APPLICATION NOTE

AT04188: SAM D20/D21/D10 - How to Achieve Power Numbers

ASF PROJECT DOCUMENTATION

Preface

In this application note we will first look at some of the factors that will influence power consumption and then set up some of the test configurations used in the "Power Consumption" section in the datasheet. The examples will be set up for the SAM D20 Xplained Pro board and the SAM D21 Xplained Pro board. As of this release of this application note code or tests has not been done for the SAM D10.

Table of Contents

1. Prerequisites

This application note requires:

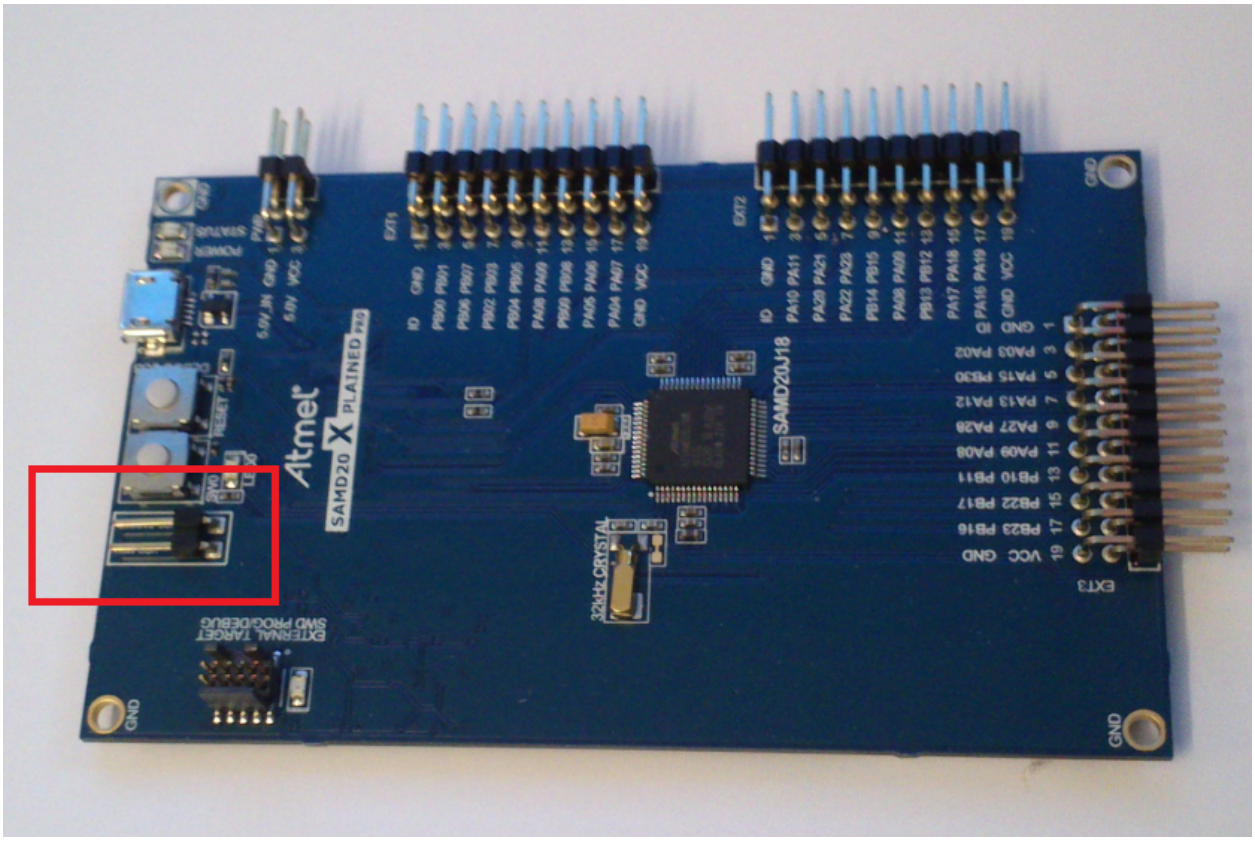
- Atmel® Studio 6.1 or newer
- The SAM D20 Xplained PRO or SAM D21 Xplained Pro
- A multimeter
- SAM D20/D21 datasheet

2. Using The Xplained Pro for Power Measurements

To get accurate power measurements it is necessary to have a good test setup where it is known that the current measured is powering the part that is of interest to evaluate. If this is not known and the current you are measuring is source for more than the part that is of interest, getting good measurements becomes more difficult.

The power measurements done in this application note are being performed on the SAM D20 Xplained Pro and the SAM D21 Xplained Pro. The Xplained Pro has got a pair of pins that are specifically designed to perform power measurements. These pins have to be shorted when a multimeter is not connected to the board. The pins are marked "VCC_MCU" and "VCC_TARGET" and can be found next to the two buttons on the board. And as seen in Figure 2-1: Pins for Power Measurements on the SAM D20 Xplained Pro on page 4 in the red rectangle.

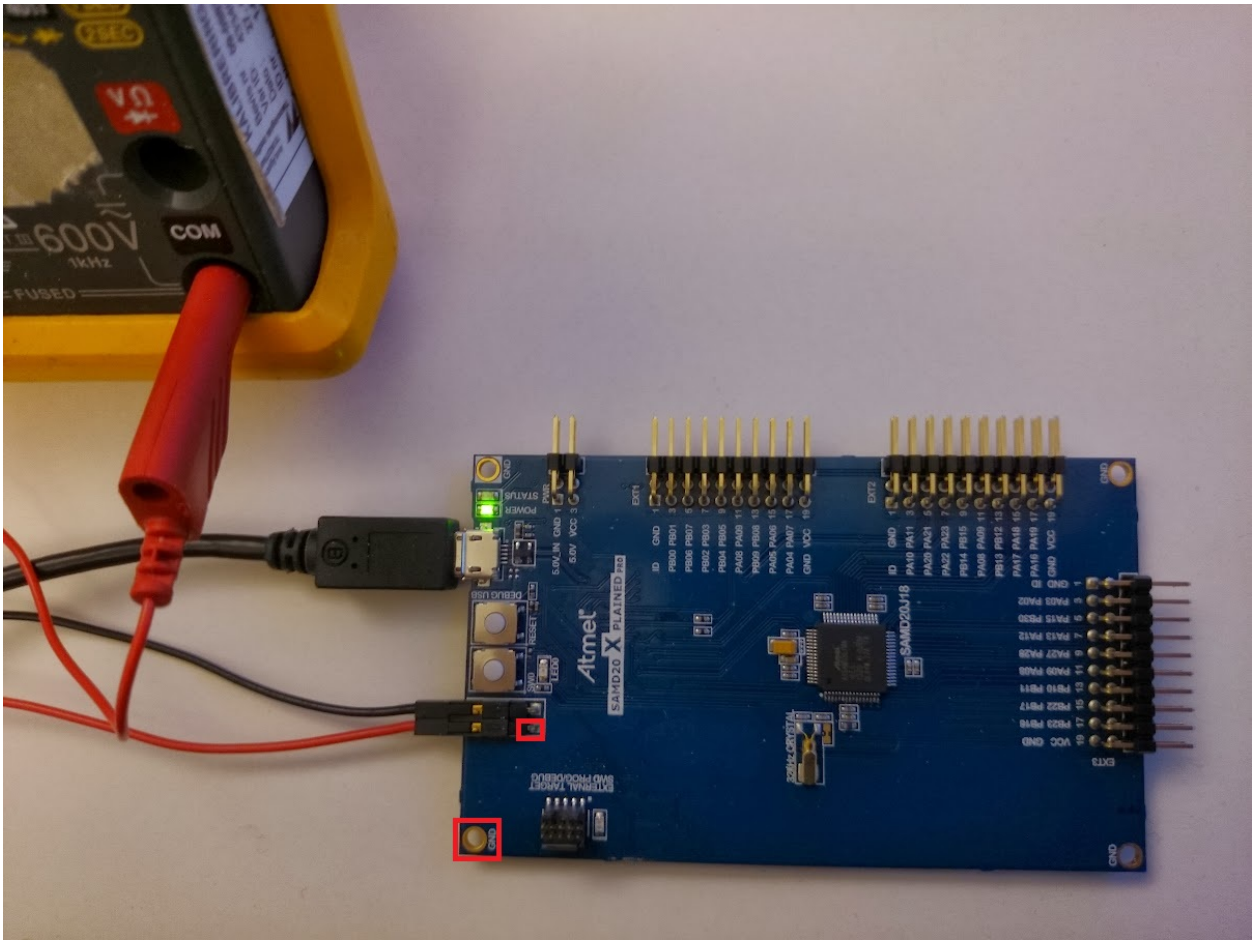
Figure 2-1. Pins for Power Measurements on the SAM D20 Xplained Pro



It is important to note that there are some limitations when it comes to using the Xplained Pro board for power measurements. The board is powered through the USB connection or a separate 5V connector. The voltage coming from the USB port or the 5V connector is then passed through a voltage regulator to generate a voltage of approximately 3.3V. This is used by the device on the board. This will limit all measurements on the Xplained PRO to be done at approximately 3.3V.

To determine the exact voltage the chip is supplied with the voltage of the board should be measured. This should be done when the multimeter, that is used for power measurements, is connected to the board. The reason for this is that the multimeter has got a small resistance in it when measuring power that will lower the voltage supplied to the device. The points to measure between can be found in figure Figure 2-2: Points for Voltage Measurements on the SAM D20 Xplained Pro on page 5.

Figure 2-2. Points for Voltage Measurements on the SAM D20 Xplained Pro



Note that if it is desired to power the board from the pins in the power header instead of the USB port, the pins to use are the ones marked "5.0V_IN" and "GND" on the header closest to the USB port. The pin marked "5.0V" is an output pin for 5V. Be careful not to connect a supply to this pin as this can destroy the board.

3. Reducing Power Consumption on the SAM D20, D21, and D10

There are several factors that will contribute to higher power consumption in any design. In all cases the application requirements will impact the minimal power consumption possible. However, with any given hardware architecture and application requirement there are design decisions and hardware features that can reduce the power consumption of an application.

Software design decisions such as using only polled mode or using interrupts, and sleep modes to reduce power. Reducing Leakage current from pins not being used, reducing the clock speed of the CPU and peripherals to the minimum necessary clock speed, or turning off clocks to peripherals not in use, can all contribute to getting lower power consumption from the device.

3.1 The Application Software Design

The software design choices will often be given by the application the device will be used for. A common software design is to run a state machine inside a while loop and then poll the peripherals continuously to check if they have any updates to their state, and then take some specific action if a condition is met. Often this approach will result in many cycles being spent waiting for updates from peripherals and thereby higher power consumption.

This can be improved by inserting sleep instructions to the CPU in the while loop. The CPU will still have to be woken up so in order to do that either peripherals that needs to be used can be responsible for waking the chip or a TC or the RTC can be configured to wake the device periodically.

The interrupt routine will in most cases be used to set a flag to be read and handled in the state machine running in the while loop or it may do some minor task that does not take many cycles to execute.

3.2 Executing Code From RAM

In some cases, executing code from RAM can save power. If all code is being executed from RAM the FLASH and the NVM controller can be turned off.

A different way which executing code from RAM can save power is when it is necessary to insert wait states when executing from FLASH. If the application switches between standby and active mode running code from RAM can in some cases allow you to spend less time in active mode.

If executing code from RAM is of interest, an application note on how this can be done can be found on www.atmel.com search for AT07347.

3.3 Power Consumption and the SAM D20 Synchronous Clock System

Any device will consume more power when its logical gates transition from one logic level to a different logic level. The more gates and the higher the clock rate the higher the power consumption will be. The SAM D20 has got a very flexible clock system where it is possible to turn off clocks and also run the peripherals and buses of the device with differently prescaled clocks when needed. This can help in reducing power consumption.

The clock system can be divided into the asynchronous domain and the synchronous domain. The synchronous clock domain consist of the CPU, FLASH and the synchronous buses that link the CPU to the peripherals of the device.

The SAM D20 has got three different buses which separates the clock control for turning peripheral clocks on or off. The Asynchronous Peripheral Buses (APB) A, B and C connects to the peripherals of the device and the High-speed peripheral bus (HPB). For an overview of which peripheral is connected to which bus see the block diagram in the "Block Diagram" chapter in the SAM D20 datasheet. The HPB connects the CPU to the FLASH and the SRAM and the three APB buses.

In order to get as low a power consumption as possible as many of the unused peripheral clocks that can be turned off should be turned off. This can be easily done using the ASF drivers developed for the SAM D20. The function to use in order to achieve this is the system_apb_clock_clear_mask() function. This function takes two arguments, an enum containing the bus id and a bit mask for the different peripheral clocks to disable. If it is necessary to turn clocks back on use the system_apb_clock_set_mask() function.

It is possible to turn of the clock to the power manager (PM) on the APBA bus, but the PM is used to turn off clocks when entering IDLE sleep modes 1 and 2. If IDLE sleep mode 1 and/or 2 are to be used the PM must be running when entering these sleep modes. The PM also controls the reset. If the PM is turned off it will not be possible to get it back on without a power on reset.

If the HPB bus clock to the NVM controller is turned off this will cause the device to stall as it will no longer be able to access FLASH. This should only be done if all code is going to be executed from RAM.

3.4 Power Consumption and the SAM D20 Generic Clock Generator

Most peripherals run on an asynchronous clock. This clock comes from the generic clock generator. In the generic clock generator it is possible to generate different clocks for different peripherals. On the SAM D20 and the SAM D21 there are eight different clock generators. On the SAM D10 there are six generators. Each generator can divide its clock source so even if only one clock source is used for all generators it is still possible to get eight different clock frequencies.

The clock system can also be configured so that the generators are automatically turned off if no peripheral is requesting its clock. The generator will then be automatically turned back on if a peripheral requests it's clock. This functionality is dependent on setting the ONDEMAND bit in the all the peripherals using the generator. Further the generators can be made to stay on in standby if a peripheral has the RUNSTANDBY bit on. The clock request mechanism will still be working even if the RUNSTANDBY bit is on.

3.4.1 Saving Power By Connecting Peripherals to Unused Generators

By default all peripherals will be connected to GCLK generator number zero (GCLKGEN[0]). As this generator is used to generate the main clock it will always be on in active mode as well as the idle modes. The clock from this generator will then be propagated further down the clock tree than necessary. This will cause higher power consumption.

In the power measurements that are presented in the datasheet all peripherals are connected to a GCLK generator that is not connected to a source oscillator. The only exception to this is the DFLL which has its reference clock connected to a 32kHz clock crystal. In the code distributed with this appnote code can be found in the function switch_gclkgen_to_peripherals().

Switching unused peripherals to a GCLK generator that is not setup with a source oscillator is something that will save power in all applications. ASF, above 3.19, will include code that switches peripherals to unused generators. For those who does not want to go through an update of ASF the code found in switch_gclkgen_to_peripherals() right after system_clock_init() which can be found in system_init() has executed.

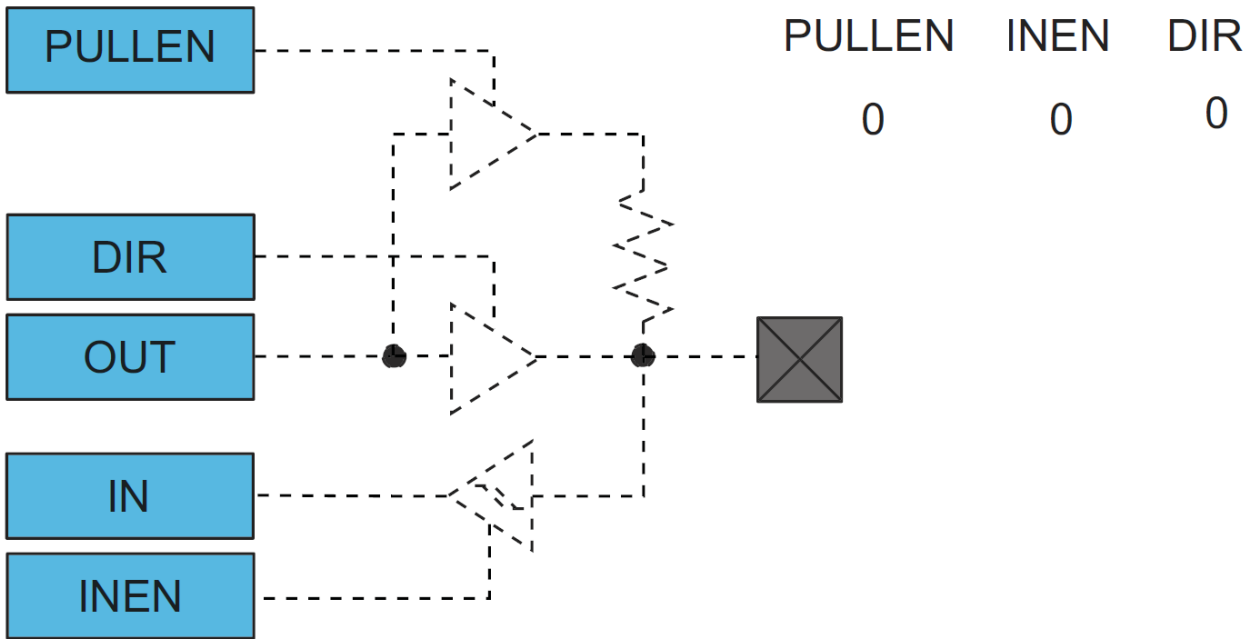
If ASF is not used either run the code as seen in switch_gclkgen_to_peripherals() before any peripherals are configured with a GCLK generator or switch the unused peripherals to a generator not running.

3.5 Power Consumption and I/O Pins

Leakage current from I/O pins will make the power consumption go up. For pins that are being used for IO, this leakage current must be reduced with correctly configuring the pins for the application. In addition the hardware outside the device must be taken into consideration to reduce current.

For unused pins the default state of the pins for the SAM D20 will give the lowest current leakage. Pins in the default state will be as shown in figure Figure 3-1: Pin Configuration for Low Power on page 8 with no connections going into or out of the device.

Figure 3-1. Pin Configuration for Low Power



If no pins are configured differently than the default configuration there is no need to do any configuration of the pins in order to lower the power consumption. It should be noted that when starting a new project with the SAM D20 Xplained Pro using ASF the pins connected to the on board button and LED0 will be configured as input and output respectively. If the button and LED0 is not being used the system_init() function being called in main() can be replaced with system_clock_init() as this will only configure the clocks as described in the conf_clocks.h file.

3.6 Power Consumption and Voltage

For many devices other than the SAM D20 there will be a large difference in power consumption when the device is running on lower voltage.The reason for this is that logical transitions will require more energy. And in many cases the entire device will be in one power domain.

For the SAM D20 the supplied voltage will have less impact on the power consumed by the device. The reason for this is that the SAM D20 uses two internal voltage regulators to power the core and the peripherals. One regulator is used in active mode and in the three IDLE modes, sourcing 1.1V, and one low power regulator is used when the device is in standby mode.

Because the low power regulator runs the core and peripherals at a lower voltage than the active mode regulator, power consumption of peripherals being used in standby mode will be lower than when being sourced from the 1.1V regulator.

There will still be slightly lower power consumption on the SAM D20 when running at lower voltages than 3.3V, as the regulators will not have such a large voltage over itself. And also parts of the device such as I/O pins and analogue parts is not wholly in the core power domain.

4. SAM D20 Datasheet Power Measurements

Included with this application note is code to do some of the power measurements that can be found in the datasheet.

One project is written for the SAM D20 Xplained Pro and one is written for the SAM D21 Xplained Pro as such some of the define names used will be related to names and pins found on the SAM D20/D21 Xplained Pro board. The compiled code should run without problems on any board. It should however be noted that boards such as the STK®600 is not suited for power measurements as there are no pins on it that can be used to only measure power going in to the device.

4.1 FLASH Wait States

If it is desirable to do power measurements on a different board with lower voltages then it is necessary to consult table 32-30 "Maximum Operating Frequency" in the in the "NVM Characteristics" section in the datasheet. This table shows how many FLASH wait states are necessary to use at different voltages and operating frequencies.

To change the wait state settings with the code supplied it is necessary to change the CONF_CLOCK_FLASH_WAIT_STATES define in the conf_clock.h file from 1 to the appropriate value from table 32-30. This ensures that when the system_clock_init() function is being run from main it will first set the number of wait states before changing the clock source used by the CPU from the 8MHz clock to the DFFL 48MHz clock. If it is not done in this order there is a risk that a read from flash will cause the CPU to crash when to few wait states are being used.

For the power measurements done at 3.3V at 48MHz it is necessary to use 1 FLASH wait state.

It is important to note that the number of FLASH wait states will impact the power measurements. If the number of wait states goes up the power consumption will go down because the CPU will have to wait additional cycles each time it does reads from FLASH. The measurements for the SAM D20 are done with no more than the necessary wait states.

4.2 Configuring the Power Manager

The Power Manager controls the reset of the device and it also allows for manual enabling and disabling of the synchronous clocks given to the peripherals in the SAM D20. In addition it is used to set the prescaler for the peripheral bus and the CPU clock.

For the power measurements presented in the electrical characteristics chapter many buses and peripherals are configured to be turned off. Many of these clocks will by default be turned off and only enabled if they are necessary for any given peripheral. In table 15.1 "Peripheral Clock Default State" the different peripheral clocks and their default state after reset are listed.

From table 15.1 we can see that the clocks that need to be disabled to replicate the conditions in the Power Consumption section are the CLK_DSU_APB, CLK_PAC1_APB, CLK_EIC_APB, CLK_PAC0_APB, CLK_PORT_APB, and the CLK_ADC_APB clocks. The rest of the clocks not needed are by default disabled.

In addition to disabling the synchronous clocks on the APBx buses some of the clocks on the high speed bus need to be disabled. The clocks to be disabled are the CLK_HPB1_AHB, CLK_HBP2_AHB and CLK_DSU_AHB clocks.

The power consumption numbers in the datasheet are measured with the APBA clock divided by four. This is also done by using the PM.

The final configuration of the PM is only necessary to do when testing power consumption in sleep modes. The Clocks Failure Detector needs to be disabled when going in to sleep. This feature monitors the clock of the CPU and compares it to the OSCULP32K. When going into sleep modes the CPU clock will be turned off. When this happens and the Clock Failure Detect is active this will cause the power consumption to be high.

In the example code the Clock Failure Detect is always off. It can be turned on by setting CONF_CLOCK_CPU_CLOCK_FAILURE_DETECT to true in conf_clock.h

4.3 BOD33

The BOD33, responsible for monitoring the Voltage to the 3.3V domain, is turned off during the power measurements. The function for turning off the BOD33 is turn_off_bod33().

Note that if running with the BOD33 on, this may lead to differences in the differential of the power consumption dependent on what algorithm is being run. This is not directly due to the BOD33 drawing more power dependent on the algorithm but due to the program being located in a different part of FLASH. if an algorithm is located so that the data that needs to be accessed is at two different rows this can cause higher power consumption. The reason being that the CPU will have to access both rows during execution which can cause higher power consumption during execution.

4.4 Pin Setup

Leakage current from incorrectly setup pins can contribute to unnaturally high power consumption. In the SAM D20 the default pin configuration is the setup that will give the lowest power consumption. In this configuration all GPIO pins are configured without any connections to the internal port functionality. This is the setup that can be seen in Figure 21-9. "I/O Configuration - Reset or Analog I/O: Digital Output, Input and Pull Disabled" in the datasheet.

The usual ASF setup for the SAM D20 Xplained Pro board includes setup to enable the SW0 button and LED0 for input and output. This is not wanted here and will cause a higher measured power consumption. The regular system_init() function has been removed and only the clock setup function is used (system_clock_init()).

The clock configuration used in the power measurements uses the 32kHz crystal on the SAM D20 Xplained Pro and it is then multiplied using the DFLL. Should it be necessary to change the clock setup or if for other reason a check of the main clock frequency is necessary this can be done by using the configure_gclk_pin() function. This function will configure pin PB14 to output GCLK0. In addition to adding this function it is necessary to enable the clock output in conf_clock.h by setting CONF_CLOCK_GCLK_0_OUTPUT_ENABLE to true.

4.5 Power Measurements

The SAM D20 datasheet gives multiple test cases for Power consumption. The test cases given here will only cover the tests done at 3.3V. If the tests are to be performed at a lower voltage it is not possible to use the SAM D20 Xplained Pro. The number of FLASH wait cycles will also have to be changed with lower voltages.

The different algorithms will require different parts of the CPU to be used. A good indication of what each algorithm requires from the CPU can be found by looking in the Disassembly view in Atmel Studio when debugging.

4.5.1 Fibonacci

The Fibonacci algorithm used here is a multiple recursive algorithm. This means that the function used to calculate the Fibonacci number calls itself more than once, in the case of the Fibonacci sequence twice. This will cause many load and store operations to and from stack. As well as multiple push, pop, adds, subs, and branches.

4.5.2 While(1)

The while one loop will for the SAM D20 only execute one assembly instruction namely a branch instruction that branches back to itself. This instruction will be served from the cache when this is configured to be on as it is in the power consumption characterization. The while(1) test will therefore not get any added consumption from reading from FLASH.

Note that the tests giving power consumption as a function of frequency multiplied with a constant and then plus one constant are done with an external clock source. If similar power consumption tests are done with the internal oscillators this will add a higher constant current draw to the measurements. See the electrical characteristics chapter for typical values for the OSC8M, DPLL, and the DFLL.

4.5.3 Prime Number Calculation

The prime number calculation will require an even larger number of instructions when it is executed, compared to the Fibonacci algorithm. The algorithm is not part of the datasheet but is presented here so that it can be used in comparison with other manufacturers who do give numbers for this algorithm. This algorithm will not be able to fit into the cache and it should cause some reads from FLASH.

4.5.4 CoreMark

The CoreMark algorithm is mainly for measuring the performance of any given CPU and not so much the power consumption. The CoreMark algorithm does however perform a larger variety of tasks and as such it is a good test for power consumption.

The license for the CoreMark algorithm does not permit republication so if it is desirable to perform these tests, the code can be found under license at the EEMBC website (http://www.eembc.org/coremark/). The CoreMark algorithm will not be further discussed in this application note.

4.6 Sleep Mode Power Measurements

Code to do sleep mode power measurements can be found in the code bundled with this appnote. The setup of buses and clocks is the same as described in the electrical characteristics chapter of the datasheet.

4.6.1 idle_modes

There are three Idle modes on the SAM D20. The difference between these Idle modes is what buss clocks are left still running. Idle 0 is the Idle mode with most clocks still running while Idle 2 is the lowest power Idle mode with the fewest clocks running.

4.6.2 Standby

Standby is the sleep mode with the lowest power consumption. In standby the core voltage is supplied from the low power regulator. The synchronous clock domain is off but parts of the asynchronous clock domain can be configured to be running. Which parts is configured by setting the run in standby bit in the peripherals that have this bit.

There are two power measurements done for standby one with the RTC configured to run in standby and one without.

For power measurements done in standby it is very important to have accurate measurement tools. The reason for this is that multimeter will often have calibration errors that become more apparent at low currents. When measuring the power with the Fluke multimeter the multimeter shows a current measurement between 2 and 3µA in standby.

4.6.2.1 Run in Standby

Several of the peripherals in the SAM D20 and SAM D21 have the option to run in standby. This is enabled by writing a one to the RUNSTDBY bit in the corresponding peripheral. When this is done the oscillator used by the peripheral will be kept running. If the oscillator is connected to GCLK Generator 0 (main clock oscillator) the clock from the oscillator will be propagated to many internal gates and therefor make the power consumption go up. For this reason it is recommended to connect peripherals to a different oscillator from the oscillator being used as the main clock.

5. Revision History

Doc. Rev.	Date	Comments
42248A	06/2014	Initial document release.
42248B	10/2014	Update to include D21/D10 and peripherals switching generator technique.

Atmel | Enabling Unlimited Possibilities®

Atmel Corporation 1600 Technology Drive, San Jose, CA 95110 USA **T:** (+1)(408) 441.0311 **F:** (+1)(408) 436.4200 | **www.atmel.com**