
AT12705: SAM L21 ADC Sampling using Low-Power Features

APPLICATION NOTE

Description

The ultra-low-power Atmel® Cortex® M0+ SAM L21 device achieves very low power consumption while performing tasks in sleep modes. In this application note it is described how to configure a SAM L21 Xplained Pro to sample a sequence of inputs using the ADC while achieving a very low power consumption in Standby sleep mode. The ADC sampling is triggered by RTC periodic events and the results are stored in RAM using DMA. Sampling a sequence of three inputs 32 times per second results in an average power consumption of 1.7µA.

Table of Contents

Description.....	1
1. Introduction.....	3
1.1. Prerequisites.....	3
2. Example Overview.....	4
2.1. Low-power Features.....	5
2.2. Peripherals.....	6
2.3. Main Function.....	8
2.4. Data Visualizer.....	9
2.4.1. Getting Started.....	9
2.4.2. Power Measurements.....	10
3. Software License.....	11

1. Introduction

This application note and the related example project shows the implementation of a low-power sampling application using Atmel SAM L21. The device samples a sequence of inputs 32 times per second with an average power consumption of 1.7µA. To ensure a minimal power consumption SAM L-series low-power features are applied, such as main voltage regulator selection, sleepwalking with power domain gating, and utilizing the ultra-low-power internal oscillator. The various techniques for reducing the power consumption will be described in the following section.

The related example project is available as a .zip-file from the SAM L product page¹ and is implemented using Atmel Software Framework (ASF). It will be necessary to open the project to see the full implementation as only key parts of the code will be presented in this application note.

1.1. Prerequisites

Running the example described in this application note requires:

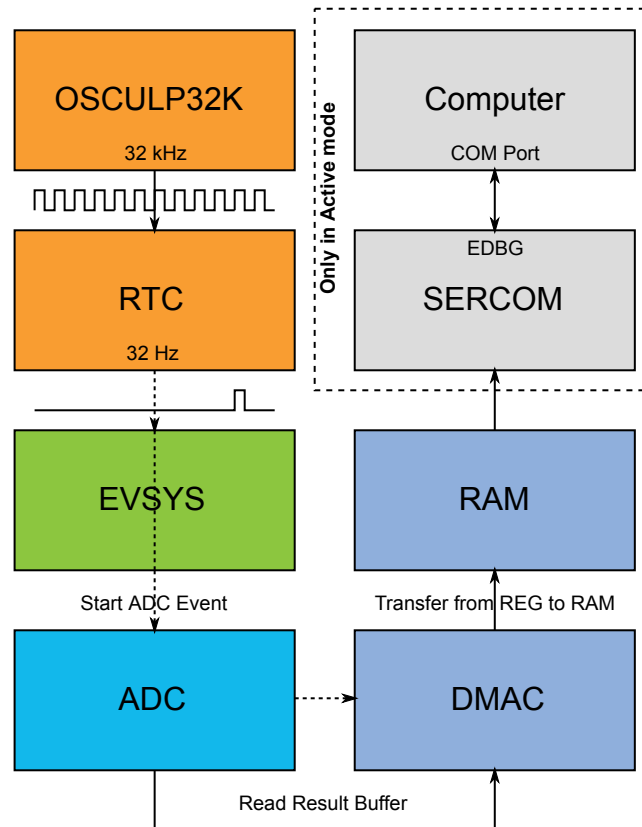
- Atmel Studio 6.2 Service Pack 2
- Atmel Software Framework 3.21.0 or higher
- SAM L21 Xplained PRO Evaluation Kit with USB cable
- Atmel Data Visualizer Extension (recommended)

¹ <http://www.atmel.com/products/microcontrollers/arm/sam-l.aspx?tab=documents>

2. Example Overview

The example project aims to show how to achieve a very low average power consumption when sampling a sequence of three inputs 32 times per second. The RTC is used to generate events, initiating each conversion sequence. The three individual samples are transferred from the result register to RAM using DMA. The application can operate in either active mode or sleep mode, controlled with a button on the board. In active mode, USART communication is enabled to transmit the latest ADC samples to a COM port on the connected computer. In sleep mode the RTC, ADC, and DMA will continue to operate as sleepwalking tasks.

Figure 2-1 Application Schematic



The configurations and most important implementations from the example project are described and shown in this section. Firstly, the actions needed to achieve the low power consumption are presented. Secondly, peripheral configurations are described. Refer to the related example project for the complete implementation.

More details regarding the SAM L-series low-power features and techniques can be found in the following application notes:

- Atmel AT06549: Ultra Low Power Techniques²
- Atmel AT04296: Understanding Performance Levels and Power Domains³

² http://www.atmel.com/Images/Atmel-42411-Ultra-Low-Power-Techniques-AT06549_Application-Note.pdf

³ http://www.atmel.com/Images/Atmel-42412-Understanding-Performance-Levels-and-Power-Domains-AT04296_Application-Note.pdf

2.1. Low-power Features

Main Voltage Regulator Selection

The main voltage regulator can operate from two different internal regulators; LDO or buck regulator. The buck regulator is the most power efficient and is used in this example. The voltage regulator can be switched on the fly by writing to the Voltage Regulator Selection bit in the Voltage Regulator System Control register (VREG.SEL), shown in the below code snippet.

```
SUPC->VREG.bit.SEL = 1;
```

Enable Voltage Reference in Standby

The voltage reference used for the ADC conversion should be available in sleep modes but only when requested. It is therefore necessary to set the Run in Standby bit and the On Demand Control bit in the Voltage References System Control register (VREF.RUNSTDBY/ONDEMAND), shown in the below code snippet.

```
SUPC->VREF.bit.ONDEMAND = 1;  
SUPC->VREF.bit.RUNSTDBY = 1;
```

Enable Voltage Regulator in Standby

The ADC will require a voltage regulator capable of delivering an output sufficient for conversions. The main voltage regulator is therefore preferred also in Standby sleep mode where a low-power voltage regulator (LPVREG) is selected by default. To enable the main voltage regulator in Standby sleep mode the Run in Standby bit in the Voltage Regulator System Control register (VREG.RUNSTDBY) must be set.

```
SUPC->VREG.bit.RUNSTDBY = 1;
```

Enable Dynamic Power Domain Gating

Power domain gating allows the device to power only selected parts of the device. In Standby sleep mode, domains containing solely disabled peripherals will by default be in retention state to reduce current leakage and thereby lower the power consumption. Domains containing enabled peripherals can dynamically switch between active and retention state depending on the state of the peripherals. This feature is controlled by the Dynamic Power Domain Gating for Power Domain x bits in the Standby Configuration register (STDBYCFG.DPGPDx), which must be set to allow dynamic switching. In the example project, dynamic power domain gating is enabled for PD0 containing e.g. ADC and OSCCTRL, and also enabled for PD1 containing e.g. DMAC. Enabling dynamic switching of PD0 and PD1 is shown in the below code snippet. In the example project, enabling dynamic power domain gating reduces the average power consumption by ~35µA.

```
PM->STDBYCFG.bit.DPGPD0 = 1;  
PM->STDBYCFG.bit.DPGPD1 = 1;
```

Clock Configurations and Performance Level Selection

The performance level technique consists of adjusting the voltage of the regulator output to reduce power consumption. Performance levels are automatically selected depending on main clock frequency. By selecting GCLK generator frequencies below 24MHz and both CPU and peripheral clock frequencies

below 12MHz, the device will operate at the lowest and most power efficient performance level PL0. In the example project, the main frequency is 4MHz. Clock settings are configured in `conf_clocks.h`:

```
/* SYSTEM_CLOCK_SOURCE_OSC16M configuration - Internal 16MHz oscillator */
# define CONF_CLOCK_OSC16M_FREQ_SEL      SYSTEM_OSC16M_4M

/* Configure GCLK generator 0 (Main Clock) */
# define CONF_CLOCK_GCLK_0_ENABLE        true
# define CONF_CLOCK_GCLK_0_RUN_IN_STANDBY false
# define CONF_CLOCK_GCLK_0_CLOCK_SOURCE   SYSTEM_CLOCK_SOURCE_OSC16M
```

Ultra-Low-Power Oscillator Selection

The ultra-low-power internal 32.768kHz oscillator OSCULP32K provides the device with the lowest oscillator power consumption. OSCULP32K is enabled by default and should be preferred whenever the power requirements are prevalent over frequency stability and accuracy. In the example project, OSCULP32K is applied as RTC clock source, as defined in `conf_rtc.h`:

```
# define RTC_CLOCK_SOURCE      RTC_CLOCK_SELECTION_ULP32K
```

Enable Sleepwalking

Sleepwalking allows a peripheral to wake up its corresponding GCLK and run while the device is in a sleep mode, without waking up the CPU. To perform a sleepwalking task a peripheral should be configured to run on demand in standby. This will allow the peripheral to run only when requested. For the ADC these settings are configured by setting the On Demand Control bit and the Run in Standby bit in the Control A register (CTRLA.ONDEMAND/RUNSTDBY). The corresponding clock source should also run on demand, to ensure it is only awake when requested by a peripheral. The OSC16M will by default run only on demand when in Standby sleep mode.

2.2. Peripherals

Real-Time Counter (RTC)

The RTC is configured to generate periodic events used to trigger ADC conversions. Periodic events are generated at periodic intervals with frequency given by:

$$f_{PER(n)} = \frac{f_{CLK_RTC_OSC}}{2^{n+3}}$$

where $f_{CLK_RTC_OSC}$ is the frequency of the internal prescaler clock CLK_RTC_OSC and $n = 0...7$ is the position of the Periodic Interval n Event Output Enable bitgroup in the Event Control register (EVCTRL.PERON). In the example we choose a periodic event frequency of 32Hz to start 32 sequential conversions per second. To achieve the desired frequency, the 32.768kHz RTC clock source OSCULP32K must be prescaled by 1024, giving $n = 7$ from the equation $2^{n+3} = 1024$. The 32Hz periodic events are routed via an event channel to the ADC. The following configurations are made to enable generation of the seventh periodic event:

```
conf_rtc_events.generate_event_on_periodic[0] = false;
...
conf_rtc_events.generate_event_on_periodic[6] = false;
conf_rtc_events.generate_event_on_periodic[7] = true;
```

Event System (EVENT)

The event system must be configured with a channel to route events from the RTC periodic event generator to the ADC start conversion user. The following configurations allocates a channel which runs asynchronously in standby with RTC periodic event 7 set as generator and ADC start conversion set as user:

```
config_evsys.generator      = EVSYS_ID_GEN_RTC_PER_7;
config_evsys.edge_detect   = EVENTS_EDGE_DETECT_NONE;
config_evsys.path          = EVENTS_PATH_ASYNCHRONOUS;
config_evsys.clock_source  = GCLK_GENERATOR_0;
config_evsys.on_demand     = true;
config_evsys.run_in_standby = true;

events_allocate(&event_channel, &config_evsys);
events_attach_user(&event_channel, EVSYS_ID_USER_ADC_START);
```

Analog-to-Digital Converter (ADC)

The ADC is configured to start converting a sequence of inputs when incoming event occurs. To minimize the power consumption, sleepwalking is enabled to ensure the ADC only runs in sleep mode when requested. The following configurations set the incoming event action to start new conversion and enables the ADC to run as a sleepwalking task:

```
config_adc.event_action     = ADC_EVENT_ACTION_START_CONV;
config_adc.run_in_standby   = true;
config_adc.on_demand        = true;
```

The sequential sampling is configured with the Sequence Control (SEQCTRL) register and the Enable Positive Input n in the Sequence (SEQENn) bitgroup. The bits are set according to the Positive Mux Input Selection. The three inputs selected in the example application are SCALED COREVCC, SCALED IOVCC, and SCALED VBAT.

```
ADC->SEQCTRL.reg = 0x2C000000;
```

Direct Memory Access Controller (DMAC)

The DMA is configured to transfer the ADC results from the result buffer to a user defined array in RAM. Since the ADC is configured to sample a sequence of inputs, the individual results must be transferred after each conversion. The following configurations defines the DMA descriptor source and destination addresses for transferring data from ADC result register to RAM.

```
config_descriptor.source_address =
    (uint32_t)(&adc_instance.hw->RESULT.reg);
config_descriptor.destination_address =
    (uint32_t)&adc_result_store[NO_OF_INPUT_SCAN];
```

The transfer is initiated when the ADC signals that a conversion has ended, using a dedicated event channel. This is configured using the following code line in the DMA configuration function:

```
config_dma.peripheral_trigger = ADC_DMAC_ID_RESRDY;
```

External Interrupt Controller (EIC)

The SW0 button on the Xplained Pro board is in the application used to switch between active mode and sleep mode. To identify a button push, EIC is configured to detect a falling slope on the SW0 button

interrupt line. A callback function is registered and called when a button push is detected. The interrupt wakes the device if sleeping and the callback function switches a power mode variable, which keeps track of the current mode.

Serial Communication Interface (SERCOM USART)

The USART is configured for serial communication between the application and a computer. The communication is used to print out the latest ADC samples when the device is in active mode. Communication to and from the USART is relayed via the EDBG, which enumerates as a USB communication device to the PC using the same port and cable as for programming.

Revision A Workarounds

Some of the SAM L21 revision A erratas are fixed in the example project using workarounds. These workarounds are applied if the `SAM_L21_REV_A` define is present. Fix for errata reference 13901 is added to the `extint_detection_callback()` function. A workaround is also implemented for errata reference 14268 regarding the ADC voltage reference. For this purpose the DAC is enabled to ensure the analog reference is available when requested by the ADC.

2.3. Main Function

The application is implemented in the main function. Initially, the power saving features are applied before all the drivers are configured and enabled. In the while-loop the application switches between normal mode and sleep mode when a button push is detected. When in normal mode, the device is ready to receive and send data over USART to print out the latest ADC results. By sending an 'S' over USART, the latest sequence of results are transmitted and printed in the open serial COM port terminal window. In sleep mode the conversion and transferring of data runs as sleepwalking tasks.

```
/** Main function */
int main(void)
{
    system_init();

    /** Enable power saving features */
    select_buck_regulator_as_main_vreg();
    enable_power_domain_gating_pd01();
    enable_vref_and_vreg_runstdby();

    /** Configure RTC, ADC, EVSYS, EXTINT and USART */
    configure_rtc();
    configure_adc();
    configure_evsys();
    configure_extint_channel();
    configure_usart();

    /** Configure DAC workaround if Rev. A */
    #ifdef SAM_L21_REV_A
    configure_dac();
    #endif

    /** Configure DMA channel and descriptor */
    configure_dma_channel(&dma_channel);
    configure_dma_descriptor(&descriptor_0);

    /** Add descriptor and start transfer job */
    dma_add_descriptor(&dma_channel, &descriptor_0);
    dma_start_transfer_job(&dma_channel);
}
```

```

/** Enable interrupts */
system_interrupt_enable_global();

/** Set STANDBY as sleep mode */
system_set_sleepmode(SYSTEM_SLEEPMODE_STANDBY);

/** Set default power mode to normal */
powermode = POWER_MODE_NORMAL;

while (1) {
    /** Device awake */
    if (powermode == POWER_MODE_NORMAL) {
        usart_read_buffer_job(&usart_instance,
            (uint8_t *)rx_buffer,
            MAX_RX_BUFFER_LENGTH);
        if (send_data) {
            send_data = false;
            char buffer[100];
            sprintf(buffer, "ADC1: %u ADC2: %u ADC3: %u\r\n",
                adc_result_store[0],
                adc_result_store[1],
                adc_result_store[2]);
            usart_write_buffer_job(&usart_instance,
                (uint8_t *)buffer,
                strlen(buffer));
        }
    }

    /** Device asleep*/
    if (powermode == POWER_MODE_SLEEP) {
        system_sleep();
    }
}
}

```

Note: When SERCOM is configured and enabled via EDBG, the serial COM port on the PC must be enabled to avoid current leakage increasing the power consumption. In the example project, opening the COM port reduces the average current consumption by ~90µA.

2.4. Data Visualizer

Atmel Data Visualizer can be downloaded as an extension to Atmel Studio. The Data Visualizer can be used for power analysis, oscilloscope view with trigger functionality, and much more. The Data Visualizer is a helpful tool for this example project to analyse and measure the power consumption.

2.4.1. Getting Started

1. Start Atmel Data Visualizer.
2. The current measurement headers on the SAM L21 Xplained Pro board should be connected to only measure the MCU consumption, as no I/O consumption is of interest in this application. This implies setting the MCU-header to MEASUREMENT-position and I/O-header to BYPASS-position.
3. Connect the SAM L21 Xplained Pro EDBG with a Micro-USB cable to your PC.
4. Select your board in the drop-down menu and hit the Connect-button.
5. Select the Power-interface and then hit the Start-button, both indicated in the below figure. Your board will be reset and a graph showing real-time power measurements will be displayed.

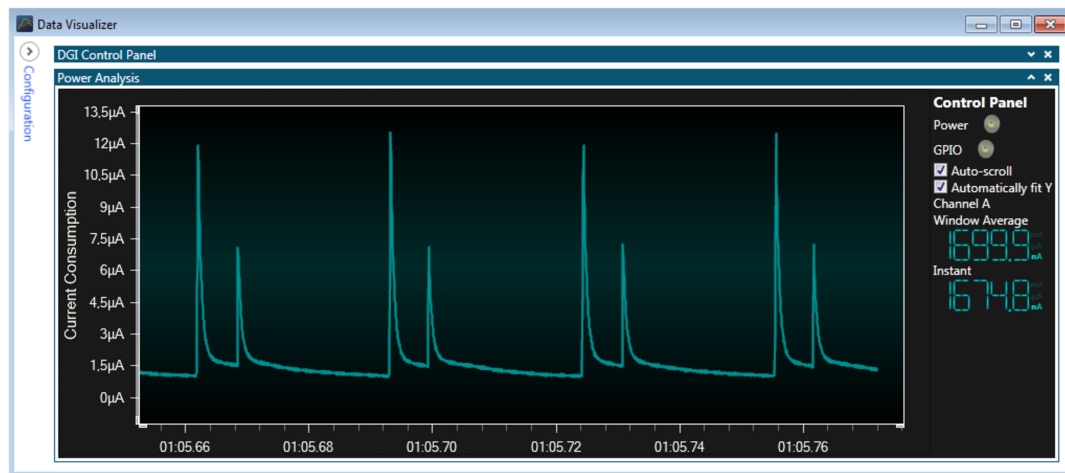


6. When the Auto-scroll functionality is enabled, the Power Analysis window is automatically moving to display the latest samples. In this mode the Window Average will indicate an exponential moving average of the MCU current consumption. This implies that e.g. after entering sleep, the Window Average calculations will converge towards the actual sleep mode average current consumption.

2.4.2. Power Measurements

The below figure is captured while the device is in Standby sleep mode running sleepwalking tasks. The larger spikes indicates the power profile for sampling using ADC and transferring the results using DMA. The time interval between two large spikes is measured to ~0.03 seconds, corresponding to a frequency of ~32Hz, which is the desired start ADC conversion rate. The smaller spikes are due to the buck regulator and a capacitor, which is periodically charging/discharging. In between the sleepwalking tasks the device is running the RTC in Standby sleep mode. The Window Average indicates an average current consumption of 1.7µA.

Figure 2-2 Data Visualizer Screen Shot



3. Software License

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The name of Atmel® may not be used to endorse or promote products derived from this software without specific prior written permission.
4. This software may only be redistributed and used in connection with an Atmel® microcontroller product.

THIS SOFTWARE IS PROVIDED BY ATMEL "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT ARE EXPRESSLY AND SPECIFICALLY DISCLAIMED. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.



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

© 2015 Atmel Corporation. / Rev.: Atmel-42471A-SAM-L21-ADC-Sampling-using-Low-power-Features_AT12705_Application Note-06/2015

Atmel®, Atmel logo and combinations thereof, Enabling Unlimited Possibilities®, and others are registered trademarks or trademarks of Atmel Corporation in U.S. and other countries. ARM®, ARM Connected® logo, and others are the registered trademarks or trademarks of ARM Ltd. Other terms and product names may be trademarks of others.

DISCLAIMER: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. EXCEPT AS SET FORTH IN THE ATMEL TERMS AND CONDITIONS OF SALES LOCATED ON THE ATMEL WEBSITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS AND PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and products descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

SAFETY-CRITICAL, MILITARY, AND AUTOMOTIVE APPLICATIONS DISCLAIMER: Atmel products are not designed for and will not be used in connection with any applications where the failure of such products would reasonably be expected to result in significant personal injury or death ("Safety-Critical Applications") without an Atmel officer's specific written consent. Safety-Critical Applications include, without limitation, life support devices and systems, equipment or systems for the operation of nuclear facilities and weapons systems. Atmel products are not designed nor intended for use in military or aerospace applications or environments unless specifically designated by Atmel as military-grade. Atmel products are not designed nor intended for use in automotive applications unless specifically designated by Atmel as automotive-grade.