blocking application programming techniques.

RTOS [3] ➔ Complex running environment for multithreading and preemptive scheduling.

- Coremark is configured for testing both CPU and the hardware FPU of SAM E54 microcontroller using stack memory allocation.
- The SAM E54 **[4]** is an ARM Cortex M4F microcontroller configured at 120 MHz clock speed with 4KB cache enabled and auto Flash Wait states.
- GCC compiler is used with O3 optimization and HardFloat enabled for maximum performance and HW FPU usage.

Bare-metal Results

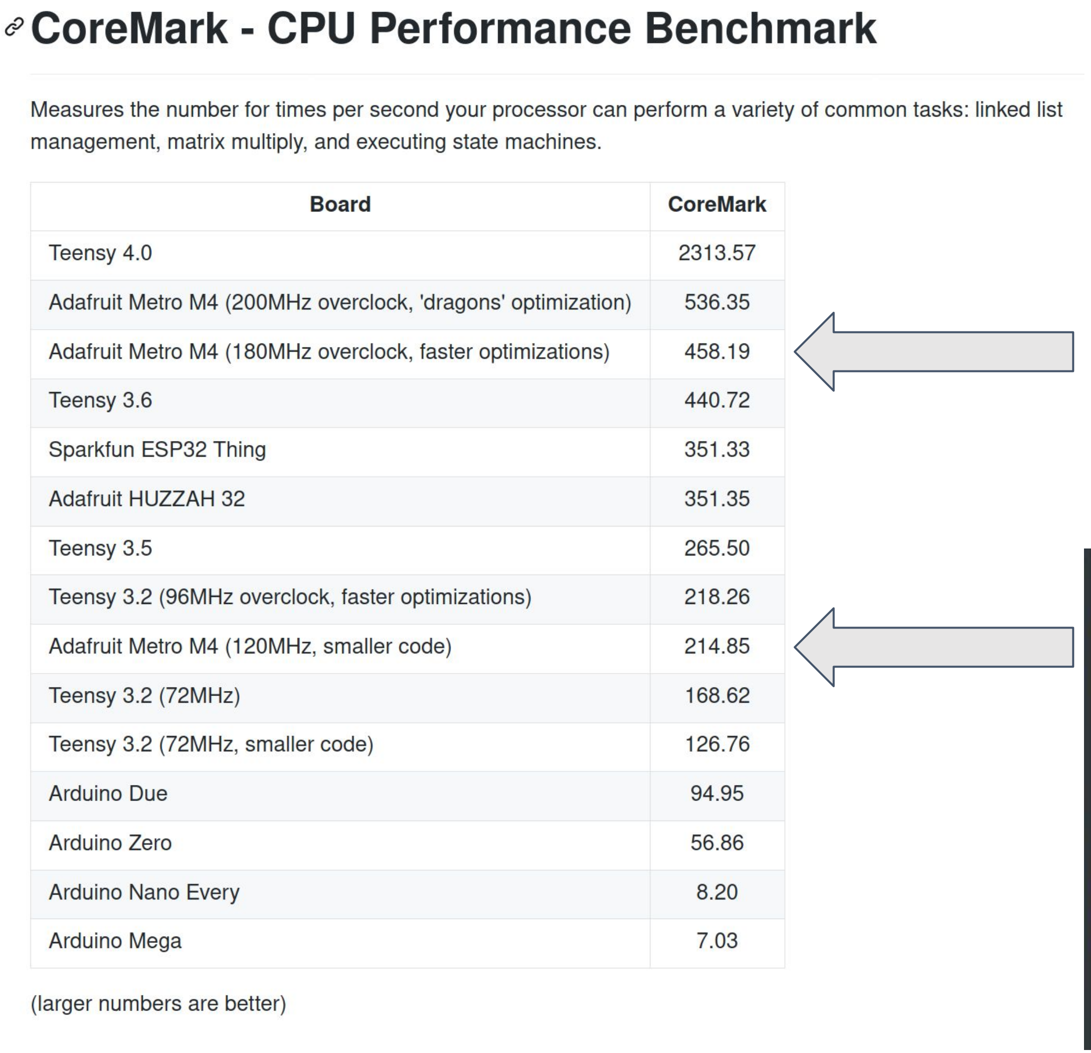
Bare-metal Setup No2 @ 120Mhz with GCC Optimization O3, HardFloat enabled:



```
Program Start!
2K performance run parameters for coremark.
CoreMark Size     : 666
Total ticks       : 1530065920
Total time (secs): 12.750604
Iterations/Sec    : 313.710632
Iterations        : 4000
Compiler version : GCC9.2.1 20191025 (release) [ARM/arm-9-branch revision 277599]
Compiler flags    : (flags unknown)
Memory location   : STACK
seedcrc           : 0xe9f5
[0]crclist        : 0xe714
[0]crcmatrix      : 0x1fd7
[0]crcstate       : 0x8e3a
[0]crcfinal      : 0x65c5
Correct operation validated. See README.md for run and reporting rules.
CoreMark 1.0 : 313.710632 / GCC9.2.1 20191025 (release) [ARM/arm-9-branch revision 277599
```

Bare-metal Coremark score is: 313.710632

Bare-metal Reference Results [2]



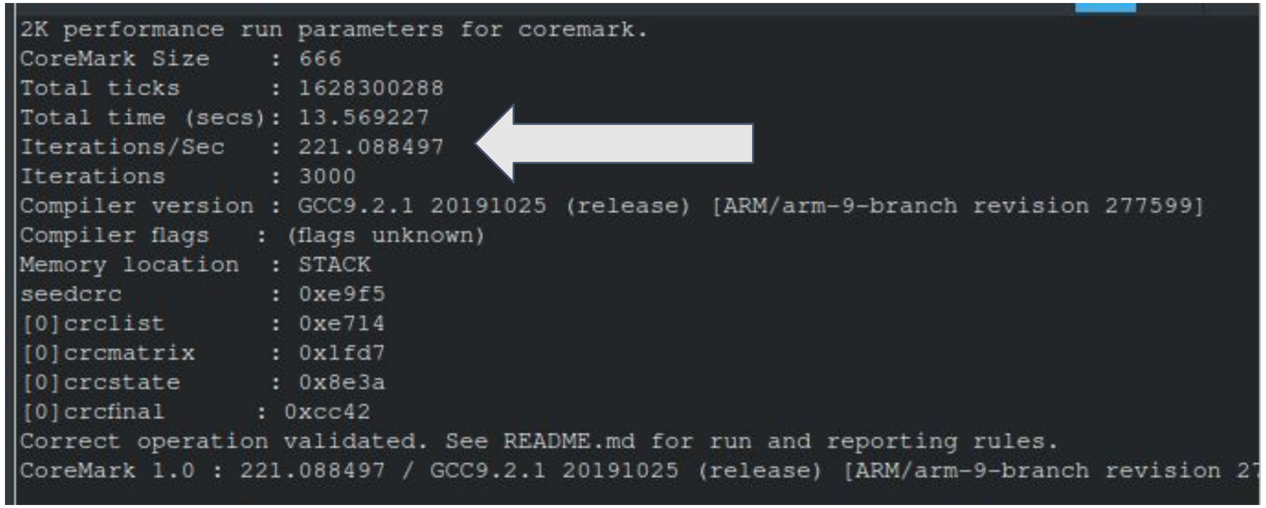
CoreMark - CPU Performance Benchmark

Measures the number for times per second your processor can perform a variety of common tasks: linked list management, matrix multiply, and executing state machines.

Board	CoreMark
Teensy 4.0	2313.57
Adafruit Metro M4 (200MHz overclock, 'dragons' optimization)	536.35
Adafruit Metro M4 (180MHz overclock, faster optimizations)	458.19
Teensy 3.6	440.72
Sparkfun ESP32 Thing	351.33
Adafruit HUZZAH 32	351.35
Teensy 3.5	265.50
Teensy 3.2 (96MHz overclock, faster optimizations)	218.26
Adafruit Metro M4 (120MHz, smaller code)	214.85
Teensy 3.2 (72MHz)	168.62
Teensy 3.2 (72MHz, smaller code)	126.76
Arduino Due	94.95
Arduino Zero	56.86
Arduino Nano Every	8.20
Arduino Mega	7.03

(larger numbers are better)

Bare-metal Setup No1 @ 120Mhz with GCC Optimization O1, HardFloat disabled (no HW FPU usage) for comparison:



```
2K performance run parameters for coremark.
CoreMark Size    : 666
Total ticks      : 1628300288
Total time (secs): 13.569227
Iterations/Sec   : 221.088497
Iterations       : 3000
Compiler version : GCC9.2.1 20191025 (release) [ARM/arm-9-branch revision 277599]
Compiler flags   : (flags unknown)
Memory location  : STACK
seedcrc          : 0xe9f5
[0]crclist       : 0xe714
[0]crcmatrix     : 0x1fd7
[0]crcstate      : 0x8e3a
[0]crcfinal     : 0xcc42
Correct operation validated. See README.md for run and reporting rules.
CoreMark 1.0 : 221.088497 / GCC9.2.1 20191025 (release) [ARM/arm-9-branch revision 27
```

RTOS Results at 1KHz and 10KHz

1KHz with time-slice disabled (2 Threads sequential run)

Thread1 Iterations/Sec : 311.704288

Thread2 Iterations/Sec : 311.696671

mean: 311.7004795

RTOS overhead vs Bare-Metal: 311.7004795 / 313.710632 => **0,641%**

1 KHz Tick rate (1ms) with time slice enabled (2 Threads)

Thread1 Score, Iterations/Sec : 155.498741

Thread1 Score, Iterations/Sec : 155.498883

TOTAL all threads (SUM) score is 310.9976201

Time-slice context switch overhead: 310.997620 / 311.7004795 => **0,257%**

10 KHZ with time-slice disabled (2 Threads sequential run)

Iterations/Sec : 309.171110

Iterations/Sec : 309.201471

mean: 309.1862905

RTOS added overhead vs Bare-Metal: 309.1862905 / 313.710632 => **1,44%**

10 KHz Tick rate (100us) with time slice enabled (2 Threads)

Iterations/Sec : 152.027356

Iterations/Sec : 152.027307

TOTAL all threads score is 304.054687

Time-slice extra context switch overhead for 2 threads: 304.0546/309.1862 => **1,6597%**

10 KHz Tick rate (100us) with time slice enabled (3 Threads)

Iterations/Sec : 101.323983

Iterations/Sec : 101.325034

Iterations/Sec : 101.325034

TOTAL all threads score is 303.974060

Time-slice context switch overhead for 3 threads: 303.9740/309.1862 => **1.6858%**

RTOS Results at 100 KHz

100KHz with time-slice disabled (2 Threads sequential run)

Iterations/Sec : 285.299670

Iterations/Sec : 284.902143

mean: 285.100891

RTOS added overhead vs Bare-Metal: 285.100891 / 313.710632 => **9,12%**

100 KHz Tick rate (10us) with time slice enabled (2 Threads)

TOTAL all threads score is 230.947723

100 KHz Tick rate (10us) with time slice enabled (3 Threads)

TOTAL all threads score is 231.022888

100 KHz Tick rate (10us) with time slice enabled (4 Threads)

TOTAL all threads score is 231.188141

Time-slice extra context switch overhead for 2 threads: **18,99%**

Time-slice extra context switch overhead for 3 threads: **18,968%**

Time-slice extra context switch overhead for 4 threads: **18,96%**

RTOS Results at 200 KHz

200KHz with time-slice disabled (2 Threads sequential run)

Iterations/Sec : 258.702799

Iterations/Sec : 257.901446

mean score: 258,302124

RTOS added overhead vs Bare-Metal: 258,302124 / 313.710632 => **17,66%**

200 KHz Tick rate (5us) with time slice enabled (2 Threads)

TOTAL all threads score is 149.477417

200 KHz Tick rate (5us) with time slice enabled (3 Threads)

TOTAL all threads score is 149.545410

200 KHz Tick rate (5us) with time slice enabled (4 Threads)

TOTAL all threads score is 149.618805

Time-slice extra context switch overhead for 2 threads: **42,10%**

Time-slice extra context switch overhead for 3 threads: **42,08%**

Time-slice extra context switch overhead for 4 threads: **42,05%**

Conclusion

- Bare-Metal has the fastest score versus RTOS scores because there are no CPU systick interrupts.
- Running the same code (Coremark) more than one threads is easy on a preemptive capable RTOS.
- The higher the RTOS ticks rate the more CPU is used by RTOS kernel.
- FreeRTOS overhead CPU usage is less than 2% when tick rates is 10 KHz.
- FreeRTOS overhead CPU usage is 10% and more when tick rates are 100 KHz and higher at 120 MHz CPU clock.
- FreeRTOS time-slice if disabled no round-robin scheduling is used on equal priority tasks (threads) so they will be executed serialy.
- No extra CPU scheduling overhead happens when more than 2 threads are running.
- FreeRTOS round-robin scheduling algorithm does not guarantee exactly equal CPU time is going to be shared between tasks with equal priority and time-slice feature enabled.

WHERE?
WHEN? WHY?
WHEN?
WHY?
WHO? WHERE? WHAT? HOW?
WHO?
WHAT? HOW? WHY?
WHEN?
WHAT?
WHERE?
WHO?
WHEN?
HOW? HOW?
WHEN?
WHERE? WHICH? WHOSE? WHEN? WHY?
WHOSE?
WHEN? WHERE?
HOW?
HOW?
WHERE? WHICH? WHOSE? WHEN? WHY?
WHICH?
WHOSE?
WHAT? HOW? WHY?
WHO? WHERE? WHAT? HOW?
WHY? HOW?
WHY?
WHOSE?
WHOSE? WHEN?
WHAT? HOW? WHY?
WHO? WHERE? WHAT? HOW?
WHERE? WHICH? WHOSE? WHEN? WHY?
WHAT?
WHY? HOW? WHERE?
WHO? WHOSE?
WHERE? WHAT? HOW?
WHAT?
WHY?
WHO?
HOW?
WHERE?
WHERE? WHAT? HOW?
WHERE? WHICH? WHOSE? WHEN? WHY?
HOW? WHERE? WHY?
WHERE?
HOW?
HOW? WHO?
WHO? WHERE? WHAT? HOW?
WHAT? HOW? WHY?
WHERE?
WHAT?
HOW?
HOW?
WHAT?
HOW?
WHERE?

References

[1] - https://www.eembc.org/coremark/

[2] - https://github.com/PaulStoffregen/CoreMark

[3] - https://www.freertos.org

[4] - https://www.microchip.com/wwwproducts/en/ATSAME54P20A

[5] - https://www.microchip.com/DevelopmentTools/ProductDetails/PartNO/ATSAME54-XPRO