SAMC21 Design Notes

Anthony Kramer

**Why are we going to use the SAMC21 instead of:**

**Raspberry Pi, Beaglebone Black, other SOCs?**

MCU vs MPU

The SAMC21 is based off an ARM Cortex M0+, a real time microcontroller. It has a single 2 stage instruction pipeline running the Thumb 2 instruction set. The real time aspect of the SAMC21 is important to the autonomous car systems because a large deal of the functions of the car are time sensitive. CAN-Bus, SPI, and Asynchronous GPIO communications all need precise timings reliably.

The ARM Cortex A8 that most SOCs use right now has a symmetric superscalar pipeline that supports both Thumb 2 and NEON instruction sets. While the A8 can be clocked as high as 1Ghz vs the M0+s lackluster 48Mhz, it can execute multiple commands out of order. While this is great for an OS, it is terrible for time sensitive processes. Basically, it boils down to this, it is harder to program (and not to mention design) for a A8 when we are not using it for its intended use case.

References:

ARM Cortex M0+ Information

<https://developer.arm.com/products/processors/cortex-m/cortex-m0-plus>

ARM Cortex A8 Information

<https://developer.arm.com/products/processors/cortex-a/cortex-a8>

**Arduino?**

Intercommunication protocol

While the Arduino Uno’s, Mega’s, and Yun’s library and community support is second to none, it has its serious limitations. The Uno uses an ATmega328p that is clocked at 16Mhz and has 32KB of memory. The Mega uses an ATmega2560 that is clocked at 16Mhz and has 256KB of memory. The SAMC21 is clocked at 48Mhz and has 256KB of memory. However, these are not the important deciding factors, as we could have made this work. The big difference that mattered was the fact that the Arduino is not scalable.

Both the Uno and the Mega lack native support for CAN-Bus, relying on chips like the MCP2515 and MCP2551 to provide a link between SPI and CAN. This creates another problem, because that then uses the Arduino’s SPI port. The SAMC21 has two native CAN-Bus interfaces, for a total of 4 CAN-Bus lines. The SAMC21 can have 6 SPI ports if needed, along with every other bus you could think of.

References:

Arduino Uno R3 Information:

<https://store.arduino.cc/usa/arduino-uno-rev3>

Arduino Mega R3 Information:

<https://store.arduino.cc/usa/arduino-mega-2560-rev3>

SAMC21 Datasheet:

<http://www.mouser.com/ds/2/36/Atmel-42365-SAMC21_Datasheet-894213.pdf>

**Microchip PICs or Revolution Education’s PICAXE?**

Documentation:

Microchip PICs would be my second choice if for some reason we could not get the SAMC21. There is a broad selection of chips for every purpose you could think of. They are very cheap. But I can tell you from experience they are an absolute bitch to work with. I have a few back at home that I tried to use in a robotics project, unsuccessfully. They are mostly programmed in assembly by editing registers and they are programmed through JTAG using a HEX file. The biggest drawback is because they have so many chips, any community solution to a problem is probably incompatible for the chip you are trying to program.

The PICAXE series of MCUs have the opposite problem. I love these chips and will stand by them because they are so easy to use. However, they are the MCU equivalent of Python, great for any task, as long as you have a month of Sundays to run a task. PICAXEs are slow! Even the “fastest” PICAXE, the 40X2, can barely handle I2C, if you can even figure out how to program it.