An Open-Source Multi-Platform SDK for Programming SCuM

Alexandre Abadie
alexandre.abadie@inria.fr
Inria
Paris, France

Titan Yuan
titan@berkeley.edu
UC Berkeley
Berkeley, CA, USA

Thomas Watteyne
thomas.watteyne@inria.fr
Inria
Paris, France

Filip Maksimovic
filip.maksimovic@inria.fr
Inria
Paris, France

ABSTRACT

We introduce the SCuM SDK, an open source and multi-platform software development kit that can be used to build and load a firmware on the SCuM chip. The SCuM SDK is designed to help newcomers, especially those with prior experience with ARM, to start programming SCuM. The SCuM SDK is fully open-source and can program, boot and calibrate SCuM much faster than the existing tools.

CCS CONCEPTS

• **Computer systems organization → Embedded software**.

KEYWORDS

SCuM, SDK, microcontroller, build system

1 INTRODUCTION

The single-chip micro mote (SCuM) is an ARM Cortex-M0-based microcontroller that operates without an external crystal oscillator [3]. SCuM is the first crystal-free chip that can interoperate with radios while maintaining standards compatibility. With no external components needed, it allows for sub-millimeter-scale form factor devices and brings us one step closer to realizing the Smart Dust vision.

The crystal-free nature of SCuM requires that much care be taken with respect to the calibration of the internal clocks of the chip. In fact, by default and because of external conditions, the actual internal clock frequency cannot be precisely known in advance by software. A non-calibrated clock might lead to non-functional transfers over radio or UART, which require very accurate clock frequencies for sampling data. Due to the lack of a crystal, calibration has to be performed using an accurate external clock source to tune the internal clock speed. This accurate calibration clock source can be provided by a radio source, as Maksimovic *et al.* demonstrated [2], or, more basically, from an external device directly connected to SCuM. Despite these intrinsic limitations, Chang *et al.* have demonstrated SCuM usability for the Internet of Things [1].

So far, to build a firmware usable with SCuM, only the ARM Keil μVision integrated development environment (IDE) can be used. This IDE generates sufficiently small binaries, but is proprietary and limits binary sizes on the free version. Moreover, using it is restricted to users with a Windows computer. Alternative initiatives exist [5] but they rely on wrapping ARM Keil μVision with the Wine emulator and can only be used on a Mac.

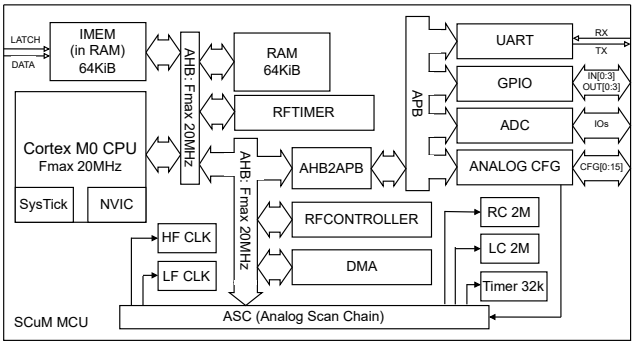

Figure 1: SCuM chip block diagram.

In order to improve the User eXperience (UX) when programming the SCuM chip, this paper introduces the SCuM SDK[1], a fully open-source Software Development Kit (SDK). The SCuM SDK can be used regardless of the operating system used by developers and is based on open-source, freely available, and up-to-date standard development tools. Compared to the current workflow, the SCuM SDK provides developers with an optimized programming workflow, taking the best of all the programming components involved. This results in faster compilation and load times of a firmware in SCuM.

2 BACKGROUND

The hardware design of SCuM is described in detail in [4]. Fig. 1 reproduces the SCuM block diagram from a higher-level point of view, which is closer to the one that a firmware developer would look at. An important aspect of SCuM is that it does not have any dedicated flash memory, so all the instruction code is stored in volatile RAM memory. This has two implications. First, and most importantly, the instruction code is lost after power-cycling the chip, second, the firmware is directly loaded in the memory using a shift register wired to an external programmer device. This allows us to bypass the absence of JTAG in SCuM.

As depicted in Fig. 2, the SCuM programmer is two-fold. On the SCuM side, a Nordic nRF52840-DK programmer board is connected to several SCuM GPIOs. They are used to load the instruction code into RAM via the shift register and to boot and calibrate SCuM. On

[1] As an online addition to this paper, the SCuM SDK is published under an open-source BSD license at https://github.com/PisterLab/scum-sdk.

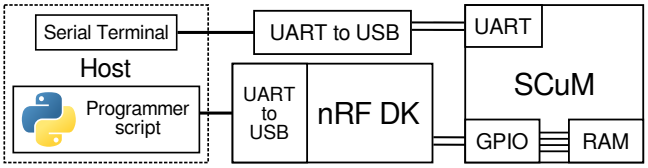

Figure 2: SCuM Programmer Setup.

the host computer, a Python script communicates with the UART to USB converter embedded in the nRF52840-DK programmer to transfer the firmware. The whole firmware is sent at once over UART after being previously padded with null bytes by the Python script to fit the 64 kB memory size. The nRF52840-DK platform is used as programmer because it is very popular, easy to program, and as a consequence widely used in the Internet of Things community. This strategy, although functional, is very inefficient because the UART baudrate is limited, e.g. to 400 kbit/s. A better strategy would be to bypass the UART interface and directly use the USB peripheral of the nRF52840 microcontroller to transfer firmware blocks at 480 Mbps. Lastly, the SCuM standard input/output is exchanged via UART at a fixed 19200 bauds to the host using a dedicated UART-to-USB converter.

3 THE SCUM SDK

The SCuM SDK uses the ARM standard way of defining the memory addresses and registers bitfields and of describing peripherals. This is done by including the ARM CMSIS headers from the newly unique `scum.h` header file. As a result, to access the memory definitions, peripheral registers, and usual ARM primitives, only `scum.h` has to be included.

Optimized Programmer. The programmer code is provided in the SCuM SDK repository. The firmware transfer protocol sends the binary firmware, split in 1024 B chunks, between the Python tool and the programmer board. The 1024 B size is a good trade-off between transfer speed and memory usage on the programmer board . The remaining null bytes are padded directly by the programmer board, via the SCuM shift register. This strategy is much faster because fewer bytes are sent over UART. Furthermore, the SCuM SDK programmer uses fewer calibration pulses, from 25 to 10, to calibrate SCuM.

Regardless of the size of the firmware, the legacy programmer takes 7.158 s to load a firmware, to boot it, and to calibrate SCuM. The calibration step itself used to take 3.099 s of the total time when using 25 calibration pulses. Now, the calibration step decreases to 1.1 s. With the new protocol and reduced calibration steps used by the SCuM SDK programmer, the total bootload duration of a 10 kB firmware decreases to 1.672 s.

Build System. The SCuM SDK build system is multi-platform thanks to the combination of CMake and Ninja tools. The GNU ARM embedded toolchain is used to compile the C code of SCuM firmwares using the c17 version of the C standard. A firmware build can also be configured with the different optimization levels provided by CMake, e.g. Release, Debug, MinSizeRel. This allows developers to choose between faster or smaller generated binary code. Additionally, the SCuM SDK provides support for the `newlib-nano` libc that is included in the GNU ARM embedded toolchain. Finally, the SCuM SDK introduces a modular approach to declare preprocessor macros based on the inclusion of modules required by an application. This approach reduces the generated firmware size by not compiling non-required code.

Table 1: Memory footprint of the `hello_world` binary for different build configurations.

Toolchain (build type)	ROM	RAM
Keil (`-O0`)	15,124 B	10,473 B
SDK (Debug)	16,068 B	9,180 B
SDK (MinRelSize)	11,068 B	9,180 B
SDK modular (MinRelSize)	8,100 B	1,696 B

Table 1 compares the ROM and RAM size of the *hello_world* application between the legacy Keil project and three SCuM SDK build strategies. When using the lowest optimization level, the firmwares generated by the SCuM SDK and Keil have similar sizes. By default, the *hello_world* application was built with radio code that wasn't used, so using the modular approach of the SCuM SDK divides the size of the firmware in half.

4 CONCLUSION & FUTURE WORK

We present the SCuM SDK, an open-source and multi-platorm set of tools and software that can be used to build and load a firmware on SCuM. The SCuM SDK provides a programming API for ARM microcontrollers using CMSIS as well as access to convenient libc primitives, using newlib-nano. It also provides sample applications to help newcomers learn how to use SCuM GPIO, UART, RFTimer and radio peripherals. We believe that the SCuM SDK opens the door for porting other existing software platforms or real-time operating systems (RTOS) to SCuM.

ACKNOWLEDGMENT

This project has received funding from the European Union's Horizon Europe Framework Programme, project "OpenSwarm", under Grant Agreement No. 101093046.

REFERENCES

[1] Tengfei Chang, Timothy Claeys, Mališa Vučinić, Xavi Vilajosana, Titan Yuan, Brad Wheeler, Filip Maksimovic, David Burnett, Brian Kilberg, Kris Pister, et al. 2020. Industrial IoT with Crystal-Free Mote-On-Chip. In *IEEE Symposium on VLSI Circuits (VLSI)*. IEEE, New York, NY, USA, 1–2.

[2] Filip Maksimovic, Austin Patel, David Burnett, Thomas Watteyne, and Kristofer SJ Pister. 2023. A Time Synchronized Multi-Hop Mesh Network with Crystal-Free Nodes. In *GLOBECOM 2023-2023 IEEE Global Communications Conference* (Kuala Lumpur, Malaysia). IEEE, New York, NY, USA, 4158–4163.

[3] Filip Maksimovic, Brad Wheeler, David C Burnett, Osama Khan, Sahar Mesri, Ioana Suciu, Lydia Lee, Alex Moreno, Arvind Sundararajan, Bob Zhou, et al. 2019. A Crystal-Free Single-Chip Micro Mote with Integrated 802.15.4 Compatible Transceiver, Sub-mW BLE Compatible Beacon Transmitter, and Cortex M0. In *Symposium on VLSI Circuits (VLSI)* (Kyoto, Japan). IEEE, New York, NY, USA, C88–C89.

[4] Sahar Mesri. 2016. Design and User Guide for the Single Chip Mote Digital System. Retrieved 2025 from https://people.eecs.berkeley.edu/~pister/publications/dissertations/Sahar_Mesri_MS_2016.pdf

[5] Titan Yuan. 2025. SCuM Code. Retrieved 2025 from https://github.com/tryuan99/scum-code