
Using SAM L21 Ultra Low Power Modes on the SAM L21 Xplained Pro B

AN-16000

Prerequisites

- **Hardware Prerequisites**
 - Atmel® | SMART™ SAM L21 Xplained Pro B revision 5 (or newer)
 - Embeds an ATSAML21J18B revision C (or newer)
 - One Micro USB cable (type A/Micro B)
- **Software Prerequisites**
 - Atmel® Studio 7 (Version: 7.0.943 or higher)
 - Atmel Start (version 1.0.64.0 or higher)
 - Data Visualizer Extension (version 2.6.475 or higher)
 - Internet connection
- **Audience:** Beginner
- **Estimated Completion Time:** 60min

Introduction

The goal of this hands on is to describe and illustrate the different low power modes available in the Atmel® | SMART™ SAM L21 Series.

It will also demonstrate some of the SAM L21 low power techniques which can be used to optimize the power consumption of any application.



Table of Contents

Prerequisites	1
Introduction	1
Icon Key Identifiers	3
1. Introduction	4
1.1 Atmel® SMART™ SAM L21 Xplained Pro B	4
1.2 Atmel® SMART™ SAM L21 Low-Power Modes	5
2. Assignment 1: Create and Configure a New Project using Atmel START	6
2.1 Project Creation	6
2.2 Configure the Oscillators and the Clocks	9
2.3 Add and Configure the External Interrupt Controller Peripheral	13
2.4 Configure SW0 Push Button using Atmel Start PINMUX CONFIGURATOR	17
2.5 Save and Export the Application on an Atmel Studio 7 project	19
3. Assignment 2: Implement SAM L21 Low Power Modes	20
3.1 ACTIVE Mode Implementation	23
3.2 IDLE Sleep Mode Implementation	30
3.3 STANDBY Sleep Mode Implementation	35
3.4 BACKUP Sleep Mode Implementation	38
3.5 OFF Sleep Mode Implementation	41
4. Conclusion	43
5. Appendix: Solution Project (direct entry in IDLE sleep mode) ..	44
6. Appendix: Project waiting in ACTIVE mode (SAML21-PowerModes project)	45
7. Appendix: Atmel Studio 7 Help Viewer	47
8. Appendix: Upgrade Embedded Debugger (EDBG) Firmware ..	52
9. Revision History	54

Icon Key Identifiers



INFO

Delivers contextual information about a specific topic



TIPS

Highlights useful tips and techniques



TO DO

Highlights objectives to be completed



RESULT

Highlights the expected result of an assignment step



WARNING

Indicates important information



EXECUTE

Highlights actions to be executed out of the target when necessary

1. Introduction

The goal of this hands on is to describe and illustrate the different low power modes available in the Atmel® | SMART™ SAM L21 Series.

It will cover the following points:

- Presentation of the different SAM L21 Low Power Modes.
- How to analyze the SAM L21 Xplained Pro B power consumption using Atmel Studio 7 Data Visualizer tool.
- Crosscheck of datasheet consumption values with the ones measured on the Atmel | SMART SAML21 Xplained Pro B for the different low power modes.

It will also demonstrate some of the following SAM L21 low power techniques which can be used to optimize the power consumption of any application:

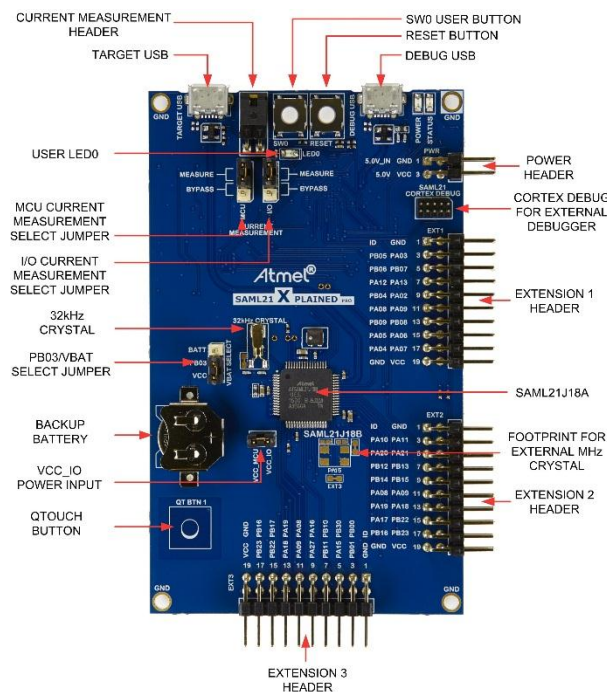
- Flexible clock architecture
- Triple regulators architecture with embedded BUCK converter
- Dynamic voltage scaling using SAM L21 Performance Levels
- Clock and Power domain gating

1.1 Atmel® | SMART™ SAM L21 Xplained Pro B

The Atmel® SAM L21 Xplained Pro evaluation kit is a hardware platform to evaluate the ATSAML21J18B microcontroller.

Supported by the Atmel Studio integrated development platform, the kit provides an easy access to the features of the Atmel ATSAML21J18B and explains how to integrate the device in a custom design.

The Xplained Pro MCU series evaluation kits include an on-board Embedded Debugger so that no external tools are necessary to program or debug the ATSAML21J18B.

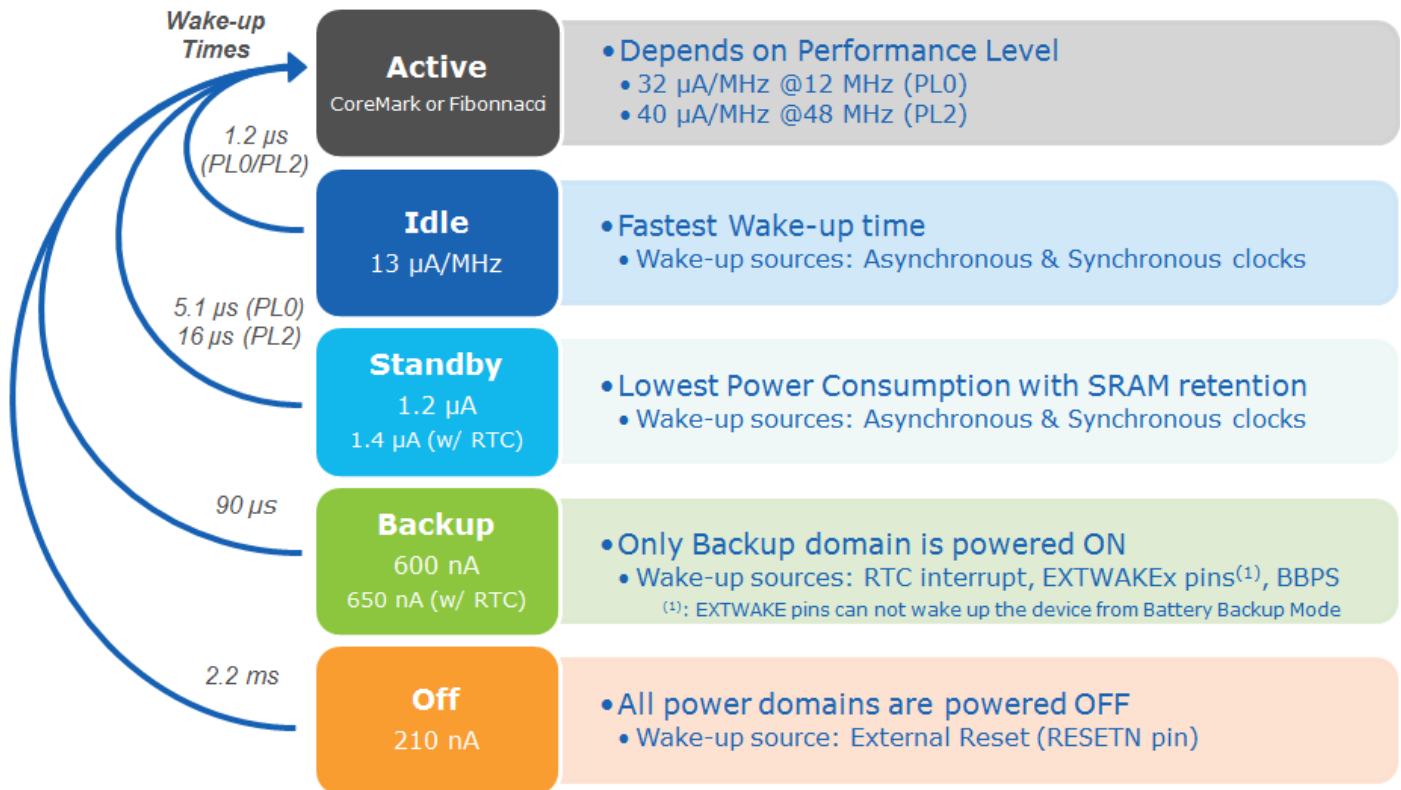


1.2 Atmel® | SMART™ SAM L21 Low-Power Modes

SAM L21 offers five different power modes allowing tradeoff between performance and power consumption of any application.

Each of these modes is listed in the picture below:

SAM L21 Key Numbers (VDDIN=3,3V, T=25°C, BUCK mode)



ACTIVE: in ACTIVE mode, the CPU is executing the application code.

- All clock domains & power domains are active, allowing software execution & peripheral operation.

IDLE: the IDLE mode allows power optimization with the fastest wake-up time.

- The CPU is stopped, and peripherals are still working.

STANDBY: the STANDBY mode is the lowest power configuration while keeping the state of the logic and the content of the RAM.

- In this mode, all clocks are stopped except those configured to be running sleepwalking tasks.

BACKUP: the BACKUP mode allows achieving the lowest power consumption aside from OFF mode.

- The device is entirely powered off except for the backup domain.

OFF: in OFF mode, the device is entirely powered-off.

2. Assignment 1: Create and Configure a New Project using Atmel START

We will use Atmel START tool to create our application project, configure the clocks of our system and add the required driver(s).



INFO

Atmel START (<http://start.atmel.com>) is a tool which will help the developer to select and configure software components, drivers, middleware and example projects to tailor your embedded application in a usable and optimized manner.

The workflow is quite straight forward: filter MCUs by requirements before starting a project.

Next you add components to your project, configure each component, export the project and add it into your favorite IDE for further development.

Our application will require in Atmel START to:

- Configure the clocks of the system that fit with the SAM L21 power consumption measurement conditions described in the SAM L21 Datasheet (Electrical Characteristics section).
- Add the SAM L21 External Interrupt Controller driver (EIC), which will be used to generate the interrupts and events required to exit the different low power modes.

2.1 Project Creation



TO DO

Create New Project

- Open a browser and go to <http://start.atmel.com>
- Select CREATE NEW PROJECT:



- Click on “Show only boards” from RESULTS section then select the SAM L21 Xplained Pro.
- Click on CREATE NEW PROJECT to complete the project creation:

RESULTS

☐ Show all
 ☒ Show only boards
 ☐ Show only devices

Name	Architecture	Package	Pins	Flash	SRAM
■ SAM B11 Xplained Pro					
■ SAM C21 Xplained Pro					
■ SAM D10 Xplained Mini					
■ SAM D11 Xplained Pro					
■ SAM D20 Xplained Pro					
■ SAM W25 Xplained Pro					
■ SAMD21J18A Low Voltage Motor Control Starter Kit					
■ SAM D21 Xplained Pro					
■ SAM L21 Xplained Pro					
■ SAM L22 Xplained Pro					
■ SAM R21 Xplained Pro					

11 of 292 boards and devices

CREATE NEW PROJECT >



INFO

It is possible to add MIDDLEWARE as DRIVERS before creating the project by selecting them in the FILTERS section.

For this hands-on, it is not required.

FILTERS

HARDWARE

MIDDLEWARE

DRIVERS

☒
☒
☒



RESULT

The project is created in Atmel START and you have now access to the DASHBOARD view:

Atmel START [Return To Front Page](#) | [Help And Support](#)

[VIEW CODE](#) [SAVE CONFIGURATION](#) [EXPORT PROJECT](#)

MY SOFTWARE COMPONENTS

[Add software component](#)

Clicking "Add software components" will allow you to add peripherals (modules), middleware and example project to your MCU-projects


APPLICATION ⓘ

[My Project](#) ⚙️

MIDDLEWARE +

DRIVERS + [Show system drivers](#) ⓘ

SELECTED BOARD: SAM L21 XPLAINED PRO

 The Atmel® | SMART™ SAM L21 Xplained Pro evaluation kit is a hardware platform to evaluate the ultra low power Atmel ATSAML21J18B microcontroller. Supported by the Atmel Studio integrated development platform, the kit provides easy access to the features of the Atmel® | SMART™ SAM L21 and explains how to integrate the device in a customer design.



INFO

You can check that choosing the SAM L21 Xplained PRO board automatically selects the ATSAML21J18B (TQFP64) as device which is the one mounted on it.

SELECTED DEVICE: ATSAML21J18B

GENERAL

Name	ATSAML21J18B
CPU	CORTEX-M0PLUS
Flash	264 KB
SRAM	40 KB
Package	TQFP64

SUPPORTED PERIPHERALS

AC	1	OSC32KCTRL	1
ADC	1	OSCCTRL	1
AES	1	PAC	1
CCL	1	RTC	1
DAC	1	SERCOM	6
DSU	1	SysTick	1
EIC	1	TC	5
GCLK	1	TCC	3
MCLK	1	TRNG	1
NVMCTRL	1	USB	1

2.2 Configure the Oscillators and the Clocks



TO DO

Review Product Datasheet Current Consumption Measurement Conditions

Oscillators and Clocks configuration used to measure the SAM L21 current consumption, are available in the *Electrical Characteristics > Power Consumption* section of the product datasheet.



INFO

SAM L21 Product Datasheet can be found at the following link:
<http://www.atmel.com/devices/ATSAML21J18B.aspx>

Here they are:

- Oscillators
 - XOSC (crystal oscillator) stopped
 - XOSC32K (32KHz crystal oscillator) running with external 32KHz crystal
 - When in active Performance Level 2 (PL2) mode, DFLL48M runs at 48MHz and uses XOSC32K as reference
 - When in active PL0 mode, the internal Multi RC Oscillator is running at specified frequency
- Clocks
 - DFLL48M used as main clock source when in PL2 mode. In PL0 mode, OSC16M used at 4, 8, or 12MHz
 - Clock masks and dividers at reset values: All AHB & APB clocks enabled, CPUDIV=1, BUPDIV=1, LPDIV=1



RESULT

So to sum up, in order to measure the current consumptions in the different low power modes, we will setup our clocks and oscillators in the following conditions:

- OSC16M is used as main clock source running at 12MHz (PL0 mode)
- XOSC, XOSC32K, DFLL48M are stopped
- All AHB & APB clocks enabled (reset values)
- CPUDIV=1, BUPDIV=1, LPDIV=1 (CPU / Backup / Low Power Clocks Dividers)

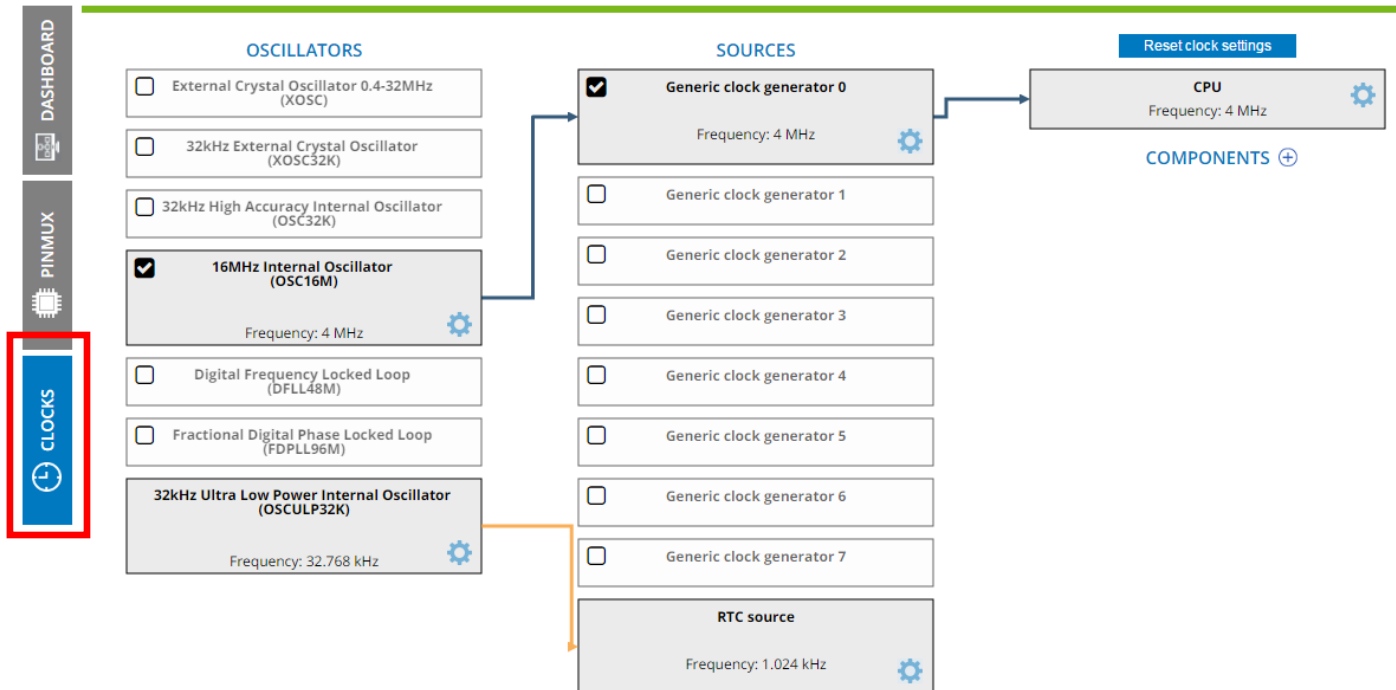


TO DO

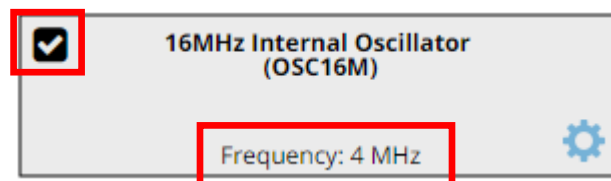
Configure Oscillators and Clocks using Atmel START Clock Configurator

- Select CLOCKS to access the CLOCK CONFIGURATOR tool:

CLOCK CONFIGURATOR



- Check that the internal OSC16M oscillator is enabled and is configured to run at 4MHz:




INFO

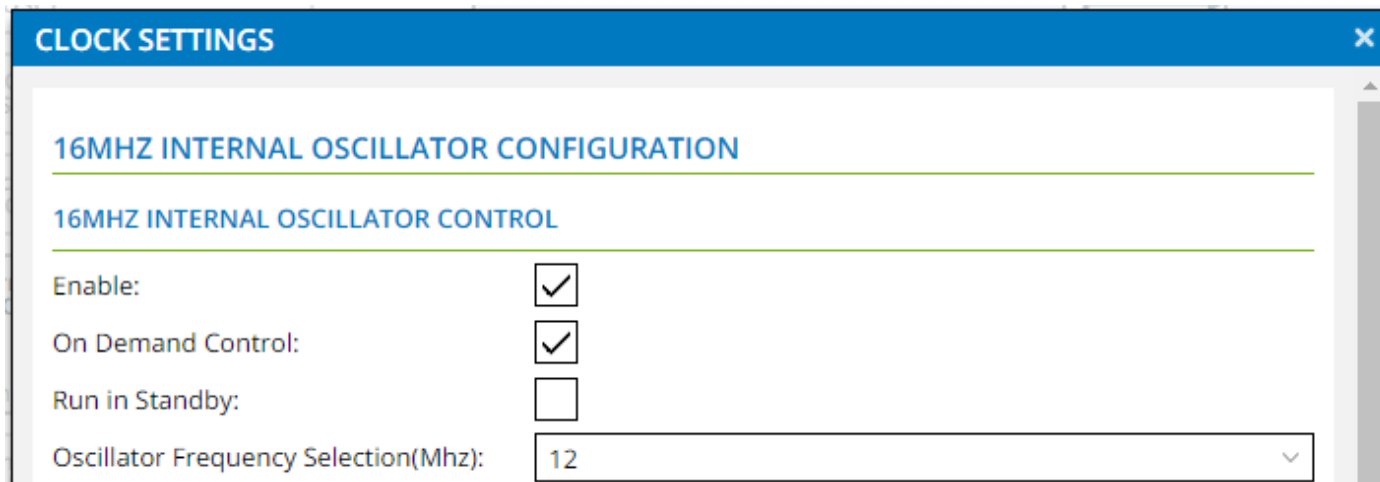
4MHz is the default SAM L21 operating frequency at power up.



INFO

You can check the 16MHz Internal Oscillator (OSC16M) is enabled by default (Enable checkbox) as the 32kHz Ultra Low Power Internal Oscillator (OSCULP32K).

- Click on the OSC16M cog wheel  and update the Oscillator Frequency from 4MHz to 12MHz then click on Close:



CLOCK SETTINGS

16MHZ INTERNAL OSCILLATOR CONFIGURATION

16MHZ INTERNAL OSCILLATOR CONTROL

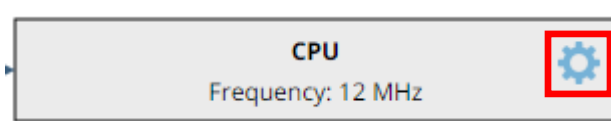
Enable: ☒

On Demand Control: ☒

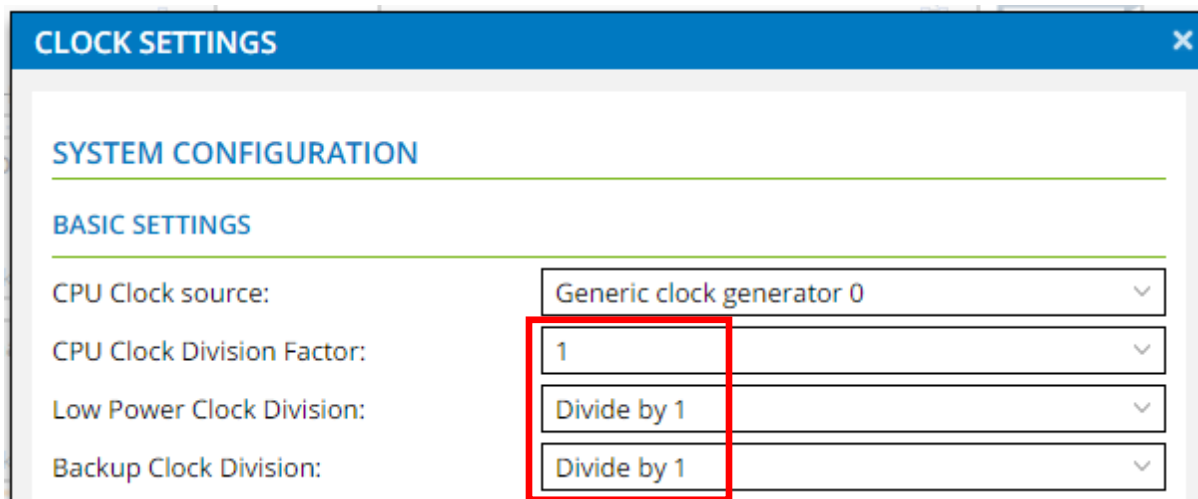
Run in Standby: ☐

Oscillator Frequency Selection(Mhz):

- Open the CPU Clock Settings Dialog window by clicking on the cog-wheel:



- Set the Low Power and Backup Clock Dividers to 1:



CLOCK SETTINGS

SYSTEM CONFIGURATION

BASIC SETTINGS

CPU Clock source:

CPU Clock Division Factor:

Low Power Clock Division:

Backup Clock Division:

Running out of flash at 12MHz requires to set the right number of wait state(s) for the flash:

46.10. NVM Characteristics

Table 46-37. NVM Max Speed Characteristics

	Conditions	CPU Fmax (MHz)			
		0WS	1WS	2WS	3WS
PL0 (-40/85°C)	V _{DDIN} >1.6 V	6	12	12	12
	V _{DDIN} >2.7 V	7.5	12	12	12
PL2 (-40/85°C)	V _{DDIN} >1.6 V	14	28	42	48
	V _{DDIN} >2.7 V	24	45	48	48

- Set the NVM Wait States to 1 then click on Close:

CLOCK SETTINGS

SYSTEM CONFIGURATION

BASIC SETTINGS

CPU Clock source:

Generic clock generator 0

CPU Clock Division Factor:

1

Low Power Clock Division:

Divide by 1

Backup Clock Division:

Divide by 1

NVM SETTINGS

NVM Wait States:

1

2.3 Add and Configure the External Interrupt Controller Peripheral

As described in the Atmel® | SMART™ SAM L21 Low-Power Modes paragraph, we can use any asynchronous clocks (interrupts) as wake-up sources for both the IDLE and STANDBY whereas only a few wake-up sources are available to exit the BACKUP sleep mode.

The OFF mode requires the assertion of the RESETN pin (RESET button on the board) to wake-up.

So we just need to identify the wake-up sources of the IDLE, STANDBY and BACKUP sleep modes.

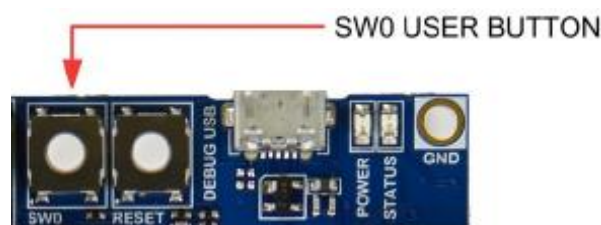
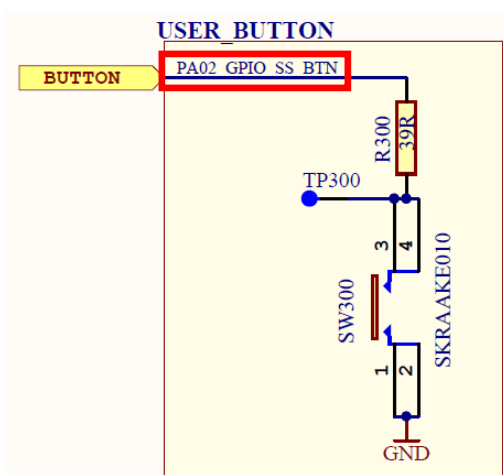


TO DO

Find out the Wake-up Source(s) of the different sleep modes

The External Interrupt Controller (EIC) allows external pins called EXTINT[x] to be configured as interrupt lines. So any EIC interrupts can be used to wake up the device from both **IDLE** and **STANDBY** sleep modes.

On the SAM L21 Xplained Pro B, we have one mechanical button called **SW0** which is connected to the I/O **PA02** of the device:



That I/O is multiplexed with the EIC external interrupt pin EXTINT[2]:

Table 2-1. PORT Function Multiplexing Table extract from SAM L21 Product Datasheet

I/O pin	Supply	A
		EIC/RSTC
PA00	VSWOUT	EXTINT[0]/ EXTWAKE[0]
PA01	VSWOUT	EXTINT[1]/ EXTWAKE[1]
PA02 (5)	VDDANA	EXTINT[2]/ EXTWAKE[2]

As a consequence, we will use PA02 and so SW0 mechanical push button as wake-up sources for both the IDLE and STANDBY sleep modes.

Concerning the BACKUP sleep mode, ones of its wake-up sources are the External Wake-up pins called EXTWAKE[x].

On the SAM L21, PA02 I/O is also multiplexed with that external wake-up pin EXTWAKE[2]:

Table 2-2. PORT Function Multiplexing Table extract from SAM L21 Product Datasheet

I/O pin	Supply	A
		EIC/RSTC
PA00	VSWOUT	EXTINT[0]/ EXTWAKE[0]
PA01	VSWOUT	EXTINT[1]/ EXTWAKE[1]
PA02 (5)	VDDANA	EXTINT[2]/ EXTWAKE[2]

As a consequence, we will also use PA02 and so SW0 mechanical push button to exit from BACKUP sleep mode.



RESULT

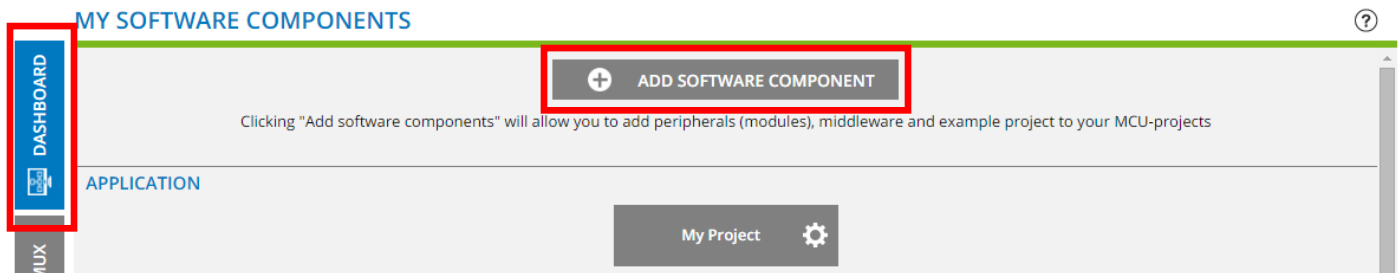
SW0 (PA02) button will be used as our wake-up source for the different sleep modes (excluding the OFF mode) which requires to add the External Interrupt Controller (EIC) to be able to support it.




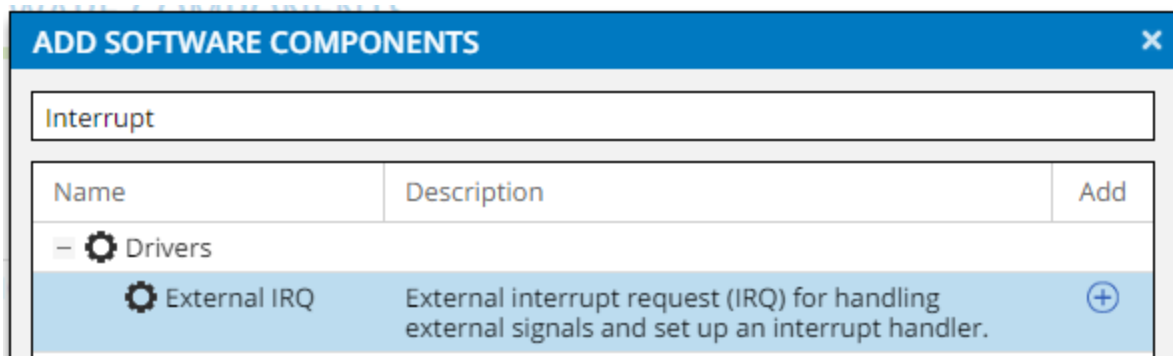
TO DO

Add the External Interrupt Controller (EIC) driver

- In Atmel START, select DASHBOARD and click on “Add software component”:

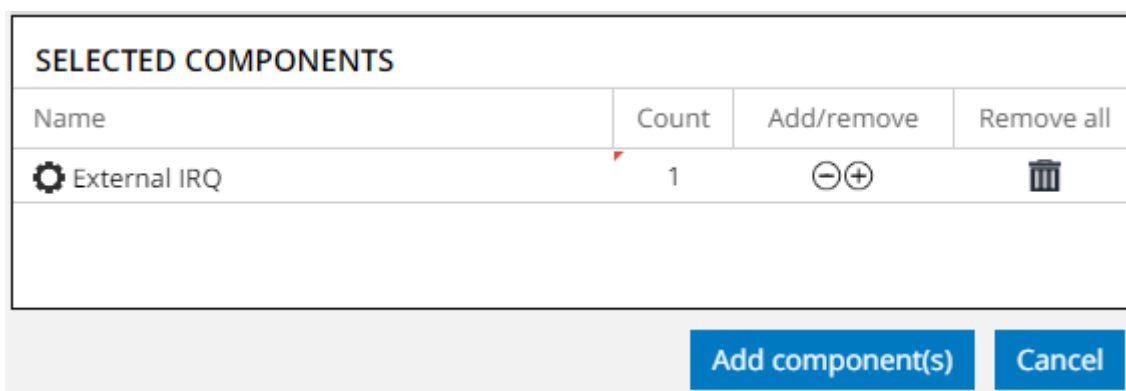


- Type “Interrupt” in the filter, select External IRQ driver and add it :



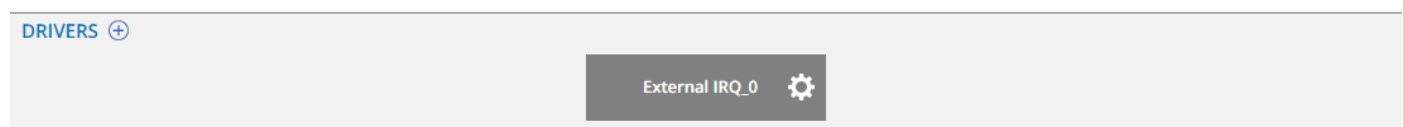
External IRQ is now displayed in the SELECTED COMPONENTS view

- Complete the addition of the EIC driver by clicking on Add component(s):



RESULT

The EIC driver is added to the application.





TO DO

Configure the External Interrupt Controller (EIC) driver

- Click on EXTERNAL_IRQ_0 component to allow its configuration
- Select for the EXTINT2 signal, the PA02 I/O line which is connected to our SW0 mechanical button:

EXTERNAL_IRQ_0



SIGNALS		
EXTINT/0:	----	Enabled
EXTINT/1:	----	Enabled
EXTINT/2:	PA02	Enabled
EXTINT/3:	----	Enabled

- In BASIC SETTINGS, change Clock Selection to “Clocked by ULPOSC32K” which is the SAM L21 internal ultra-low power 32kHz source in order to optimize power consumption:

HAL:DRIVER:EXT IRQ (DEFAULT) CONFIGURATION ON EIC

BASIC SETTINGS

Clock Selection:

Clocked by ULPOSC32K

- Enable EIC Interruption for the EXTINT2 I/O line:
 - Enable its Filter in order to remove potential spikes
 - Configure the Sense configuration as “Falling-edge detection” in order to generate an interruption on a falling edge which will correspond to a push on the button.

INTERRUPT 2 SETTINGS

Enable: ☒

External Interrupt 2 Filter Enable:



External Interrupt 2 Event Output Enable:



Input 2 Sense Configuration:

Falling-edge detection

External Interrupt 2 Asynchronous Edge Detection Mode:



RESULT

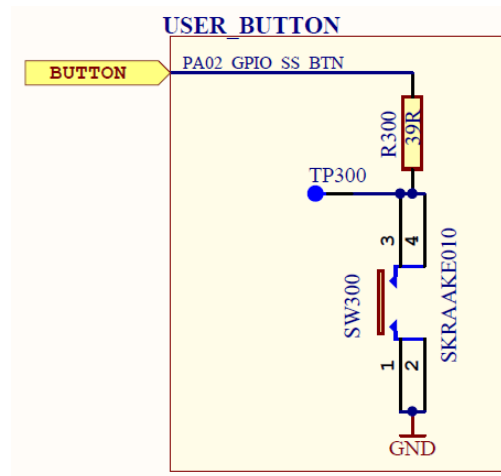
The EIC driver is added and configured.

2.4 Configure SW0 Push Button using Atmel Start PINMUX CONFIGURATOR

During reset, all SAM L21 I/O lines are configured as inputs with input buffers, output buffers and pull-up/pull-down disabled.

In our application, we want to detect a press button event on SW0 (PA02) which requires to have a logical '1' level on that I/O when the button is released.

However, as there is no external pull-up resistor connected to the push button (only a serial resistor as represented below), we must enable the PA02 internal pull-up before enabling the External Interrupt Capability:



So, to complete our application configuration in Atmel Start, we need to use the PINMUX CONFIGURATOR tool in order to setup properly the I/O default levels after reset.

The PINMUX Configurator presents an overview of all the configured pins. From the pin assignment, we can select a pin and change its pull up configuration.



TO DO Enable Internal Pull-up on PA02

- In Atmel START, select PINMUX and click on PA02 which is listed as the EXTERNAL_IRQ_0:

PINMUX CONFIGURATOR

# ↑	Pin label		Board label		Mode	Signal	
	Pad	User	Header	Pin		Label	Mode
External IRQ_0							
3	PA02	Buttons,...	SW0...	Digital ...	EXTIN...	Enabled	
No software components							
1	PA00		32kHz X...	XIN32			
2	PA01		32kHz X...	XOU...			
4	PA03		EXT1	ADC(-)			
5	PB04		EXT1	IRQ			
6	PB05		EXT1	ADC...			
7	GND...						

- Update PA02 I/O configuration:
 - Select Digital input as Pin mode
 - Select Pull-up as Pull configuration

User label:

Pin mode: Digital input

Pull configuration: Pull-up



RESULT PA02 is configured as an input with pull-up enabled.

2.5 Save and Export the Application on an Atmel Studio 7 project

We have now finished to create and configure our Atmel Start based project.

It's time now to export it as a project for Atmel Studio 7.

But before that, it's preferable to save the different configurations we did in case we need to come back later on and make some updates. For that, we will save its configuration.



TO DO Save Project Configuration

- Select SAVE CONFIGURATION, give a File Name then Click on Download Configuration

The screenshot shows the top navigation bar of the Atmel START web interface. The 'SAVE CONFIGURATION' button is highlighted with a red rectangular box. Below the navigation bar, the page title is 'HOW TO SAVE YOUR CONFIGURATION'. The left sidebar shows 'DASHBOARD' and 'MUX' selected. The main content area has two sections: 'DOWNLOAD CONFIGURATION' and 'HOW TO OPEN A CONFIGURATION LATER'. In the 'DOWNLOAD CONFIGURATION' section, the 'File name:' input field contains 'LowPowerModes.atstart' and the 'Download Configuration' button is highlighted with a red rectangular box.



RESULT Your application configuration has been saved in *.atstart file format



TO DO Export Project

- Select EXPORT PROJECT, give a File Name then Click on DOWNLOAD PACK

The screenshot shows the top navigation bar of the Atmel START web interface. The 'EXPORT PROJECT' button is highlighted with a red rectangular box. Below the navigation bar, the page title is 'EXPORT PROJECT'. The left sidebar shows 'DASHBOARD' and 'PINMUX' selected. The main content area has two sections: 'DOWNLOAD YOUR CONFIGURED PROJECT' and 'WHAT TO DO NEXT?'. In the 'DOWNLOAD YOUR CONFIGURED PROJECT' section, the 'Specify file name (optional):' input field contains 'LowPowerModes' and the 'DOWNLOAD PACK' button is highlighted with a red rectangular box.



RESULT Your application project has been exported in a *.atzip format for Atmel Studio 7

We have completed the creation and the configuration of the application. We are now ready to start coding.

3. Assignment 2: Implement SAM L21 Low Power Modes

In this assignment, we will implement the different low power modes of the SAM L21 using the project we have exported from Atmel Start.

In order to measure the different power consumptions and compare them to the product datasheet values, we will configure the application to be in the same conditions as those chosen as reference in the datasheet (see SAM L21 Power Consumption section in the Electrical Characteristics).

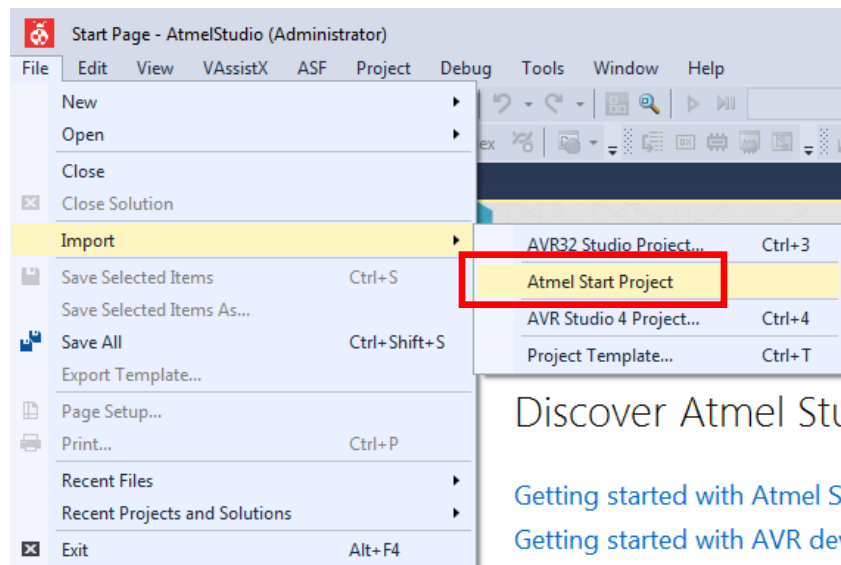
Before implementing the different power consumption modes, let's create the Atmel Studio project from the Atmel Start project configuration.



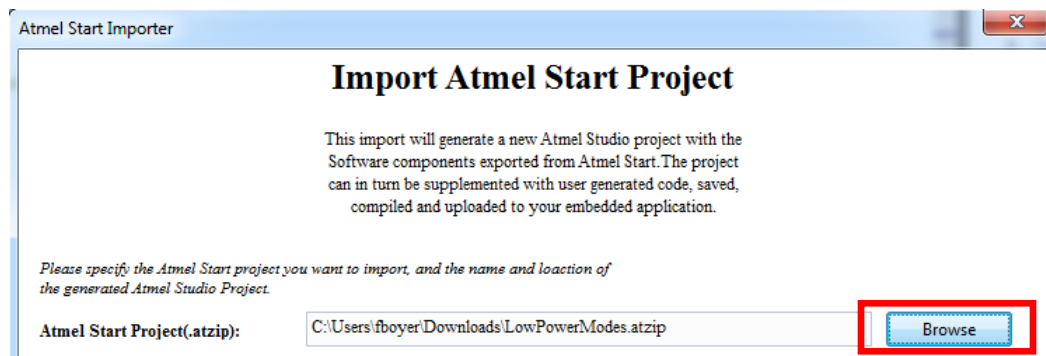
TO DO

Import Atmel Start Project

- Open Atmel Studio 7
- Select File > Import > Atmel Start Project:



- Select the project (*.atzip) you have exported from Atmel Start and click OK:



RESULT

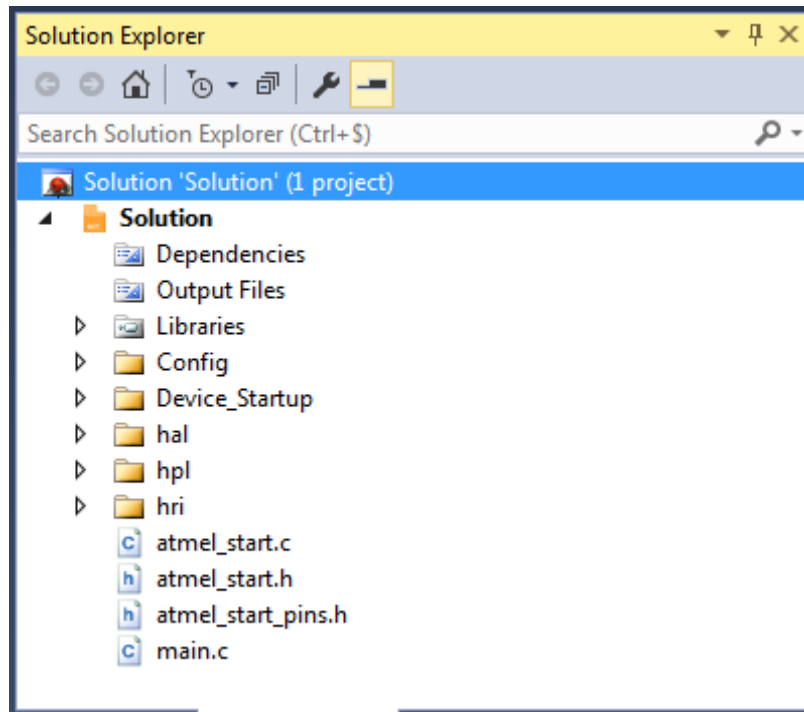
The project is created.



TO DO

Review Reference Project

Any Atmel Start-based project adds peripherals driver initialization as well as some implementation examples in order to get started very quickly: `atmel_start.c` is the file which includes all of them.



INFO

In our application, we will only find External Interrupt Controller (EIC) functions as this is the only peripheral we have selected in Atmel Start.

So, the application is going to be developed in the `main.c` file, we will call and adapt (if required), some of the functions provided in `atmel_start.c` to build our project very quickly.

- Open `main.c` file. You will see that the only function called in the current project is the `system_init()` function:

```
int main(void)
{
    system_init();

    /* Replace with your application code */
    while(1) {
    }
}
```

That function is implemented in `atmel_start.c` and includes all initialization code, which:

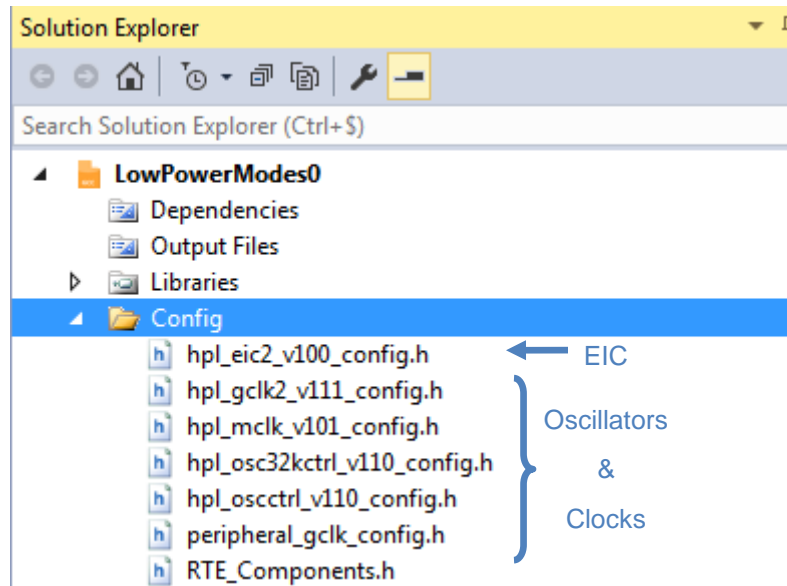
- Initializes the MCU (oscillators, clocks, flash wait states...) using `init_mcu()` function.
- Initializes the peripherals which have been selected in Atmel Start UI (the EIC in our case).

The different initialization functions which are called in `system_init()`, use the configuration's parameters that the user has selected during the Atmel Start UI configuration process.



INFO

You can retrieve these configurations in the `Config` folder:



As an example, if you open the `hpl_oscctrl_vxxx_config.h` file, you can check that the only oscillator enabled is the OSC16M:

```

// <h> 16MHz Internal Oscillator Control
// <q> Enable
// <i> Indicates whether 16MHz Internal Oscillator is enabled or not
// <id> osc16m_arch_enable
#ifndef CONF_OSC16M_ENABLE
#   define CONF_OSC16M_ENABLE 1
#endif

```

And that the selected frequency for the OSC16M oscillator is 12MHz:

```

// <y> Oscillator Frequency Selection(Mhz)
// <OSCCTRL_OSC16MCTRL_FSEL_4_Val> 4
// <OSCCTRL_OSC16MCTRL_FSEL_8_Val> 8
// <OSCCTRL_OSC16MCTRL_FSEL_12_Val> 12
// <OSCCTRL_OSC16MCTRL_FSEL_16_Val> 16
// <i> This defines the oscillator frequency (Mhz)
// <id> osc16m_freq
#ifndef CONF_OSC16M_FSEL
#   define CONF_OSC16M_FSEL OSCCTRL_OSC16MCTRL_FSEL_12_Val
#endif

```



RESULT

The main components of the project have been reviewed.

3.1 ACTIVE Mode Implementation

In active mode, the CPU is executing application code contrary to the different sleep modes where program execution is stopped and some modules and clock domains are automatically switched off.



INFO

After a power-on reset, the device is in ACTIVE mode.



TO DO

Find which configurations must be updated to implement the ACTIVE mode

Here are the conditions provided in the **Power Consumption** section of the SAM L21 product datasheet for the power consumption measurements:

- Operating Conditions
 - $V_{DDIN} = 3.3V$ ([SAM L21 Xplained Pro power supply](#))
 - CPU is running out of Flash with 1 wait state
 - Low Power Cache is enabled ([default reset configuration](#))
 - BOD33 is disabled
- Oscillators
 - In PL0 mode: OSC16M is enabled
 - In PL2 mode: XOSCK32K and DFLL48M are enabled
- Clocks
 - Main Clock source:
 - In PL0 mode: OSC16M = 4, 8 or 12MHz
 - In PL2 mode: DFLL48M = 48MHz (PL2)
 - All AHB & APB clocks are enabled ([default reset configuration](#))
 - CPUDIV=1, BUPDIV=1, LPDIV=1
 - I/Os are inactive in input mode with input trigger disable ([default reset configuration](#))
- Main Voltage Regulator
 - BUCK or LDO (default reset configuration)

In Atmel Start, we have already:

- Setup our project to run @12MHz running from the OSC16M.
- Set the right number of wait state(s) to run at 12MHz (\Leftrightarrow 1 Wait State).
- Set the CPU / Backup / Low-Power Clock Domains dividers to 1 (CPUDIV/BUPDIV/LPDIV).



RESULT

So, what we need to implement is:

- The disabling of the Brown-out Detector (BOD33).
- The selection of the embedded BUCK converter (best power efficient mode).
- The switch to Performance Level 0 (PL0) (best power efficient mode when running @12MHz).



TO DO

Disable BOD33

The 3.3V Brown-Out Detector (BOD33) monitors the voltage applied to the device and can be used to reset it if a Brown-Out condition occurs.

The BOD33 is controlled using the SAM L21 Supply Controller peripheral (SUPC).



INFO

Disabling it allows to reduce the device power consumption.

- Copy the following code just after `system_init()` function:

```
system_init();

/* Disable BOD33 */
SUPC->BOD33.reg &= ~SUPC_BOD33_ENABLE;

/* Replace with your application code */
while(1) {
}
```



RESULT

The 3.3V Brown-Out Detector is disabled.

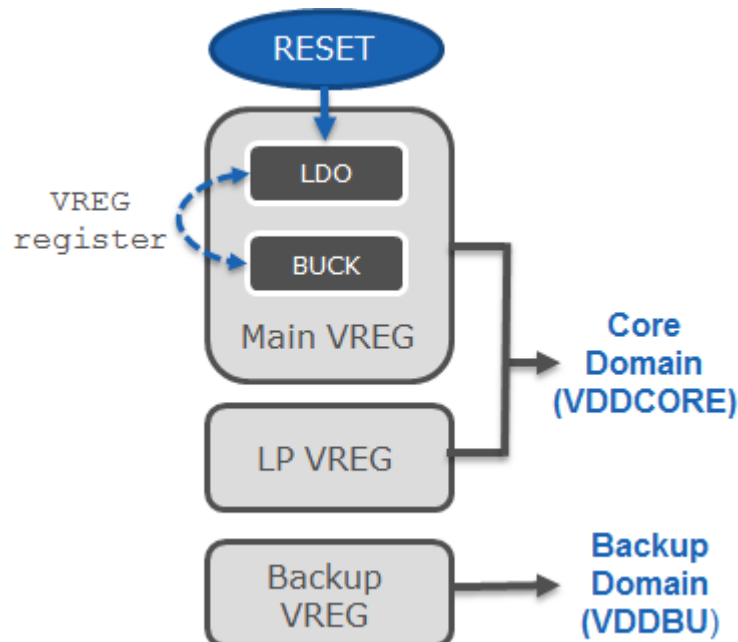


TO DO

Select Embedded BUCK Converter

The SAM L21 embeds three different voltage regulators in order to optimize the device power consumption depending on the system load in the different active and sleep modes:

- The Main Voltage regulator (MAINVREG) for ACTIVE and IDLE sleep modes.
- The Low Power voltage regulator (LPVREG) for STANDBY sleep mode.
- The Backup voltage regulator (Backup VREG) for BACKUP sleep mode.



In active mode, the main voltage regulator (MAINVREG) can be selected on the fly between the LDO (low-dropout) type regulator and the Buck converter.

The LDO main voltage regulator is enabled by default after any Reset.



INFO

Selecting the embedded BUCK allows to put the main voltage regulator in its best power efficient mode.

- Copy the following code just after BOD33 disabling:

```
/* Select BUCK converter as the main voltage regulator in active mode */
SUPC->VREG.bit.SEL = SUPC_VREG_SEL_BUCK_Val;

/* Wait for the regulator switch to be completed */
while(!(SUPC->STATUS.reg & SUPC_STATUS_VREGRDY));
```



RESULT

The embedded BUCK converter is selected.



TO DO

Select Main Voltage Regulator Performance Level 0

The SAM L21 devices have two software-selectable performance levels (PL0 and PL2) allowing the user to scale the lowest core voltage level that will support the operating frequency (dynamic voltage scaling).



WARNING

The default performance level after power-up is PL0 but the project generated by Atmel Start switches automatically the Performance Level to PL2. So, this requires the application to switch back to PL0.

Selecting the Performance Level 0, if our Main Clock is 12MHz, allows to significantly reduce the power consumption of the device:

Performance Level	VREG Mode	Max CPU Frequency	I _{VDDIN} =3,3V (Coremark or Fibonacci)
0	BUCK	12 MHz (1 WS)	32 µA/MHz
2		48 MHz (2 WS)	40 µA/MHz
0	LDO	12 MHz (1 WS)	79 µA/MHz
2		48 MHz (2 WS)	95 µA/MHz

- Copy the following code just after BUCK selection:

```
/* Set Performance Level to PL0 as we run @12MHz */
_set_performance_level(PM_PLCFG_PLSEL_PL0_Val);
```

- Add hp1_init.h include file to declare previous function:

```
#include "atmel_start.h"
#include "atmel_start_pins.h"
#include <hp1_init.h>
```



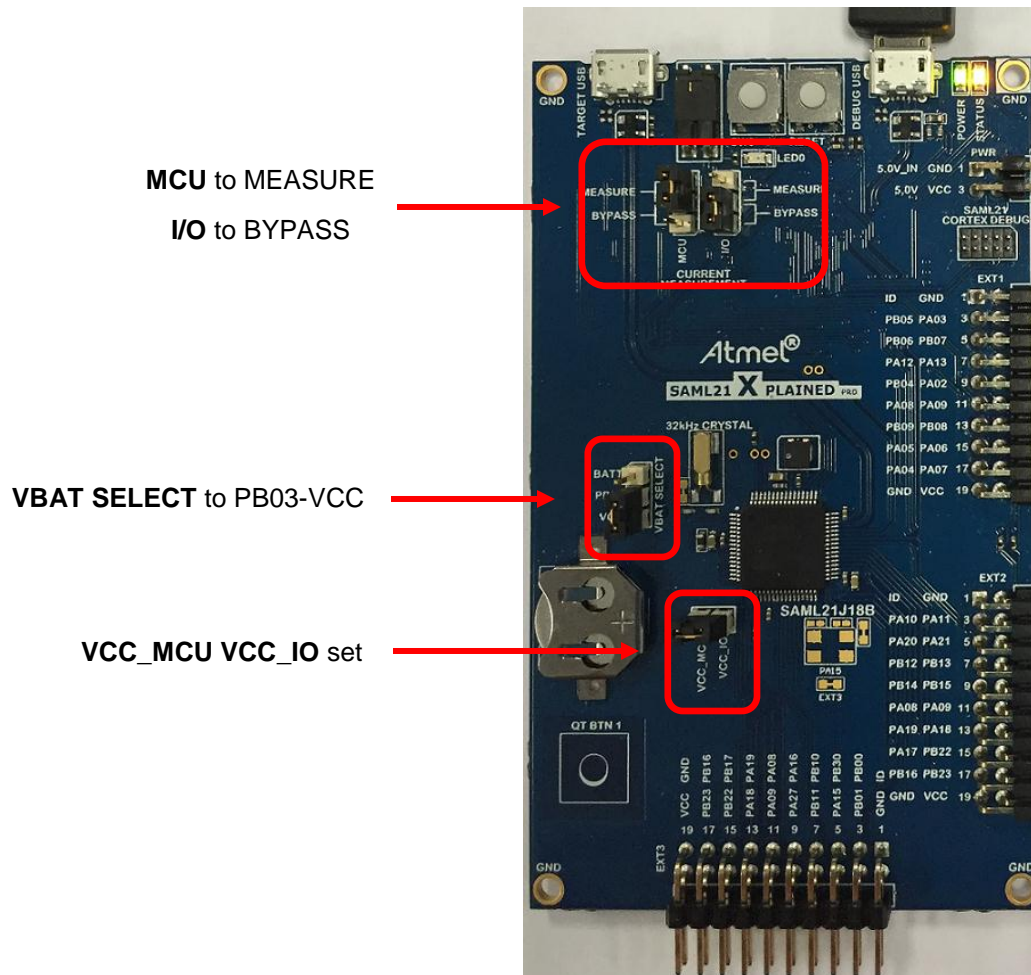
RESULT

PL0 mode is selected.



TO DO Hardware Setup

- Check the board's jumpers are correctly set:



- Power-up your SAM L21 Xplained Pro B using DEBUG USB Connector:



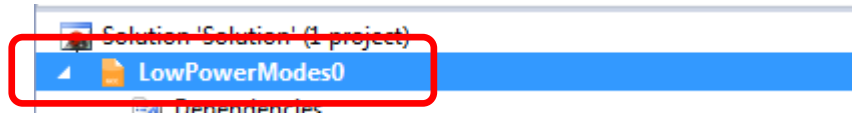
RESULT Hardware Setup is completed.



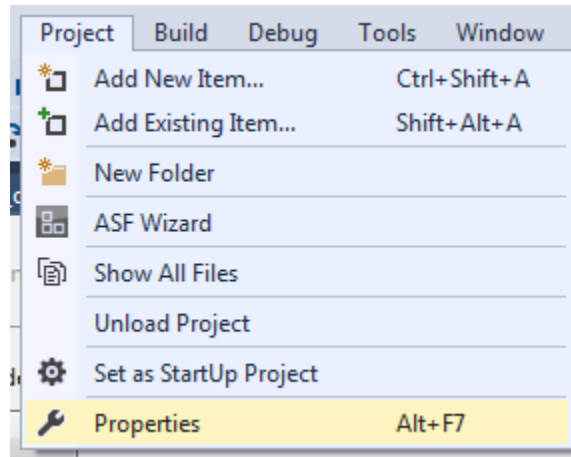
TO DO

Compile, Program and Run the Project

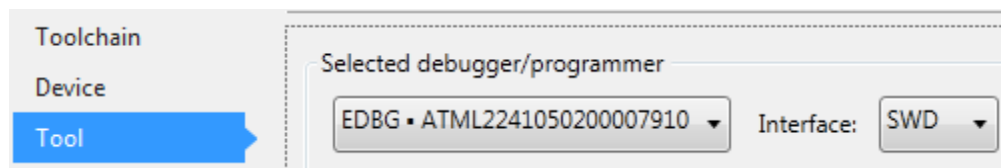
- Select SAM L21 XPRO Debugger/Programmer:
 - In Solution Explorer, select the Project:



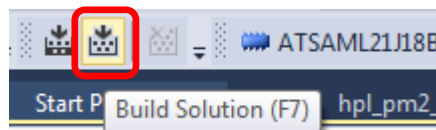
- Then, click on Project > Properties



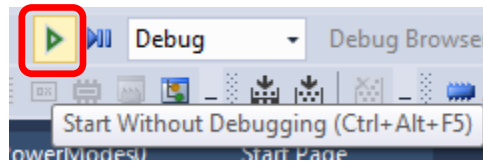
- Select Tool > EDBG as debugger/programmer and SWD as Interface:



- Compile the project by clicking on the Build Solution icon or by typing 'F7'.



- Program the application by clicking on the Start Without Debugging icon



WARNING You may be asked to update the Embedded Debugger Firmware of the board (EDBG) if this was not initially done.

In that case, please have a look at section 8 Appendix: Upgrade Embedded Debugger (EDBG) Firmware to get the procedure.

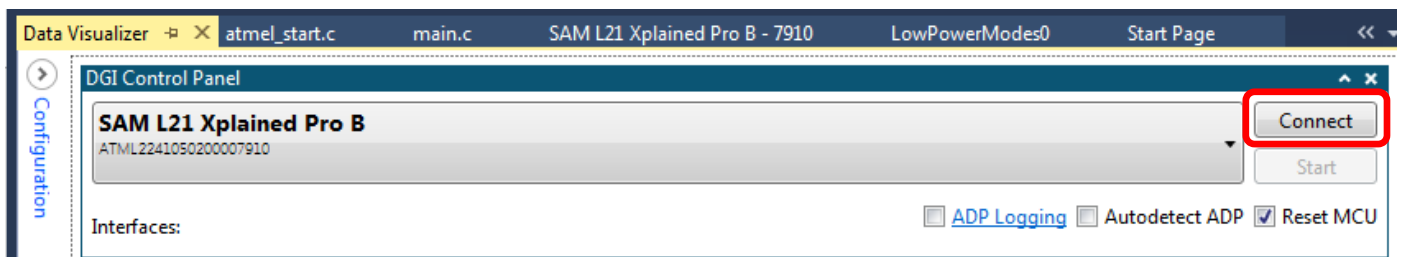


RESULT The application is programmed and runs out of the target.

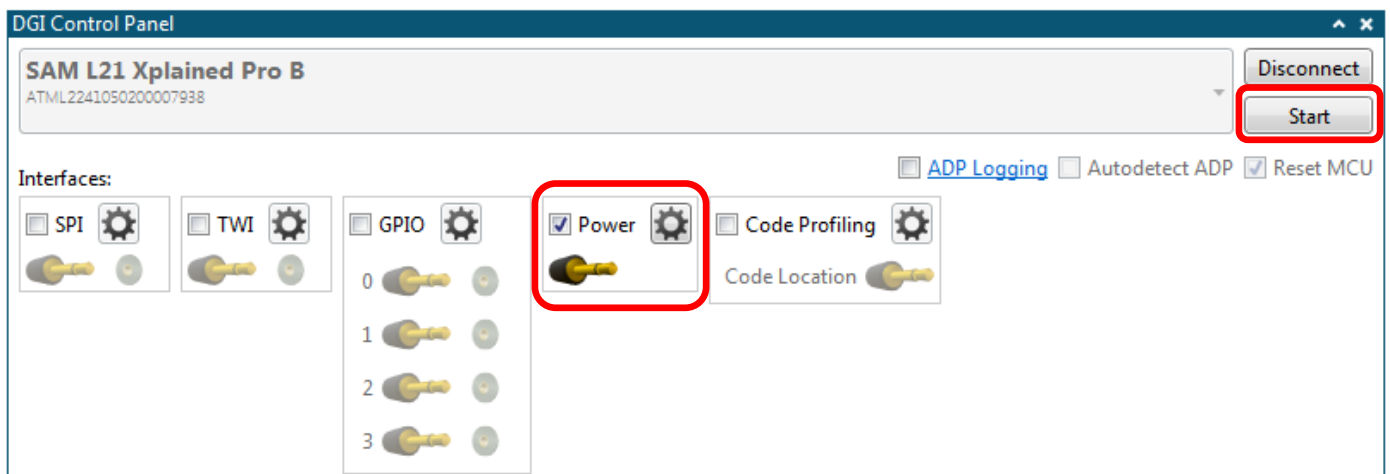


TO DO Measure Power Consumption using Data Visualizer tool

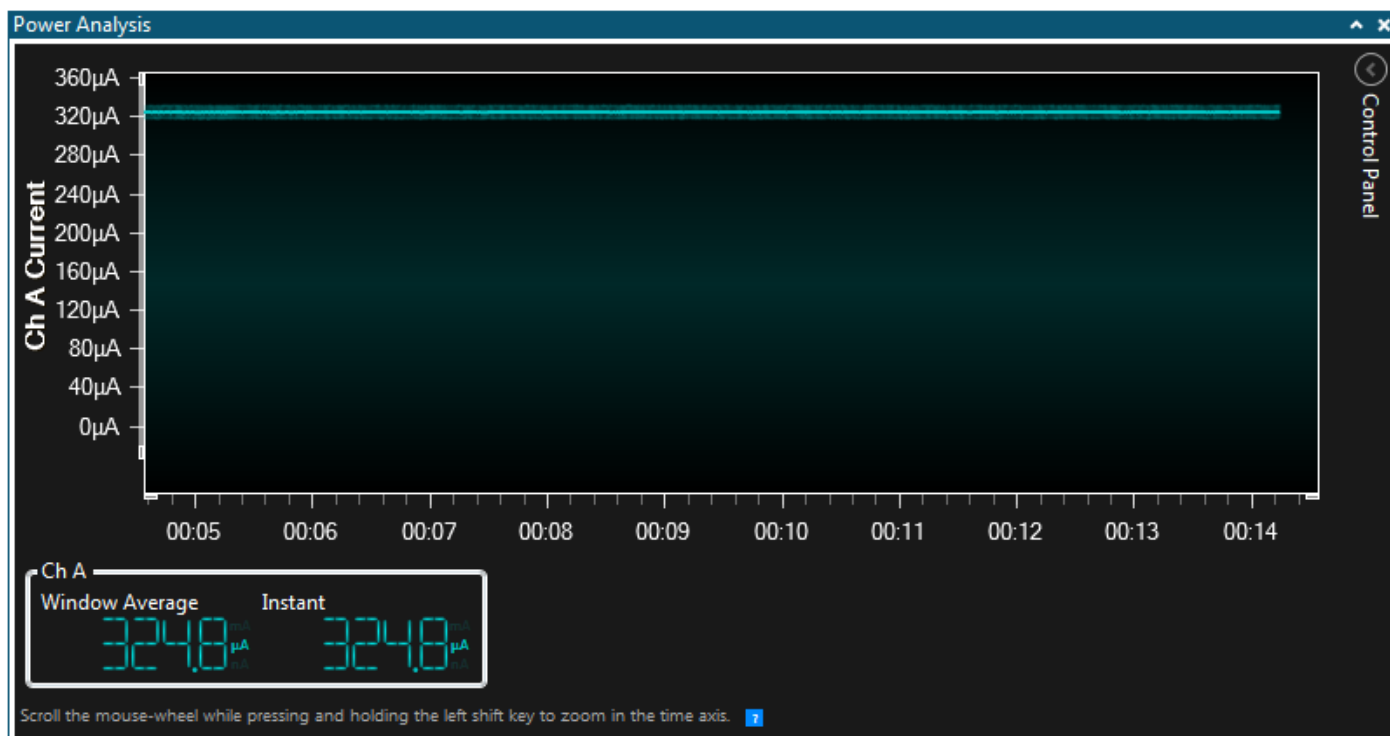
- Open Data Visualizer by clicking on Tools > Data Visualizer.
- Select SAM L21 Xplained Pro B and click on Connect.



- Select Power and click on Start.



- Read out the Windows Average value once the consumption stabilized in this mode and compute the $\mu\text{A}/\text{MHz}$ ratio:
 - Current Consumption (ACTIVE mode): μA
 - $\mu\text{A}/\text{MHz}$ ratio ($f = 12\text{MHz}$): $\mu\text{A}/\text{MHz}$



RESULT

The computed ratio should be in the range of the datasheet value:

Table 3-1. Active Current Consumption table extract from SAM L21 datasheet (Electrical Characteristics / Power Consumption section)

Mode	Conditions	Regulator	PL	Clock	Vcc	Ta	Typ.	Max.	Units
ACTIVE	WHILE1	BUCK	PL0	OSC 12MHz	1.8V		38	51	$\mu\text{A}/\text{MHz}$
					3.3V		26	41	

3.2 IDLE Sleep Mode Implementation

The IDLE mode allows power optimization with the fastest wake-up time: the CPU is stopped while all other functions can be kept running.

In our application, we use the EXTINT[2] pin (SW0 push button) from the External Interrupt Controller peripheral as interrupt source in order to wake-up from that sleep mode.

So first step is to initialize the EIC then to create a callback to handle the EIC interrupts.

Finally, we will put the application in IDLE sleep mode.



TO DO

Initialize & Configure the EIC

- Declare the `EXTERNAL_IRQ_0_init()` function after the including files:

```
#include <hpl_init.h>

extern void EXTERNAL_IRQ_0_init(void);
```



INFO

This function has been automatically generated in the `atmel_start.c` file when importing the Atmel Start project. It configures the EIC as defined in Atmel Start i.e. EXTINT[2] interrupt is connected to PA02 waiting for a falling-edge to trig an interrupt.

```
atmel_start.c SAM L21 Xplained Pro B - 7938 main.c
on_on_PA02_pressed(void)
button_on_PA02_pressed(void)
EIC_Handler(void)
EXTERNAL_IRQ_0_example(void)
EXTERNAL_IRQ_0_init(void)
GCLK_Handler(void)
MCLK_Handler(void)
OSC32KCTRL_Handler(void)
OSCCTRL_Handler(void)
system_init(void)
#include <hal_init.h>
#include <hpl_irq.h>
#include <hri_gclk_v111.h>
#include <hri_mclk_v101.h>
#include <peripheral_gclk_config.h>
#if CONF_DMAC_MAX_USED_DESC > 0
#endif

extern struct _irq_descriptor *_irq_table[PERIPH_COUNT_IRQn];
extern void Default_Handler(void);

void EXTERNAL_IRQ_0_init(void)
{
    hri_gclk_write_PCHCTRL_reg(GCLK, EIC_GCLK_ID, CONF_GCLK_EIC_SRC |
        ( 1 << GCLK_PCHCTRL_CHEN_Pos ));
    hri_mclk_set_APBAMASK_EIC_bit(MCLK);
}
```

- Call the function just after `system_init()` function:

```
system_init();

/* Enable EIC External Interrupt 2 - EXTINT[2] (PA02) */
EXTERNAL_IRQ_0_init();
```

- Define the EIC EXTINT[2] Interrupt Callback above `main()` function:

```
static void cb_eic_exint2(void)
{
}

int main(void)
{
    ...
}
```

- Register the callback just after the EIC initialization:

```
system_init();

/* Enable EIC External Interrupt 2 (PA02) */
EXTERNAL_IRQ_0_init();

/* Register EIC Callback for EXTINT[2] interrupt */
ext_irq_register(PIN_PA02, cb_eic_exint2);
```



RESULT The EIC is configured to generate interruptions on a SW0 button press.



TO DO Enter in IDLE Sleep Mode

- Copy the following code just after PL0 selection:

```
/* Enter IDLE Sleep Mode */
sleep(PM_SLEEPCFG_SLEEPMODE_IDLE2_Val);
```

- Add "`hal_sleep.h`" include file to declare previous function:

```
#include "atmel_start.h"
#include "atmel_start_pins.h"
#include <hpl_init.h>
#include "hal_sleep.h"

extern void EXTERNAL_IRQ_0_init(void);
```

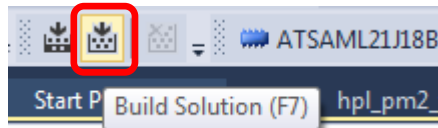


RESULT The application is programmed to enter in IDLE sleep mode.

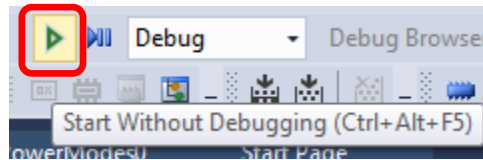


TO DO Compile and Program the Project

- Compile the project by clicking on the Build Solution icon or by typing 'F7'.



- Program the application by clicking on the Start Without Debugging icon



RESULT The application is programmed and runs out of the target.



TO DO

Measure Power Consumption using Data Visualizer tool

- Read out the Windows Average value once the consumption stabilized in this mode and compute the $\mu\text{A}/\text{MHz}$ ratio:
 - Current Consumption (IDLE mode): μA
 - $\mu\text{A}/\text{MHz}$ ratio ($f = 12\text{MHz}$): $\mu\text{A}/\text{MHz}$



RESULT

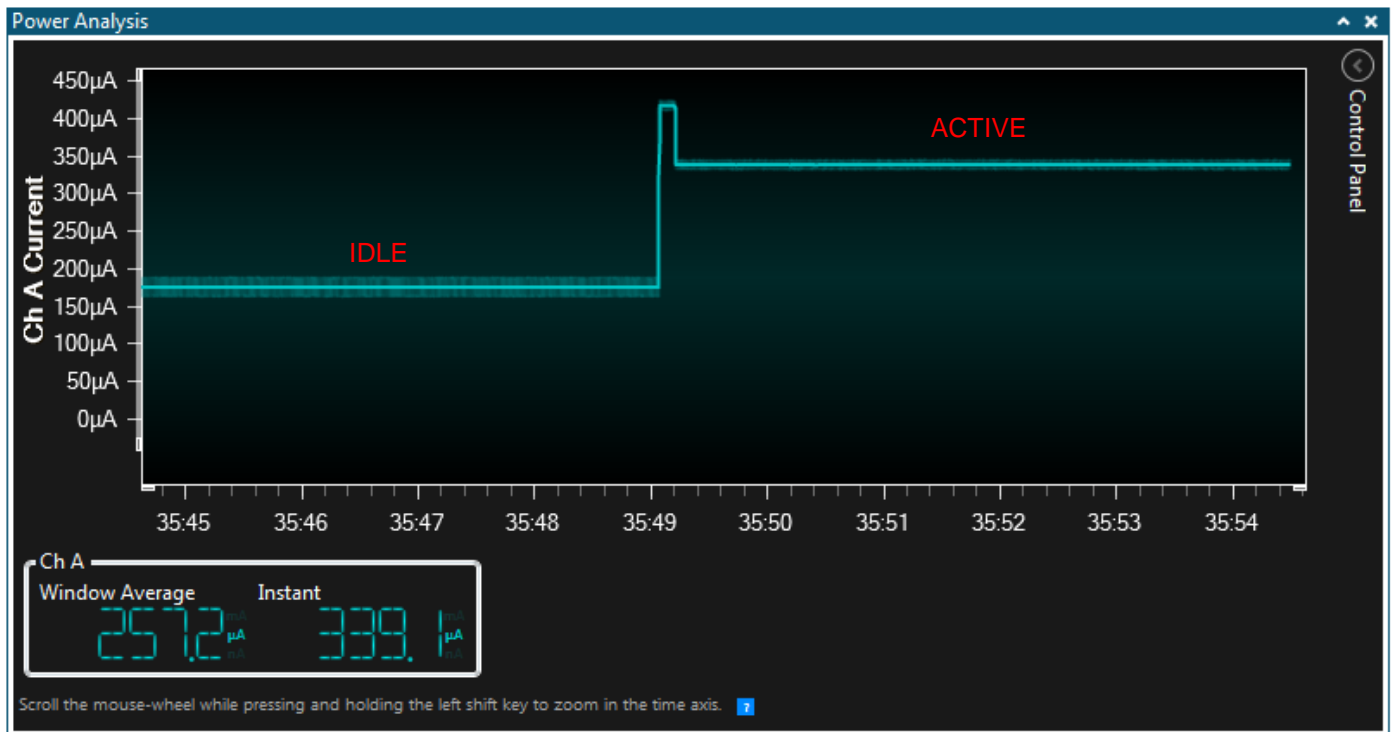
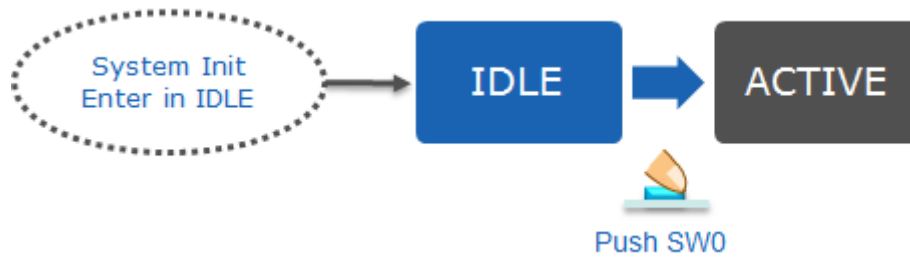
The computed ratio should be in the range of the datasheet value:

Table 3-2. Active Current Consumption table extract from SAM L21 datasheet (Electrical Characteristics / Power Consumption section)

Mode	Conditions	Regulator	PL	Clock	Vcc	Ta	Typ.	Max.	Units
IDLE		BUCK	PL0	OSC 12MHz	1.8V	Max at 85°C	17	30	
					3.3V	Typ at 25°C	13	24	

INFO

You can check that by pressing the SW0 button, you will exit the IDLE sleep mode and come back to the ACTIVE mode:



3.3 STANDBY Sleep Mode Implementation

In STANDBY sleep mode, all clocks and functions are stopped except those selected to continue running.

In this mode, all RAMs and logic contents are retained.

In our application, we still use the EXTINT[2] pin (SW0 push button) from the External Interrupt Controller peripheral as interrupts' source in order to wake-up from that sleep mode.



TO DO Enter in STANDBY Sleep Mode

The Main Voltage Regulator in STANDBY mode has a specific mode which allows to benefit from its highest efficiency: this mode is called Low Power Efficiency mode (LPEFF). This mode will be then used:



INFO This mode is only supported for a limited power supply range (2.5V to 3.6V instead of 1.62V to 3.6V).

- Copy the following code after previous implementation to enable the STANDBY sleep mode:

```
/* Enter IDLE Sleep Mode */
sleep(PM_SLEEPCFG_SLEEPMODE_IDLE2_Val);

/* Set Voltage Regulator Low power Mode Efficiency */
SUPC->VREG.bit.LPEFF = 0x1;
```

- Copy the following code after previous implementation to apply a specific erratum documented in the product datasheet (more information on the SAM L21 Errata section):

```
/* Set Voltage Regulator Low power Mode Efficiency */
SUPC->VREG.bit.LPEFF = 0x1;

/* Apply SAM L21 Erratum 15264 */
SUPC->VREG.bit.RUNSTDBY = 0x1;
SUPC->VREG.bit.STDBYPL0 = 0x1;
```

- Finally, add the following code:

```
/* Enter STANDBY Sleep Mode */
sleep(PM_SLEEPCFG_SLEEPMODE_STANDBY_Val);
```



RESULT The application is programmed to enter in STANDBY sleep mode.



TO DO Compile and Program the Project

- Compile the project by clicking on the Build Solution icon or by typing 'F7'.
- Program the application by clicking on the Start Without Debugging icon

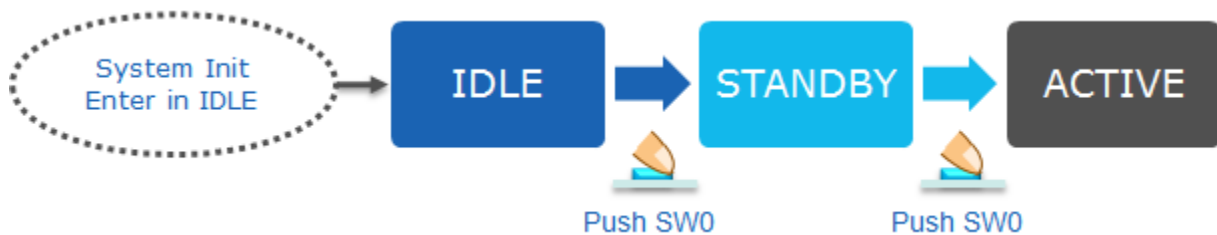


RESULT The application is programmed and runs out of the target.

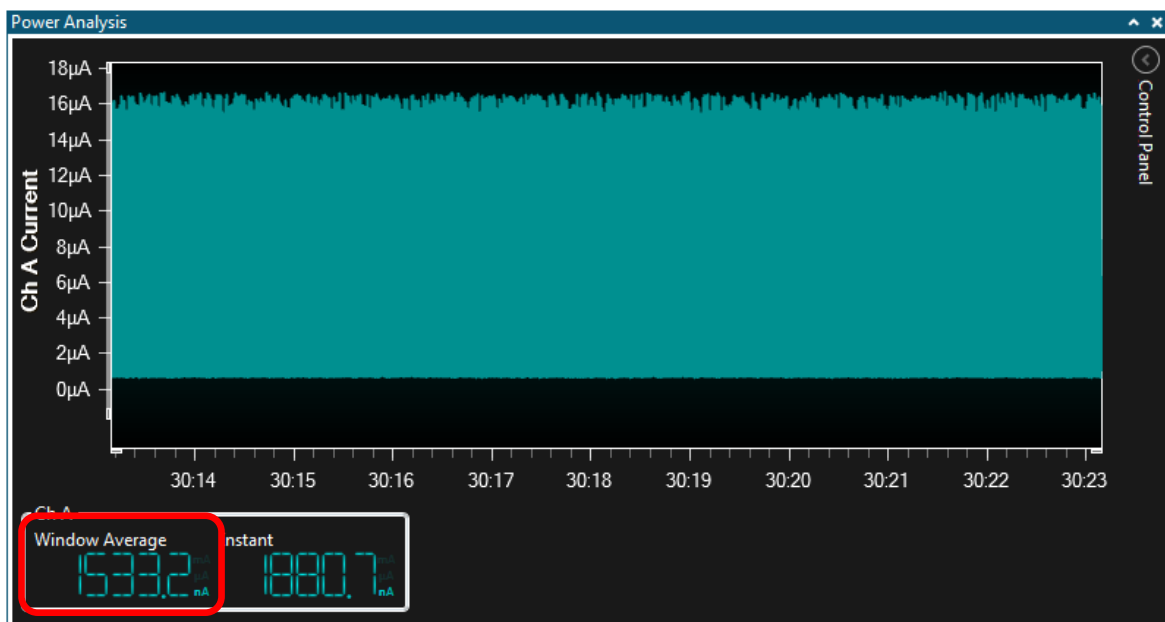


TO DO Measure Power Consumption using Data Visualizer tool

Here is the new implemented state machine of our application:



- Press SW0 button to exit IDLE and enter directly in STANDBY sleep mode
- Read out the Windows Average value once the consumption stabilized in this mode and compute the $\mu\text{A}/\text{MHz}$ ratio:
 - Current Consumption (STANDBY mode): μA



INFO This value may vary significantly depending on the temperature.



RESULT

The measured value should be in the range of the datasheet value:

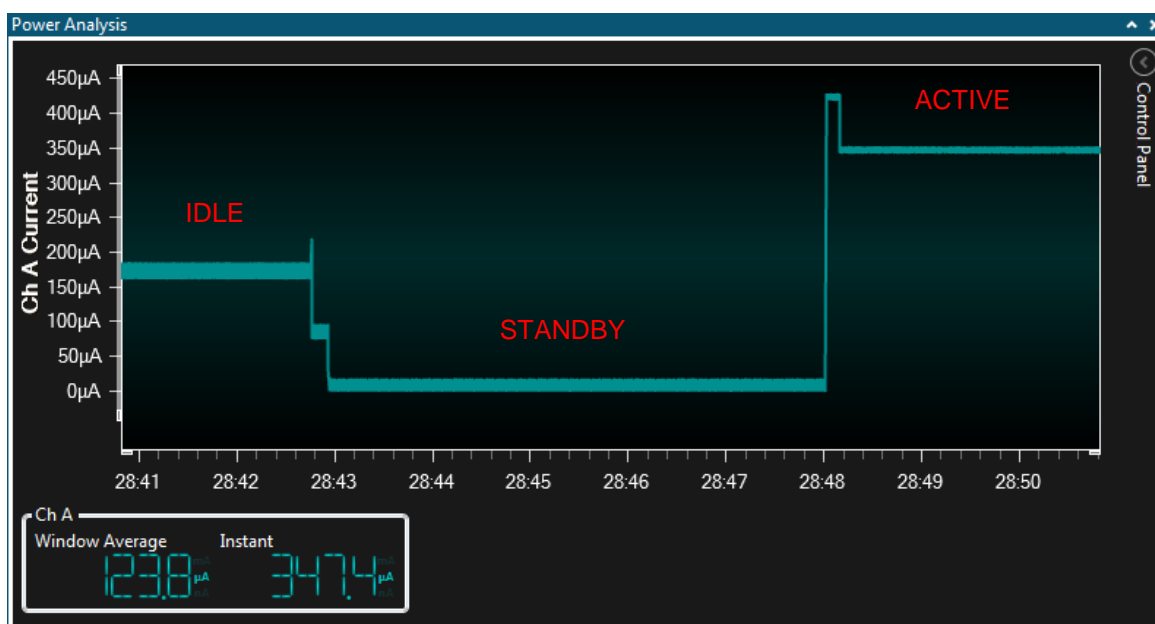
Table 3-3. Standby, Backup and Off Mode Current Consumption table extract from SAM L21 datasheet (Electrical Characteristics / Power Consumption section)

Mode	Conditions	Regulator Mode	Vcc	Ta	Typ.	Max.	Units
STANDBY	PD0, PD1, PD2 in retention state	LPEFF Disable	1.8V	25°C	1.3	3.1	µA
				85°C	23.9	60.8	
		LPEFF Enable	3.3V	25°C	1.2	2.7	
				85°C	20.7	45.8	



INFO

You can check that by pressing again the SW0 button, you will exit the STANDBY sleep mode and come back to the ACTIVE mode:



3.4 BACKUP Sleep Mode Implementation

The BACKUP mode allows achieving the lowest power consumption aside from OFF. The device is entirely powered off except for the backup domain. All peripherals in backup domain are allowed to run, e.g. the RTC can be clocked by a 32.768 kHz oscillator.

From Backup Mode, the chip can only be woken-up upon these conditions:

- Battery Backup Power Switch (BBPS): generated when the 3.3V on VDDIO is restored.
- External wake up (EXTWAKEn) pins assertion.
- Real-Time Counter interrupt.



INFO

The EXTWAKE[2] pin (SW0 button) will be used as wake-up source.



TO DO

Enter in BACKUP Sleep Mode

The wake-up level of the external wake up pin has to be configured before enabling it.

It's also possible to enable a debounce counter which is clocked by the 32 kHz.

- Add following code after previous implementation, to configure and enable the EXTWAKE[2] pin as a BACKUP wake-up source:

```
/* Enter STANDBY Sleep Mode */
sleep(PM_SLEEPCFG_SLEEPMODE_STANDBY_Val);

/* Configure EXTWAKE[2] Wakeup Debounce Counter Value to two 32kHz clock periods */
RSTC->WKDBCONF.reg = RSTC_WKDBCONF_WKDBCNT(RSTC_WKDBCONF_WKDBCNT_2CK32);
/* Configure EXTWAKE[2] Wakeup Polarity to low. */
RSTC->WKPOL.reg &= ~RSTC_WKPOL_WKPOL(0x4);
/* Enables EXTWAKE[2] feature */
RSTC->WKEN.reg = RSTC_WKEN_WKEN(0x4);
```

Backup mode exit is possible with either a high or low level EXTINT[x] pin assertion, not with an edge event. So this requires in our application to wait for the I/O to be released before entering BACKUP or we may exit directly:

- Then, add the following code to wait for SW0 button to be released:

```
/* Wait for PA02 to be released before entering in BACKUP */
while((PORT->Group[0].IN.reg & PORT_PA02)==0);
```

- Finally, add the following code to enter the sleep mode:

```
/* Enter BACKUP Sleep Mode */
sleep(PM_SLEEPCFG_SLEEPMODE_BACKUP_Val);
```



RESULT

The application is programmed to enter in BACKUP sleep mode.



TO DO Compile, Program and Run the Project

- Compile the project by clicking on the Build Solution icon or by typing 'F7'.
- Program the application by clicking on the Start Without Debugging icon

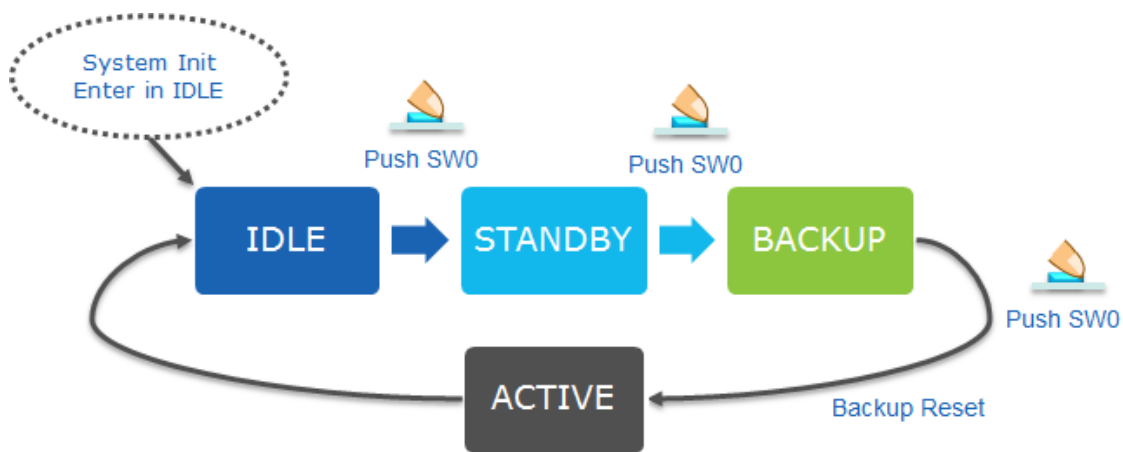


RESULT The application is programmed and runs out of the target.

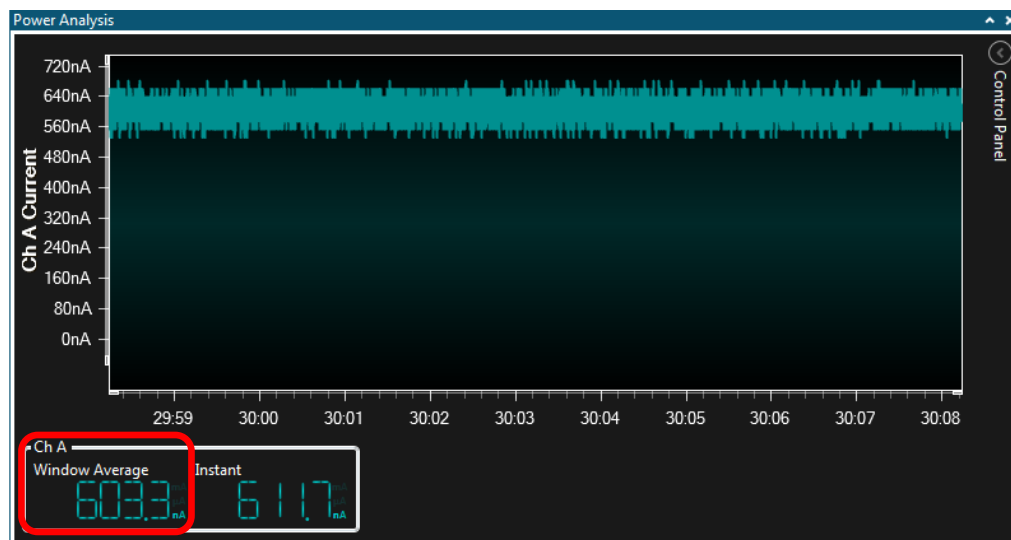


TO DO Measure Power Consumption using Data Visualizer tool

Here is the new implemented state machine of our application:



- Press SW0 button twice to exit IDLE then STANDBY and enter directly in BACKUP sleep mode
- Read out the Windows Average value once the consumption stabilized in this mode and compute the $\mu\text{A}/\text{MHz}$ ratio:
 - Current Consumption (BACKUP mode): μA





RESULT

The measured value should be in the range of the datasheet value:

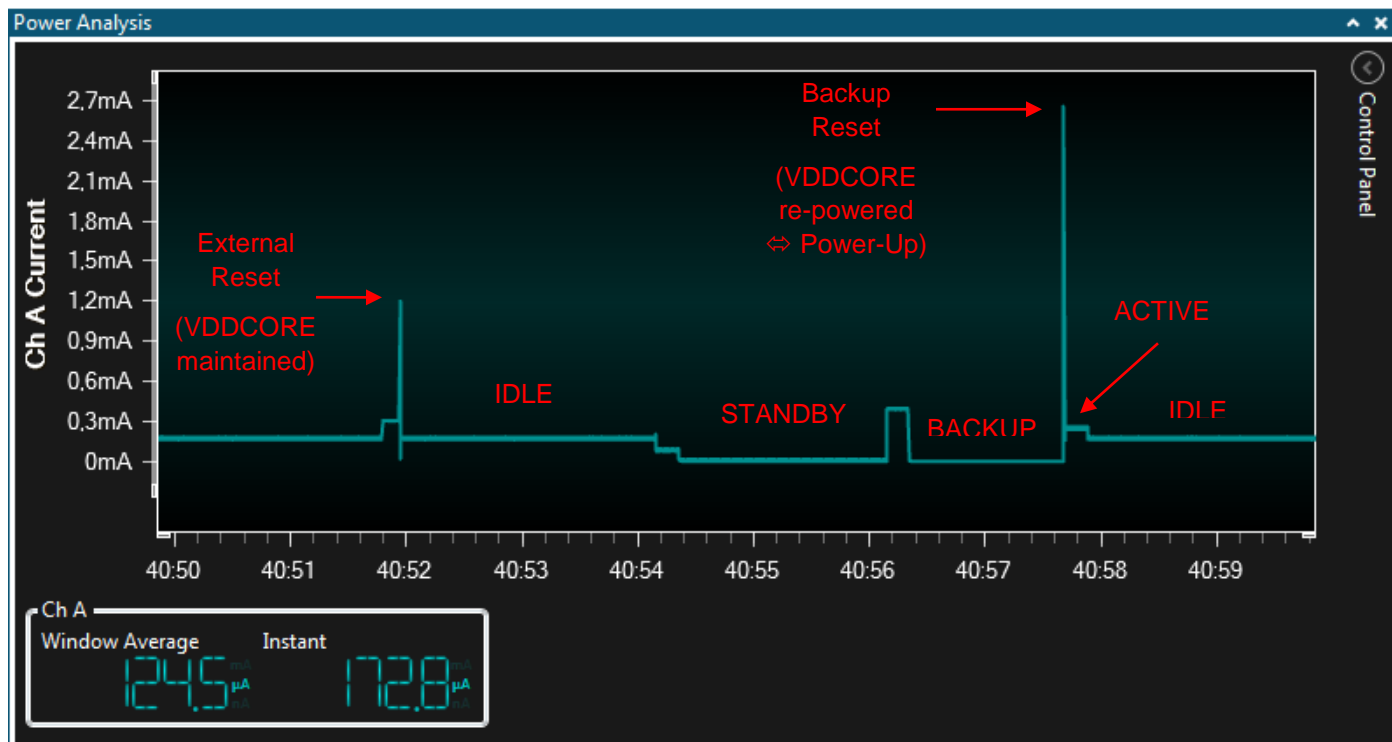
Table 3-4. Standby, Backup and Off Mode Current Consumption table extract from SAM L21 datasheet (Electrical Characteristics / Power Consumption section)

Mode	Conditions	Regulator Mode	Vcc	Ta	Typ.	Max.	Units
Measured Power Consumption	powered by VDDIN		1.8V	25°C	0.48	0.83	µA
				85°C	4.4	10.2	
			3.3V	25°C	0.59	1.1	
				85°C	5.9	14.0	
	powered by VDDIN VBAT consumption		1.8V	25°C	0.001	0.004	
				85°C	0.011	0.027	
			3.3V	25°C	0.007	0.026	
				85°C	0.036	0.076	



INFO

You can check that by pressing again the SW0 button, you will exit the BACKUP sleep mode and reset the device.



3.5 OFF Sleep Mode Implementation

In OFF mode, the device is entirely powered-off.

From Off Mode, the chip can only be woken-up by pulling the RESET pin low, or when a power Reset is done.



INFO

RESET pin is connected to the RESET button of the SAM L21 Xplained Pro so will be our wake-up source.



TO DO

Enter in OFF Sleep Mode

To test the OFF sleep mode, you need to enter it BEFORE entering in BACKUP sleep mode.



INFO

Entering OFF mode will prevent from entering BACKUP mode as its wake up source generates a RESET of the device (and vice versa)

- Copy the following code after STANDBY sleep mode entry (so before BACKUP sleep mode entry) :

```
/* Enter STANDBY Sleep Mode */
sleep(PM_SLEEPCFG_SLEEPMODE_STANDBY_Val);

/* Enter OFF Sleep Mode */
/* Entering OFF mode will prevent from entering BACKUP mode
as its wake up source generates a RESET of the device */
sleep(PM_SLEEPCFG_SLEEPMODE_OFF_Val);
```



TO DO

Compile, Program and Run the Project

- Compile the project by clicking on the Build Solution icon or by typing 'F7'.
- Program the application by clicking on the Start Without Debugging icon



RESULT

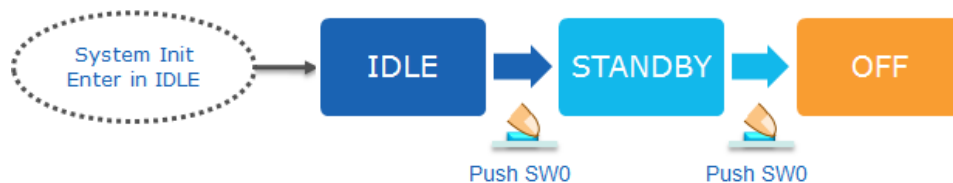
The application is programmed and runs out of the target.



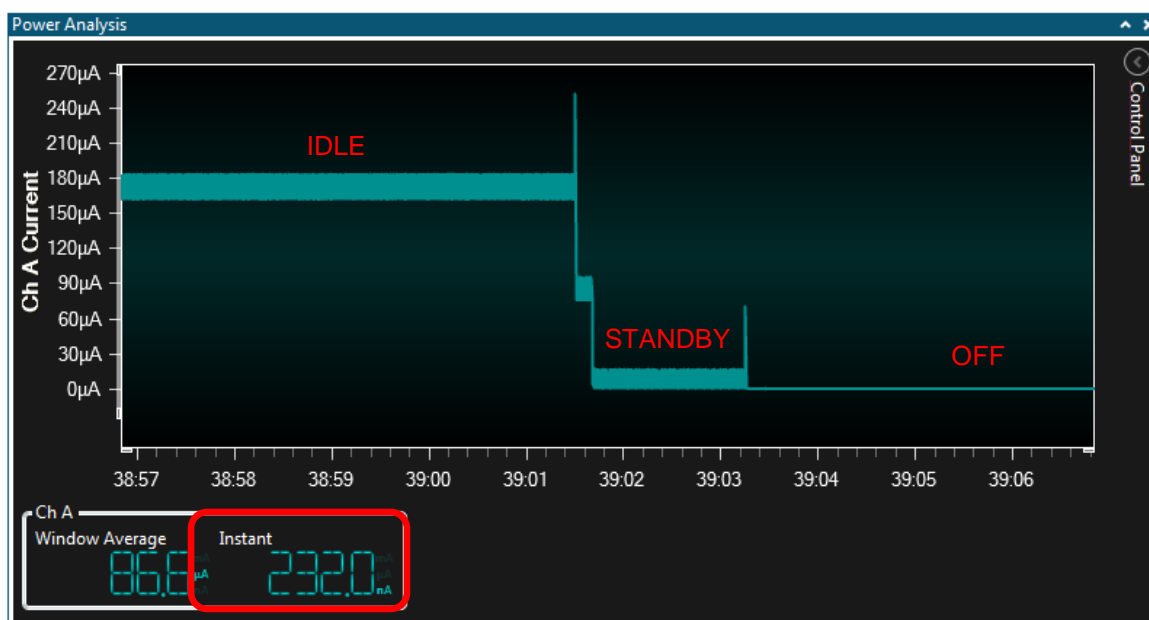
TO DO

Measure Power Consumption using Data Visualizer tool

Here is the new implemented state machine of our application:



- Press SW0 button twice to exit IDLE then STANDBY and enter directly in OFF sleep mode
- Read out the Windows Average value once the consumption stabilized in this mode and compute the $\mu\text{A}/\text{MHz}$ ratio:
 - Current Consumption (OFF mode): μA



RESULT

The measured value should be in the range of the datasheet value:

Table 3-5. Standby, Backup and Off Mode Current Consumption table extract from SAM L21 datasheet (Electrical Characteristics / Power Consumption section)

Mode	Conditions	Regulator Mode	Vcc	Ta	Typ.	Max.	Units
OFF			1.8V	25°C	0.16	0.29	μA
				85°C	2.6	5.5	
			3.3V	25°C	0.21	0.51	
				85°C	3.9	9.3	

4. Conclusion

In this hands-on, we have described and illustrated the different low power modes available in the Atmel® | SMART™ SAM L21 Series.

After completing it, you should now:

- Have a clear picture of most of the SAM L21 key power optimization features.
- Know how to use low power features in an Atmel Start-based project.
- Know how analyze power consumption on the SAM L21 Xplained Pro using Atmel Studio 7 Data Visualizer tool.

We have demonstrated some of the SAM L21 low power techniques which can be used to optimize the power consumption of any application:

- The three different voltage regulators in order to optimize the device power consumption depending on the system load in the different active and sleep modes
- The BUCK converter power efficient mode
- The Performance Level capability (dynamic voltage scaling) to adjust regulator output voltage against operating frequency
- The flexible sleep modes: IDLE, STANDBY, BACKUP and OFF.

In addition, the SAM L21 brings a unique low power technique dedicated to the STANDBY sleep mode: the Power Gating feature.

This technique allows to automatically turn off unused power domain supplies individually while keeping others powered up.

This technique combined with the SleepWalking feature (capability for a device in sleep mode, to temporarily wake-up clocks for a peripheral to perform a task without waking-up the CPU) allows to reduce the overall power consumption of the device in STANDBY mode.



INFO

You can get more info on the SleepWalking with Dynamic Power Gating feature in the dedicated SAM L21 Hands-on: *“Optimizing Power Consumption using SAM L21 SleepWalking on the SAM L21 Xplained Pro”*

5. Appendix: Solution Project (direct entry in IDLE sleep mode)

```
#include "atmel_start.h"
#include "atmel_start_pins.h"
#include <hpl_init.h>
#include "hal_sleep.h"
extern void EXTERNAL_IRQ_0_init(void);

static void cb_eic_exint2(void)
{
}

int main(void)
{
    system_init();

    /* Enable EIC External Interrupt 2 - EXTINT[2] (PA02) */
    EXTERNAL_IRQ_0_init();

    /* Register EIC Callback for EXTINT[2] interrupt */
    ext_irq_register(PIN_PA02, cb_eic_exint2);

    /* Disable BOD33 */
    SUPC->BOD33.reg &= ~SUPC_BOD33_ENABLE;

    /* Select BUCK converter as the main voltage regulator in active mode */
    SUPC->VREG.bit.SEL = SUPC_VREG_SEL_BUCK_Val;

    /* Wait for the regulator switch to be completed */
    while(!(SUPC->STATUS.reg & SUPC_STATUS_VREGRDY));

    /* Set Performance Level to PL0 as we run @12MHz */
    _set_performance_level(PM_PLCFG_PLSEL_PL0_Val);

    /* Enter IDLE Sleep Mode */
    sleep(PM_SLEEPCFG_SLEEPMODE_IDLE2_Val);

    /* Set Voltage Regulator Low power Mode Efficiency */
    SUPC->VREG.bit.LPEFF = 0x1;

    /* Apply SAM L21 Erratum 15264 */
    SUPC->VREG.bit.RUNSTDBY = 0x1;
    SUPC->VREG.bit.STDBYPL0 = 0x1;

    /* Enter STANDBY Sleep Mode */
    sleep(PM_SLEEPCFG_SLEEPMODE_STANDBY_Val);

    /* Enter OFF Sleep Mode */
    //sleep(PM_SLEEPCFG_SLEEPMODE_OFF_Val);

    /* Configure EXTWAKE[2] Wakeup Debounce Counter Value to two 32kHz clock periods */
    RSTC->WKDBCONF.reg = RSTC_WKDBCONF_WKDBCNT(RSTC_WKDBCONF_WKDBCNT_2CK32);
    /* Configure EXTWAKE[2] Wakeup Polarity to low. */
    RSTC->WKPOL.reg &= ~RSTC_WKPOL_WKPOL(0x4);
    /* Enables EXTWAKE[2] feature */
    RSTC->WKEN.reg = RSTC_WKEN_WKEN(0x4);

    /* Wait for PA02 to be released before entering in BACKUP */
    while((PORT->Group[0].IN.reg & PORT_PA02)==0);

    /* Enter BACKUP Sleep Mode */
    sleep(PM_SLEEPCFG_SLEEPMODE_BACKUP_Val);

    while(1) {
    }
}
```

6. Appendix: Project waiting in ACTIVE mode (SAML21-PowerModes project)

```
#include "atmel_start.h"
#include "atmel_start_pins.h"
#include <hpl_init.h>
#include "hal_sleep.h"

extern void EXTERNAL_IRQ_0_init(void);

/* global variables */
volatile bool exint2_flag = false;

/* EIC EXTINT2 Interrupt Callback */
static void cb_eic_exint2(void)
{
    exint2_flag = true;
}

/* Low Power Modes Entry/Exit
** ACTIVE mode can be measured after reset
** IDLE sleep mode is measured once SW0 button is pressed
** IDLE sleep mode is exited once SW0 is pressed (interrupt exit condition)
** STANDBY sleep mode is measured once IDLE sleep mode is exited
** STANDBY sleep mode is exited once SW0 is pressed (interrupt exit condition)
** BACKUP sleep mode is measured once IDLE sleep mode is exited
** BACKUP sleep mode is exited once SW0 is pressed (external wakeup condition)
** OFF sleep mode is measured if uncommented (BACKUP won't then be measured)
** OFF sleep mode is exited once RESET is pressed (external reset pin condition)
*/

int main(void)
{
    system_init();

    /* Enable EIC External Interrupt 2 - EXTINT[2] (PA02) */
    EXTERNAL_IRQ_0_init();

    /* Register EIC Callback for EXTINT[2] interrupt */
    ext_irq_register(PIN_PA02, cb_eic_exint2);

    /* Disable BOD33 */
    SUPC->BOD33.reg &= ~SUPC_BOD33_ENABLE;

    /* Select BUCK converter as the main voltage regulator in active mode */
    SUPC->VREG.bit.SEL = SUPC_VREG_SEL_BUCK_Val;

    /* Wait for the regulator switch to be completed */
    while(!(SUPC->STATUS.reg & SUPC_STATUS_VREGRDY));

    /* Set Performance Level to PL0 as we run @12MHz */
    _set_performance_level(PM_PLCFG_PLSEL_PL0_Val);

    /* Wait SW0 Button to be pressed*/
    while (!exint2_flag);
    exint2_flag = false;

    /* Enter IDLE Sleep Mode */
    sleep(PM_SLEEPCFG_SLEEPMODE_IDLE2_Val);
}
```

```

/* Set Voltage Regulator Low power Mode Efficiency */
SUPC->VREG.bit.LPEFF = 0x1;

/* Apply SAM L21 Erratum 15264 */
SUPC->VREG.bit.RUNSTDBY = 0x1;
SUPC->VREG.bit.STDBYPL0 = 0x1;

/* Enter STANDBY Sleep Mode */
sleep(PM_SLEEPCFG_SLEEPMODE_STANDBY_Val);

/* Enter OFF Sleep Mode */
/* Entering OFF mode will prevent from entering BACKUP mode
as its wake up source generates a RESET of the device */
//sleep(PM_SLEEPCFG_SLEEPMODE_OFF_Val);

/* Configure EXTWAKE[2] Wakeup Debounce Counter Value to two 32kHz clock periods */
RSTC->WKDBCONF.reg = RSTC_WKDBCONF_WKDBCNT(RSTC_WKDBCONF_WKDBCNT_2CK32);
/* Configure EXTWAKE[2] Wakeup Polarity to low. */
RSTC->WKPOL.reg &= ~RSTC_WKPOL_WKPOL(0x4);
/* Enables EXTWAKE[2] feature */
RSTC->WKEN.reg = RSTC_WKEN_WKEN(0x4);

/* Wait for PA02 to be released before entering in BACKUP */
while((PORT->Group[0].IN.reg & PORT_PA02)==0);

/* Enter BACKUP Sleep Mode */
sleep(PM_SLEEPCFG_SLEEPMODE_BACKUP_Val);
}

```

7. Appendix: Atmel Studio 7 Help Viewer

We will see some of the new Atmel Studio 7 features which will be helpful while developing the application.

It is possible to download the microcontroller data sheet /user guides /application notes within Atmel Studio 7 and also it is possible to check the relative information, register information while implementing the application.

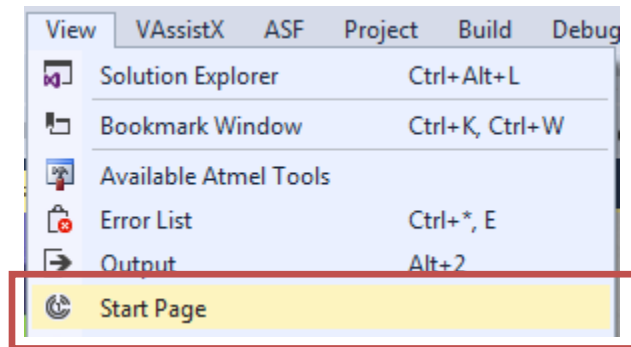
This feature makes it very easy to take a quick overview of datasheet /Application notes/user guides topics.



TO DO

Get SAM L21 Documentation

- From the menu select View -> Start page.



- Click on 'Download documentation' to launch the Atmel Studio 7 Help Viewer:

Discover Atmel Studio

[Getting started with Atmel Studio](#)

[Getting started with AVR development](#)

[Open Atmel Start Configurator](#)

[Download Atmel Studio Extensions](#)

[Download documentation](#)

- In the 'Manage Content' tab, select 'Online' and type 'SAM L21' in the search box:

[Help Viewer Home](#)
[Manage Content](#)

Add and Remove Content

Adding content will automatically refresh all local documentation with available updates

Installation source:

☒ **Online**
☐ **Disk:**

SAM L21

Name	Action
▲ Atmel	
▲ Application Notes	
▲ SAM	
AT03975: Getting Started with SAM L21 - Application Note	Add
AT03976: SAM L21 OPAMP as ADC Gain Amplifier - Application Note	Add
AT11513: ASF Manual (SAM L21) - Application Note	Add
AT12705: SAM L21 ADC Sampling using Low-Power Features - Application Note	Add
AT13382: Migrating from SAM L21 variant A to variant B - Application Note	Add
▲ Datasheets	
▲ ARM	
▲ SAM L	
SAM L21E / SAM L21G / SAM L21J - Datasheet	Add
SAM L21E / SAM L21G / SAM L21J Summary - Datasheet	Add
▲ User Guides	
▲ Xplained Pro	
SAM L21 Xplained Pro - User Guide	Add



RESULT

All the Atmel SAM L21 documentation is listed.



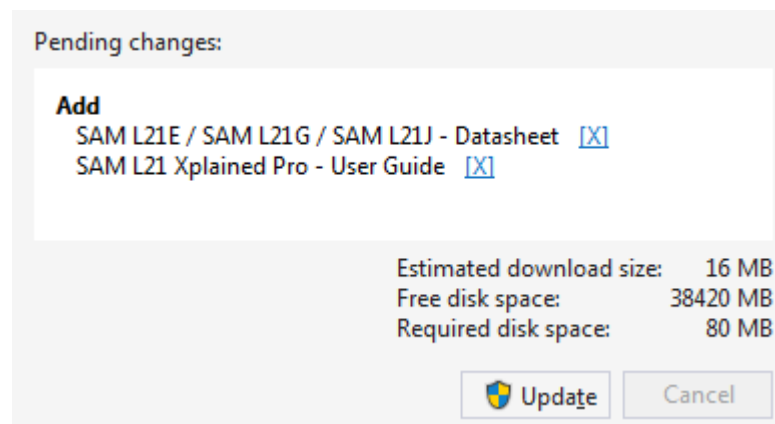
TO DO

Download the SAM L21 Datasheet / User Guide

- Select SAM L21 Datasheet and User Guide documents by clicking on their 'Add' link:

Datasheets				
ARM				
SAM L				
SAM L21E / SAM L21G / SAM L21J - Datasheet		Cancel	Add (pending)	16 MB
SAM L21E / SAM L21G / SAM L21J Summary - Datasheet		Add		1 MB
User Guides				
Xplained Pro				
SAM L21 Xplained Pro - User Guide		Cancel	Add (pending)	1 MB

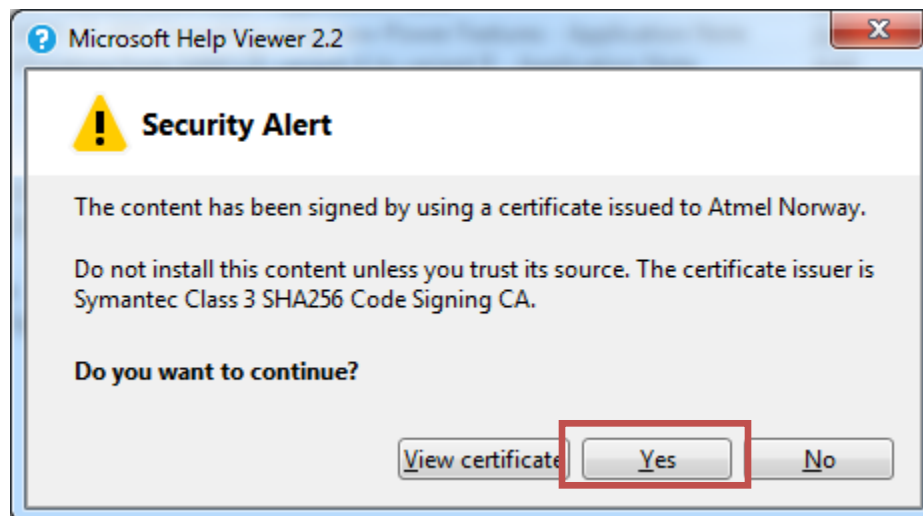
- Click on the 'Update' button



INFO

Downloading is started and download completion is indicated by the progress bar at the right bottom corner.

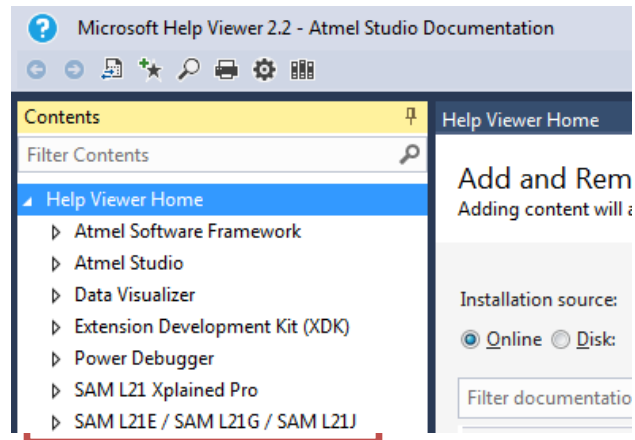
- 'Security Alert' message box will be displayed. Click 'Yes'





RESULT

The downloaded documentations are listed under 'Contents' tab



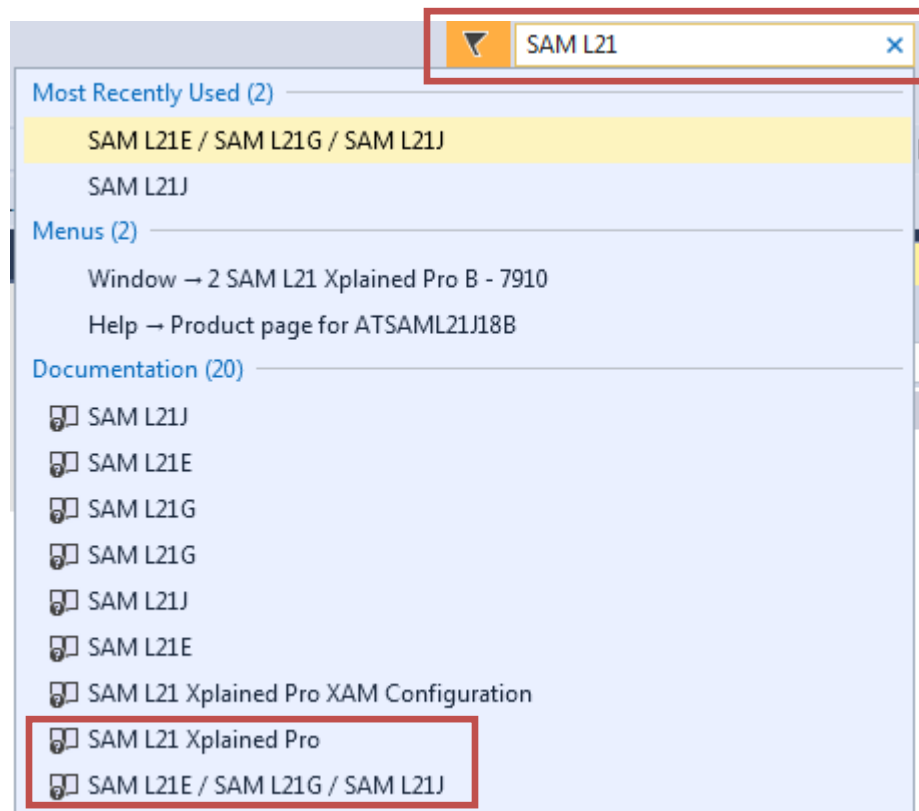


TO DO

Use Offline Documentation

There are different ways to quickly access the downloaded documentation:

- Click on Help -> View Help or 'Ctrl+F1'
- You can also use the 'QuickLaunch' box in Atmel Studio 7 at the right up corner:
 - Type 'SAM L21' in QuickLaunch search box
 - You can now select the desired documentation:



RESULT

You can now use the opened documentation

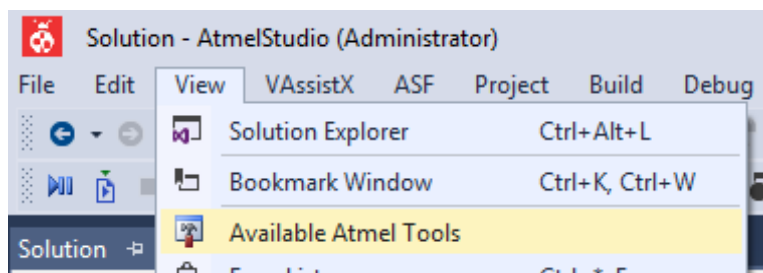
8. Appendix: Upgrade Embedded Debugger (EDBG) Firmware



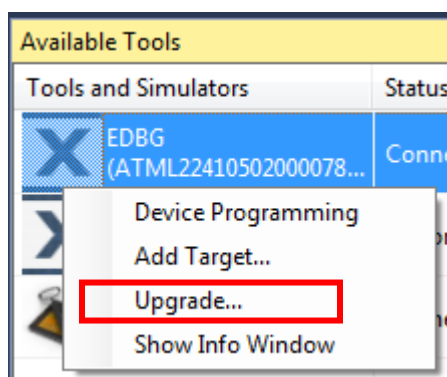
TO DO

Check & Upgrade EDBG Firmware Update

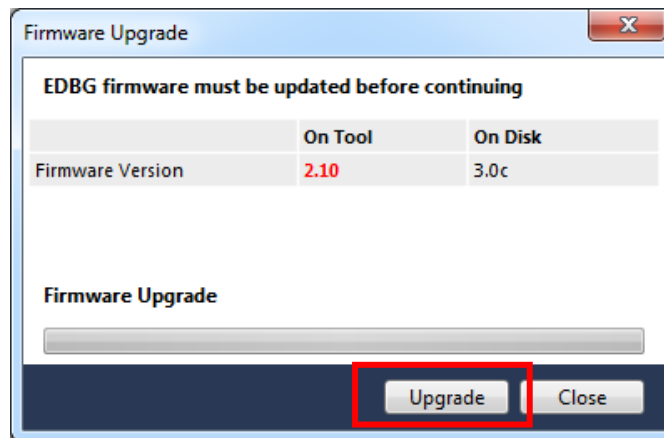
- Open Atmel Studio 7
- Connect your board to your computer using dedicated DEBUG USB connector
- Select View > Available Atmel Tools



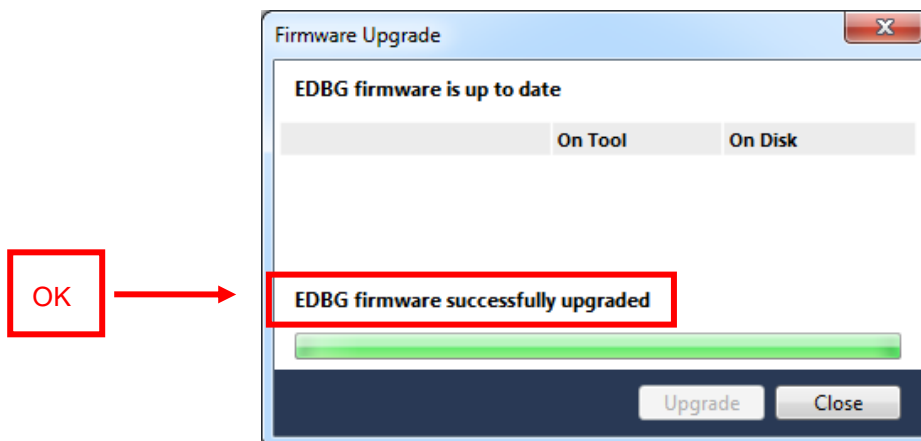
- Select your EDBG firmware and right click on it to select Upgrade...




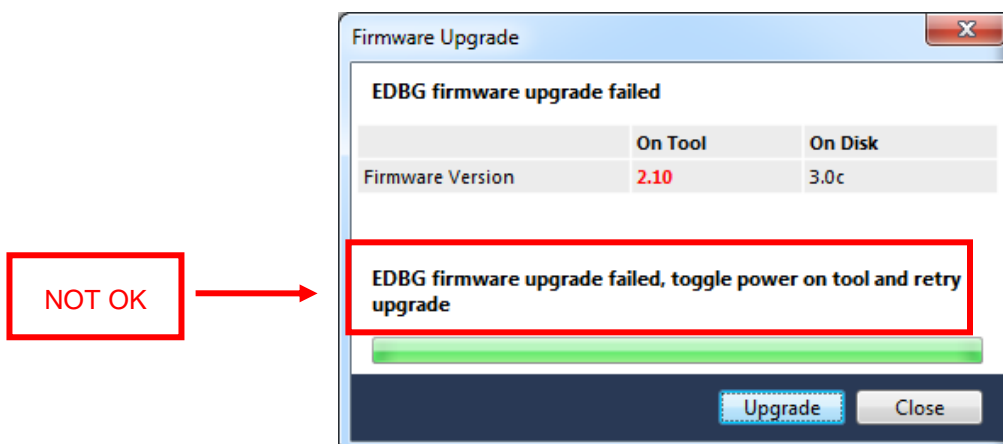
- If the EDBG firmware is NOT up to date, please click on Upgrade



 **RESULT** Your EDBG Firmware is upgraded



 **WARNING** Check you have the above message as it may happen a power down/up of the board may be required so that the process completes.



9. Revision History

Doc. Rev.	Date	Comments
-	06/2016	Updates to comply with latest Atmel Start improvements (June 2016 release)
-	03/2016	Minor Updates following first trainings / Appendix update (Help Viewer) Apply SAM L21 Erratum 15264 in both projects and documentation
-	02/2016	Initial document release



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

The Microchip name and logo are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

© 2016 Atmel Corporation.

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.