TRAINING MANUAL

Using Low Power Modes on the Atmel SAM4S-EK2

AN-8427

Prerequisites

- **Hardware Prerequisites**
  - Atmel® SAM4S-EK2 with ILI9325 LCD display
  - Atmel SAM-ICE™
  - Ammeter
- **Software Prerequisites**
  - IAR Embedded Workbench® 7.10
  - Latest J-Link / SAM-ICE Software and Documentation Pack
- **Estimated Completion Time:** 45 min

Introduction

The goal of this Hands-On is to:

- Become familiar with the SAM4S low-power modes and understand their main differences
- Measure the power consumption on the core power supply (VDDCORE) and compare it with the product datasheet

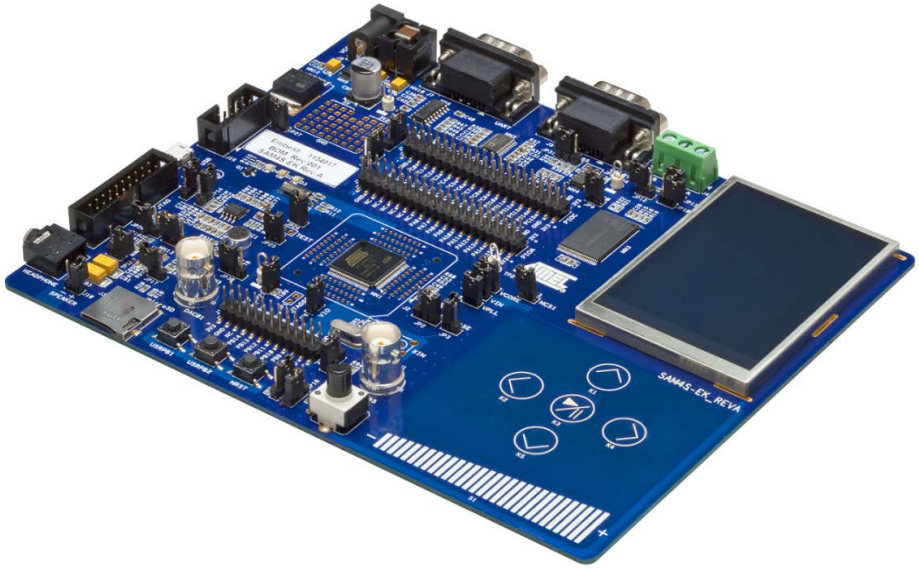


Table of Contents

Icon Key Identifiers

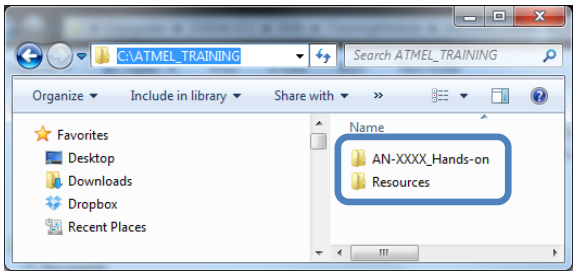
Icons are used to identify different assignment sections and reduce complexity. These icons are:

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. Training Module Architecture

Depending where the executable has been installed, you will find the following architecture which is composed by two main folders:

- `AN-XXXX_Hands-on`: contains the initial project that may be required to start and a solution
- **`Resources`**: contains required resources (datasheets, software, and tools…)

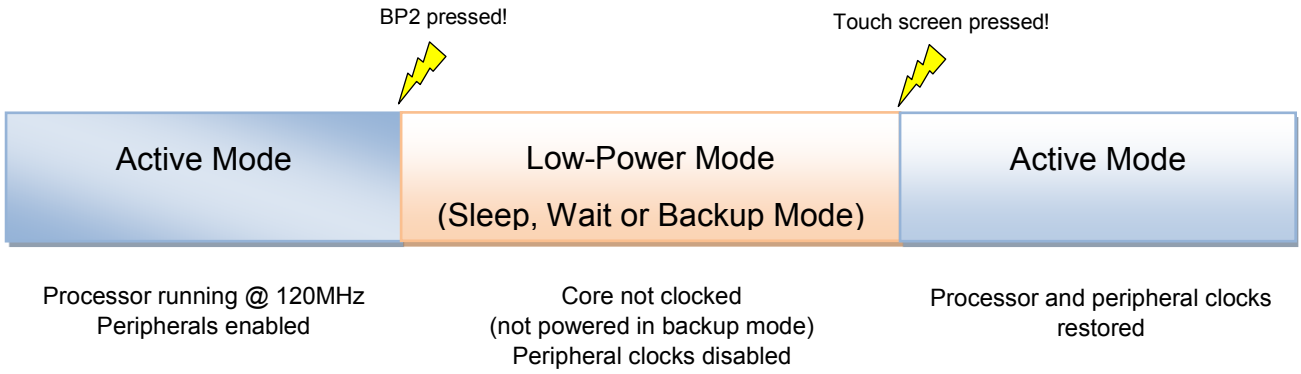


INFO Unless a specific location is specified, each time a reference is made to some resources in the following pages, the user must refer to this **`Resources`** folder.

2. Introduction

The goal of this hands-on is to develop the code used to configure the device entry in each low-power mode and measure the associated power consumption on VDDCORE.

The push button BP2 will be used to switch from active mode to low-power mode as described in the following figure:



Depending on the low-power mode used, pressing the touch screen will trigger an interrupt or an event.

- An interrupt will be used to wake up the device from sleep mode
- An event will be used to wake up the device from wait mode and backup mode

When the program starts, the device is running in active mode: the PLL is enabled and runs at 120MHz (MCK=120MHz).

In our program, the functions used to configure the SAM4S in Backup, Wait, and Sleep mode are respectively:

- `_EnterBackupMode()`
- `_EnterWaitMode()`
- `_EnterSleepMode()`

Before entering a low-power mode, the function:

- `_LowPower_Prepare()` is called to disable the peripheral clocks and configure the I/Os to minimize power consumption

When an event or an interrupt wakes up the device, the function:

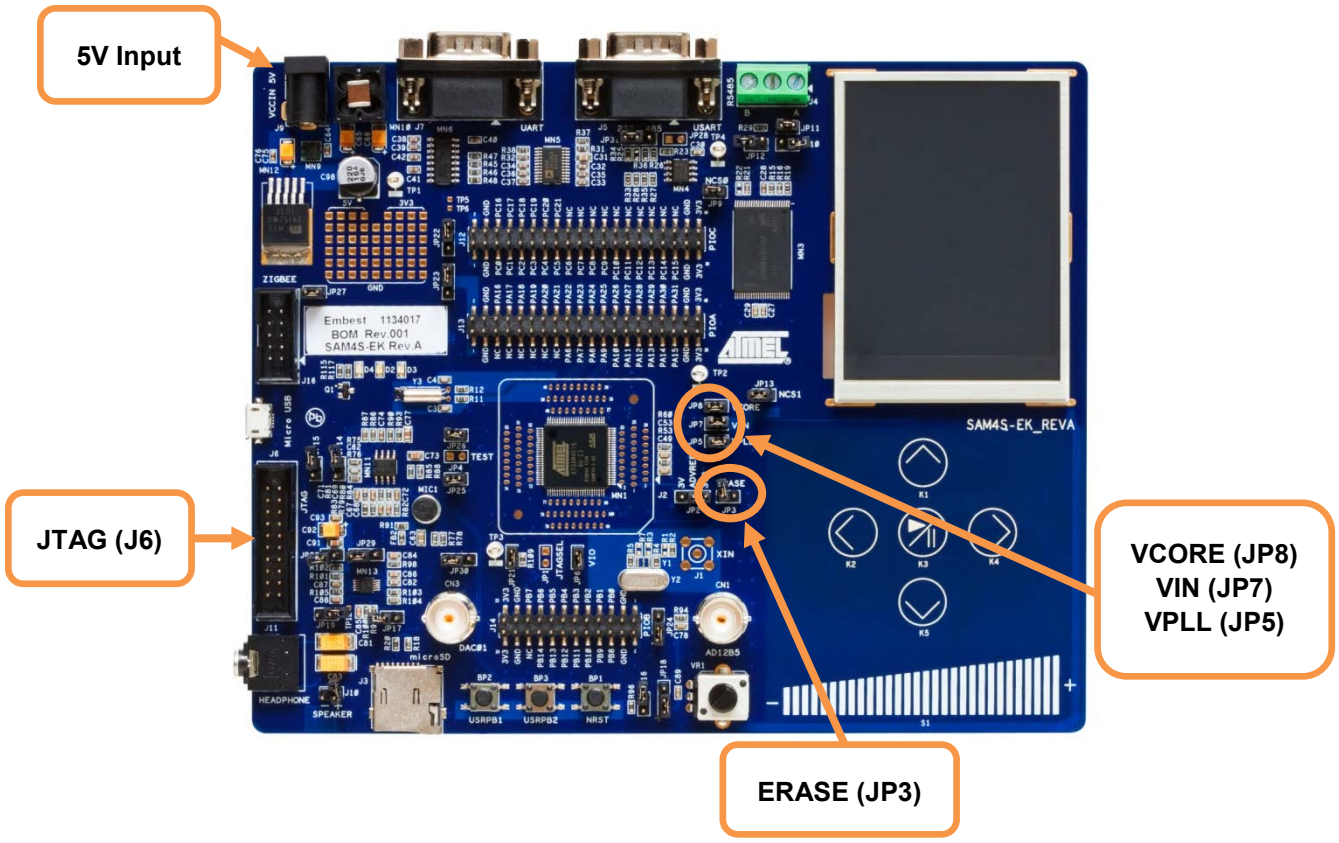
- `Exit_LP_Mode()` restores the clocks, I/Os, and peripherals configuration for Active Mode

You will work on these different functions in the following assignments.

The on board LCD will be used to display the instructions.

3. Setup

3.1 Hardware Setup



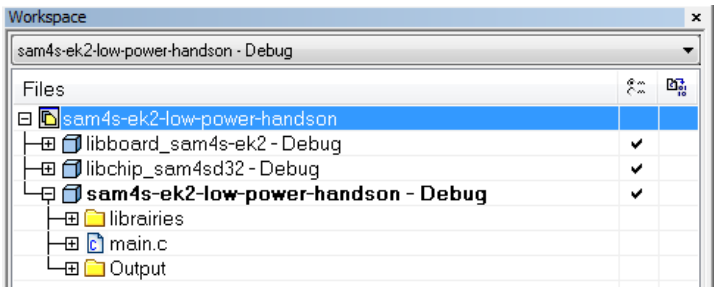
- Connect the Atmel SAM-ICE JTAG/ICE
- Make sure the ERASE (JP3) jumper is open
- Make sure the following jumpers are closed: VPLL, VIN, and VCORE
- Power-up the board by connecting the +5V DC power supply

3.2 Software Setup

- Launch IAR™ EWARM
- Drag and drop in IAR the `sam4s-ek2-low-power-handson.eww` workspace which is located in "`AN-4555_SAM4S-EK2_Using_Low_Power_Modes\assignments\`":

This IAR workspace is composed of three projects:

- **`sam4s-ek2-low-power-handson - Debug`**: this is the application
- **`libboard_sam4s-ek2 - Debug`**: this project generates the `libboard` library
- **`libchip_sam4sd32 - Debug`**: this project generates the `libchip` library



INFO Application Architecture.

In order to make new project development easier, each application is built on top of several libraries.

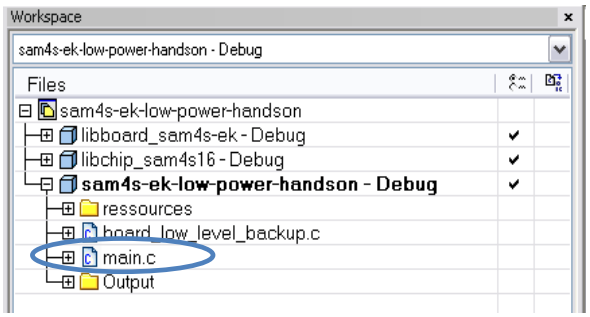
Here is a short overview of the main libraries available:

- **`libchip`** library: provides the API for the different embedded peripherals
- **`libboard`** library: provides the API for the different on-board components and board low-level initialization

These libraries can be compiled separately in the same IAR workspace by using their dedicated tabs:

TIPS `libchip` and `libboard` are libraries that will have to be compiled for each application. The other libraries are optional and their use depends on the application.

- Open the main.c file located in `sam4s-ek2-low-power-hands` on project



4. Assignment 1: Prepare the System for Low-Power Mode

In this first assignment, we will implement the `_LowPower_Prepare()` function, which will be called before entering any low-power mode.

This function will disable clocks and correctly configure I/Os to reduce power consumption.

As mentioned previously the touch screen IRQ pin (PA16) will be used as wake-up pin and so will be configured differently from other I/Os.

INFO All the functions to be used are located in both `main.c` and `pmc.c` files.

TO DO Implement `_LowPower_Prepare()` function in `main.c` file.

- Disable all the peripheral clocks by calling `PMC_DisableAllPeripherals` function **(TODO1.1)**

INFO `PMC_DisableAllPeripherals()` function can be found in `pmc.c` file which is located in `libchip_sam4sd32/source` folder.

- Disable the USB Device clock by setting `UDP` bit in the `PMC_SCDR` register **(TODO1.2)**

TIPS The structure of the `PMC_SCDR` register is shown below:

26.16.2 PMC System Clock Disable Register

Name: PMC_SCDR

Address: 0x400E0404

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	PCK2	PCK1	PCK0
7	6	5	4	3	2	1	0
UDP	–	–	–	–	–	–	–

- **UDP: USB Device Port Clock Disable**

0 = No effect.

1 = Disables the 48 MHz clock (UDPCK) of the USB Device Port.

- **PCKx: Programmable Clock x Output Disable**

0 = No effect.

1 = Disables the corresponding Programmable Clock output.

Configure all PIOs as general-purpose I/Os (GPIO) **except PA16** using `PIO_PER` registers: `PIOA->PIO_PER, PIOB->PIO_PER, PIOC->PIO_PER` (**TODO1.3**)

TIPS The structure of the `PIO_PER` register is shown below:

Name: PIO_PER

Addresses: 0x400E0E00 (PIOA), 0x400E1000 (PIOB), 0x400E1200 (PIOC)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: PIO Enable**

0 = No effect.

1 = Enables the PIO to control the corresponding pin (disables peripheral control of the pin).

- Configure all PIOs as input **except PA16** using `PIO_ODR` registers: `PIOA->PIO_ODR, PIOB->PIO_ODR, PIOC->PIO_ODR` (**TODO1.4**)

TIPS The structure of the PIO_ODR register is shown below:

Name: PIO_ODR

Addresses: 0x400E0E14 (PIOA), 0x400E1014 (PIOB), 0x400E1214 (PIOC)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: Output Disable**

0 = No effect.

1 = Disables the output on the I/O line.

- Disable all internal pull-ups **except PA16** one using `PIO_PUDR` registers: `PIOA->PIO_PUDR`, `PIOB->PIO_PUDR`, `PIOC->PIO_PUDR` (**TODO1.5**)

TIPS The structure of the `PIO_PUDR` register is shown below:

Name: PIO_PUDR

Addresses: 0x400E0E60 (PIOA), 0x400E1060 (PIOB), 0x400E1260 (PIOC)

Access: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: Pull Up Disable.**

0 = No effect.

1 = Disables the pull up resistor on the I/O line.

RESULT You have completed the implementation of the `_LowPower_Prepare()` function.

INFO An equivalent function `Exit_LP_Mode()` exists and is used to restore the active mode configuration once waken up from low power mode.

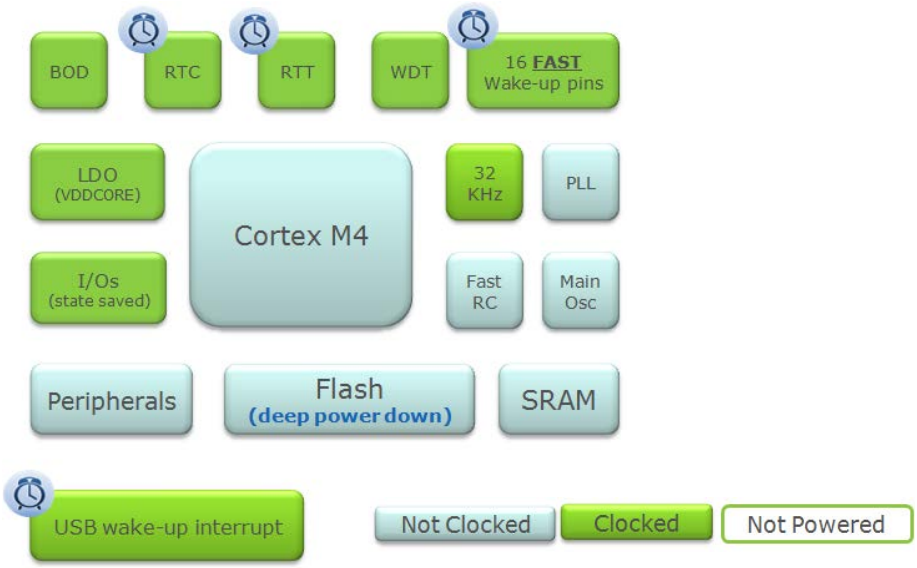
5. Assignment 2: Wait Mode Implementation

WAIT mode is used to achieve very low power consumption and a fast startup time.

In this mode, the core, peripherals, memories, and the oscillators/PLL (excluding the 32kHz oscillator) have their clocks stopped but are still powered.

A fast wake up is possible using any dedicated wake-up input pins (WKUPxx).

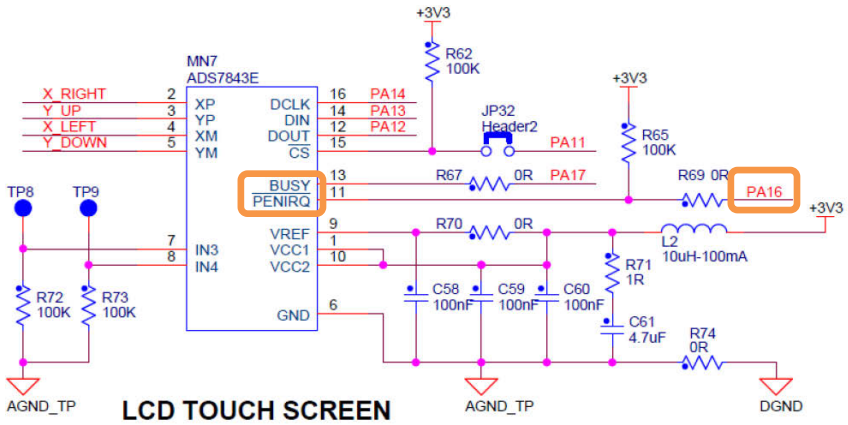
Other wake-up sources are also available for that mode and represented in the following figure:



The state of the I/Os is kept when entering this low power mode and unchanged once waken up.

INFO Concerning our application:

- Entering Wait Mode (`_EnterWaitMode()` function) is realized by pushing BP2 button
- The LDC Touch Screen Pen Detect pin (PA16) is used as a **fast startup** wake-up pin (WKUP15)



17	PA16	TK	TIOB1	PWML2	WKUP15	PIO_DCEN2		IRQ_TSC	ZigBEE IRQ0

Here are the steps to enter wait mode using the **WAITMODE** bit:

- Select the 4/8/12 MHz fast RC oscillator as Main Clock
- Configure the Wake Up pin to be used as Fast Startup pin
- Set FLPM field with the desired Flash Low Power Mode
- Set Flash wait state to 0
- Set WAITMODE bit
- Wait for Master Clock Ready MCKRDY = 1 in the PMC Status Register (PMC_SR)

TO DO Implement `_EnterWaitMode()` function.

- Switch MCK to the fast RC oscillator using `SwitchMck2FastRC(FAST_RC_OSC_4MHZ, PMC_MCKR_PRES_CLK_1)` function. (**TODO2.1**)

- Define WKUP15 input as fast startup input using `SetFastStartupInput(uint32_int inputs)` function, where `inputs` variable is WKUP15 pin (**TODO2.2**)

TIPS Use `PMC_FSMR_FSTT15` parameter to select WKUP15.

INFO `PMC_FSMR_FSTT15` is defined in `PMC.h` from `libchip_sam4sd32-debug` project in "`include/component`" directory.

Name: PMC_FSMR
Address: 0x400E0470
Access: Read-write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	LPM	–	USBAL	RTCAL	RTTAL
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
FSTT15	FSTT14	FSTT13	FSTT12	FSTT11	FSTT10	FSTT9	FSTT8
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FSTT7	FSTT6	FSTT5	FSTT4	FSTT3	FSTT2	FSTT1	FSTT0

- **FSTT0 - FSTT15: Fast Startup Input Enable 0 to 15**

0 = The corresponding wake up input has no effect on the Power Management Controller.

1 = The corresponding wake up input enables a fast restart signal to the Power Management Controller.

- Set FLPM bit in the PMC_FSMR register to put the Flash in Deep Power Down Mode which is the flash low power mode that will give the lowest flash power consumption in WAIT mode **(TODO2.3)**

TIPS The structure of the PMC_FSMR register (PMC->PMC_FSMR) is shown below:

Name: PMC_FSMR

Address: 0x400E0470

Access: Read-write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	FLPM		LPM	–	USBAL	RTCAL	RTTAL
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
FSTT15	FSTT14	FSTT13	FSTT12	FSTT11	FSTT10	FSTT9	FSTT8
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FSTT7	FSTT6	FSTT5	FSTT4	FSTT3	FSTT2	FSTT1	FSTT0

This register can only be written if the WPEN bit is cleared in "PMC Write Protect Mode Register" .

- **FSTT0 - FSTT15: Fast Startup Input Enable 0 to 15**

0 = The corresponding wake up input has no effect on the Power Management Controller.

1 = The corresponding wake up input enables a fast restart signal to the Power Management Controller.

- **RTTAL: RTT Alarm Enable**

0 = The RTT alarm has no effect on the Power Management Controller.

1 = The RTT alarm enables a fast restart signal to the Power Management Controller.

- **RTCAL: RTC Alarm Enable**

0 = The RTC alarm has no effect on the Power Management Controller.

1 = The RTC alarm enables a fast restart signal to the Power Management Controller.

- **USBAL: USB Alarm Enable**

0 = The USB alarm has no effect on the Power Management Controller.

1 = The USB alarm enables a fast restart signal to the Power Management Controller.

- **LPM: Low Power Mode**

0 = The WaitForInterrupt (WFI) or WaitForEvent (WFE) instruction of the processor makes the processor enter Sleep Mode.

1 = The WaitForEvent (WFE) instruction of the processor makes the system to enter in Wait Mode.

- **FLPM: Flash Low Power Mode**

Value	Name	Description
0	FLASH_STANDBY	Flash is in Standby mode when system enters wait mode
1	FLASH_DEEP_POWERDOWN	Flash is in deep power down mode when system enters wait mode
2	FLASH_IDLE	idle mode

- Set the flash Wait States (`FWS`) to 0 in the `EEFC_FMR` register **(TODO2.3)**

TIPS The structure of the EEFC_FMR register is shown below:

Name: EEFC_FMR

Address: 0x400E0A00

Access: Read-write

Offset: 0x00

31	30	29	28	27	26	25	24
–	–	–	–	–	CLOE	–	FAM
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	SCOD
15	14	13	12	11	10	9	8
–	–	–	–	FWS			
7	6	5	4	3	2	1	0
–		–	–	–	–	–	FRDY

- **FRDY: Ready Interrupt Enable**

0: Flash Ready does not generate an interrupt.

1: Flash Ready (to accept a new command) generates an interrupt.

- **FWS: Flash Wait State**

This field defines the number of wait states for read and write operations:

Number of cycles for Read/Write operations = FWS+1

- **SCOD: Sequential Code Optimization Disable**

0: The sequential code optimization is enabled.

1: The sequential code optimization is disabled.

No Flash read should be done during change of this register.

- **FAM: Flash Access Mode**

0: 128-bit access in read Mode only, to enhance access speed.

1: 64-bit access in read Mode only, to enhance power consumption.

No Flash read should be done during change of this register.

- **CLOE: Code Loops Optimization Enable**

0: The opcode loops optimization is disabled.

1: The opcode loops optimization is enabled.

No Flash read should be done during change of this register.

- Set `WAITMODE` bit in the `CKGR_MOR` register **(TODO2.3)**

WARNING The `CKGR_MOR register` is a key protected register, so the key “0x37” must be entered when modifying `WAITMODE` setting.

TIPS The structure of the CKGR_MOR register is shown below:

Name: CKGR_MOR

Address: 0x400E0420

Access: Read-write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	CFDEN	MOSCSEL
23	22	21	20	19	18	17	16
KEY							
15	14	13	12	11	10	9	8
MOSCXTST							
7	6	5	4	3	2	1	0
–	MOSCRCF			MOSCRCEN	WAITMODE	MOSCXTBY	MOSCXTEN

This register can only be written if the WPEN bit is cleared in “PMC Write Protect Mode Register” .

- **KEY: Password**

Should be written at value 0x37. Writing any other value in this field aborts the write operation.

- **MOSCXTEN: Main Crystal Oscillator Enable**

A crystal must be connected between XIN and XOUT.

0 = The Main Crystal Oscillator is disabled.

1 = The Main Crystal Oscillator is enabled. MOSCXTBY must be set to 0.

When MOSCXTEN is set, the MOSCXTS flag is set once the Main Crystal Oscillator startup time is achieved.

- **MOSCXTBY: Main Crystal Oscillator Bypass**

0 = No effect.

1 = The Main Crystal Oscillator is bypassed. MOSCXTEN must be set to 0. An external clock must be connected on XIN.

When MOSCXTBY is set, the MOSCXTS flag in PMC_SR is automatically set.

Clearing MOSCXTEN and MOSCXTBY bits allows resetting the MOSCXTS flag.

- **WAITMODE: Wait Mode Command**

0 = No effect.

1 = Enters the device in Wait Mode.

Note: The WAITMODE bit is write-only.

- **MOSCRCEN: Main On-Chip RC Oscillator Enable**

0 = The Main On-Chip RC Oscillator is disabled.

1 = The Main On-Chip RC Oscillator is enabled.

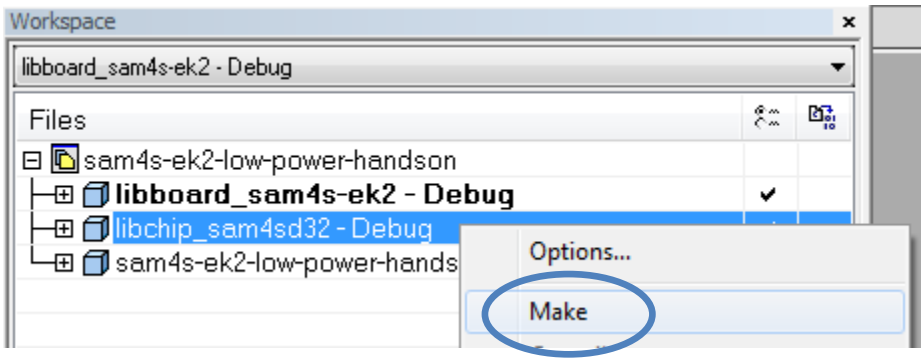
When MOSCRCEN is set, the MOSCRCS flag is set once the Main On-Chip RC Oscillator startup time is achieved.

RESULT The device is now able to enter Wait mode using `_EnterWaitMode` function.

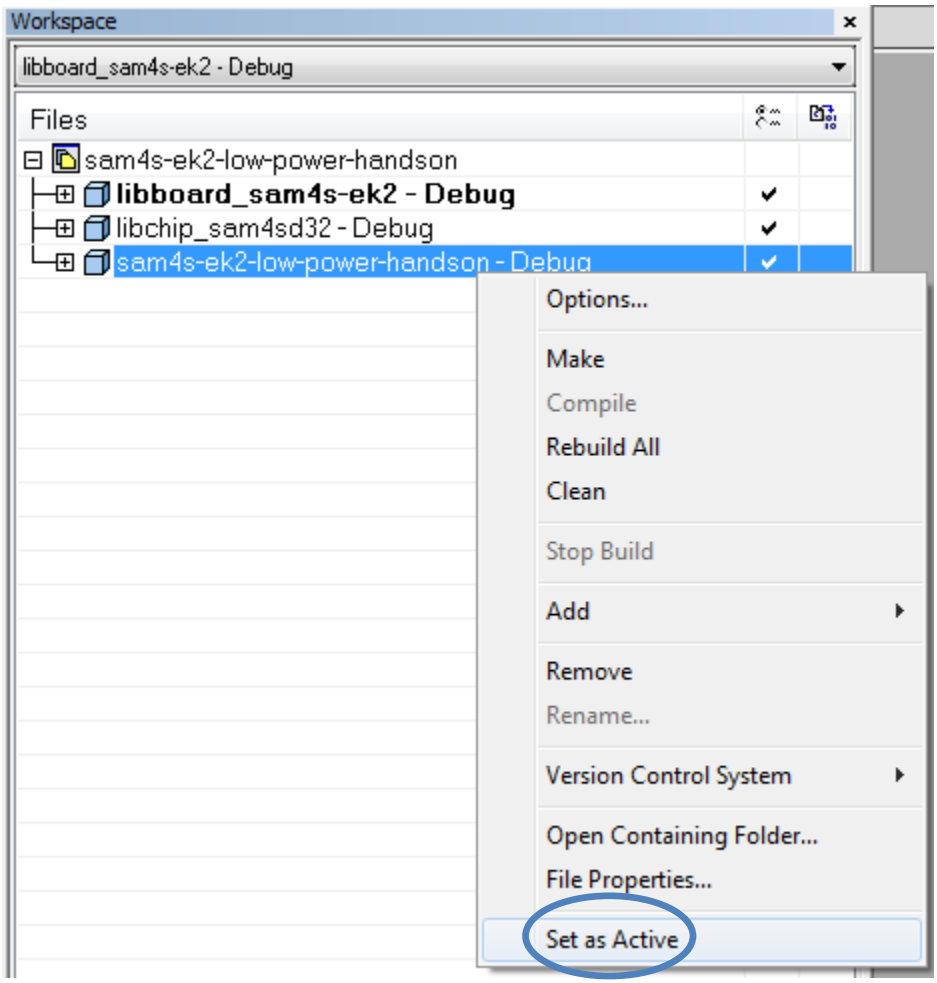
TO DO Compile and run the program.

- Check that the `lowPowerMode` global variable (defined in `main.c`) is set to `WAIT_MODE` in order to enter in the right low power mode during program execution

- Perform a build of `libchip` then `libboard` projects by right clicking on each of them in the workspace view and selecting “Make”:



- Set the application (`sam4s-ek2-low-power-handson`) as Active:



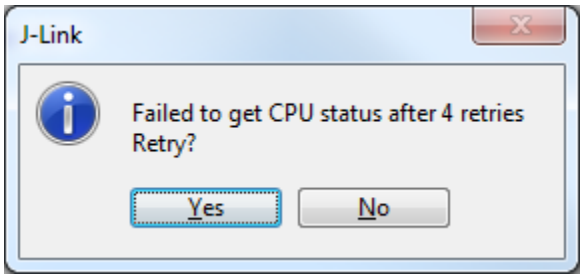
- Build and program the application in internal flash by clicking on the Download & Debug button:


WARNING It is not possible to program the Flash or enter in debug if the clock of the core is stopped. This happens if an application is in low power mode.

So, it may be necessary to perform an erase of the Flash: close jumper ERASE (JP3) when the microcontroller is powered, then open the jumper and reboot the board.

- Once the demo is programmed, launch it by clicking on the Go button: 
- Wait for the blue screen to appear on the SAM4S-EK LCD display
- As indicated on the LCD display press BP2 to switch in wait mode, then touch the LCD display to wake-up the device and come back in Active mode

INFO When the device is switched to low-power mode, the JTAG communication is lost and IAR should return a communication error which is a normal behavior as the clock of the core is then stopped:



TO DO Measure VDDCORE current consumption.

- Exit IAR Debug mode:
- Remove the +5V power supply to switch off the board
- Remove the JTAG probe
- Remove the jumper JP8 (VCORE) and plug the ammeter
- Select the ammeter current range to **200m**
- Power up the board
- Switch the SAM4S to wait mode by pushing BP2
- Change the ammeter current range to **200µ** to accurately measure the wait mode current

VDDCORE current consumption	Unit
	µA

- Compare the result with the value provided in the datasheet (SAM4S16/S8 case):

INFO In wait mode, VDDOUT consumption corresponds to VDDCORE, as both the main oscillator and the PLL (VDDPLL) are powered down.

Table 44-19. SAM4S16/S8 Typical Current Consumption in Wait Mode

Wait Mode Consumption	@25°C		@85°C	@105°C	
Conditions	VDDOUT Consumption (AMP1)	Total Consumption (AMP2)	Total Consumption (AMP2)	Total Consumption (AMP2	Unit
See Figure 44-9 on page 1116 There is no activity on the I/Os of the device. With the Flash in standby mode	20.5	32.7	344	654	µA
See Figure 44-9 on page 1116 There is no activity on the I/Os of the device. With the Flash in deep power down mode	20.5	27.8	438	589	µA

RESULT You have successfully implemented the entry in WAIT mode and have measured typical power consumption of our SAM4S on that low power mode.

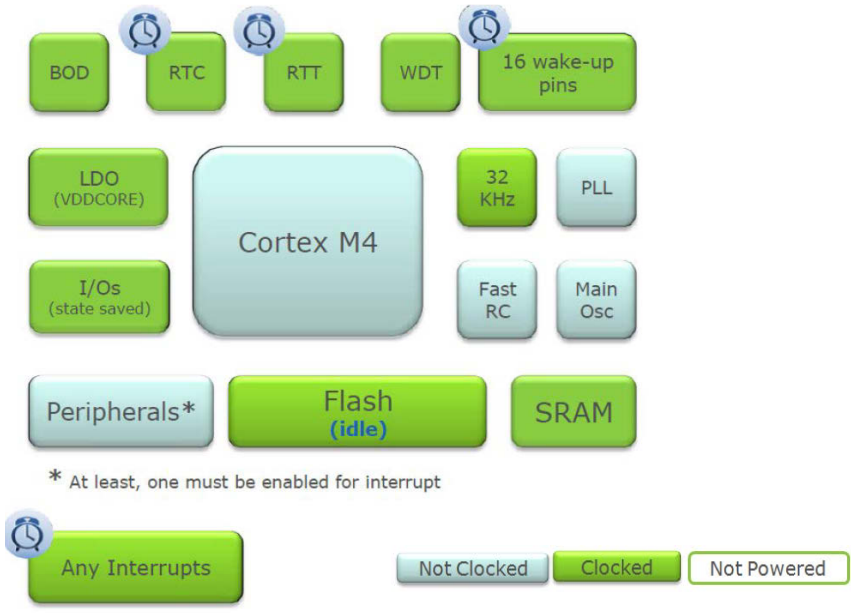
6. Assignment 3: Sleep Mode Implementation

The purpose of sleep mode is to optimize power consumption of the device versus response time with the use of interrupts to wake up the application.

In this mode, only the clock of the core is stopped. The peripheral clocks can be enabled or disabled separately as the different oscillators/PLL. So, current consumption in this mode is application-dependent.

The wake-up time is also clock-dependent and can go down to several us or can reach several milliseconds when the oscillator and PLL are disabled.

Wake-up sources available for that mode are represented in the following figure (case where all oscillators/PLL excluding the 32kHz are disabled):



The state of the I/Os is kept when entering this low power mode and unchanged once waken up.

INFO Concerning our application:

- Entering Sleep Mode (`_EnterSleepMode()` function) is realized by pushing BP2 button
- The system clock (MCK) will run at 1MHz from the Fast RC oscillator
- The LDC Touch Screen Pen Detect pin (PA16) is used as a wake-up **interrupt**

INFO Contrary to Wait Mode, the wake-up source cannot be programmed as a fast startup wake up pins. Fast startup mechanism is specific to the Wait mode. In Sleep mode case, we use the Touch Screen Pen Detect as a standard interrupt.

Here are the steps to enter Sleep mode using WFI Cortex®-M4 instruction:

- Select the 4/8/12 MHz fast RC oscillator as Main Clock and run the system at 1MHz
- Clear LPM bit
- Clear SLEEPDEEP bit
- Execute WFI Instruction

TO DO Implement `_EnterSleepMode()` function.

- Switch MCK to the fast 4MHz RC oscillator using `SwitchMck2FastRC(FAST_RC_OSC_4MHZ, PMC_MCKR_PRES_CLK_4)` function. **(TODO3.1)**

- Clear `LPM` bit in the PMC_FSMR register (`LPM = 0`) **(TODO3.2)**

Name: PMC_FSMR
Address: 0x400E0470
Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	FLPM		LPM	–	USBAL	RTCAL	RTTAL
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
FSTT15	FSTT14	FSTT13	FSTT12	FSTT11	FSTT10	FSTT9	FSTT8
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FSTT7	FSTT6	FSTT5	FSTT4	FSTT3	FSTT2	FSTT1	FSTT0

- **LPM: Low-power Mode**

0: The WaitForInterrupt (WFI) or the WaitForEvent (WFE) instruction of the processor makes the processor enter Sleep Mode.

1: The WaitForEvent (WFE) instruction of the processor makes the system to enter in Wait Mode.

- Clear `SLEEPDEEP` bit in the SCR register (`SLEEPDEEP = 0`) **(TODO3.2)**

Name: SCB_SCR
Access: Read/Write
Reset: 0x000000000

31	30	29	28	27	26	25	24
–							

23	22	21	20	19	18	17	16
–							

15	14	13	12	11	10	9	8
–							

7	6	5	4	3	2	1	0
–			SEVONPEND	–	SLEEPDEEP	SLEEPONEXIT	–

- **SEVONPEND: Send Event on Pending Bit**

0: Only enabled interrupts or events can wake up the processor; disabled interrupts are excluded.

1: Enabled events and all interrupts, including disabled interrupts, can wake up the processor.

When an event or an interrupt enters the pending state, the event signal wakes up the processor from WFE. If the processor is not waiting for an event, the event is registered and affects the next WFE.

The processor also wakes up on execution of an SEV instruction or an external event.

- **SLEEPDEEP: Sleep or Deep Sleep**

Controls whether the processor uses sleep or deep sleep as its low power mode:

0: Sleep.

1: Deep sleep.

- **SLEEPONEXIT: Sleep-on-exit**

Indicates sleep-on-exit when returning from the Handler mode to the Thread mode:

0: Do not sleep when returning to Thread mode.

1: Enter sleep, or deep sleep, on return from an ISR.

Setting this bit to 1 enables an interrupt-driven application to avoid returning to an empty main application.

- Add `__WFI()` function (which executes the `WFI` assembly instruction) **(TODO3.2)**

RESULT The device is now able to enter Sleep mode using `_EnterSleepMode` function.

TO DO Compile and run the program.

- Power down the board
- Re-connect the JTAG probe and JP8
- Power up the board
- Check that the `lowPowerMode` global variable (defined in `main.c`) is set to `SLEEP_MODE` in order to enter in the right low power mode
- Compile and program the demo in internal flash by clicking on the Download & Debug button: 
- Exit IAR Debug mode 

TO DO Measure the power consumption of the board.

- Remove the +5V power supply to switch off the board
- Remove the JTAG probe
- Remove jumper JP8 and plug in the ammeter
- Select the ammeter current range to **200m**
- Power up the board
- Switch the SAM4S to sleep mode by pushing BP2
- Change the ammeter current range to **20m** to measure the sleep mode current accurately

Power consumption	Unit
	µA

- Compare the result with the value provided in the datasheet (SAM4SD32 case):

Table 44-17. SAM4SD32/SD16/SA16 Typical Sleep Mode Current Consumption versus Master Clock (MCK) Variation with FAST RC

Sleep Mode Consumption	Typical Value @25°C		
Core Clock/MCK (MHz)	VDDCORE Consumption (AMP1)	Total Consumption (AMP2)	Unit
12	1.1	1.8	mA
8	0.8	1.2	mA
4	0.4	0.7	mA
2	0.3	0.7	mA
1	0.2	0.5	mA
0.5	0.2	0.5	mA

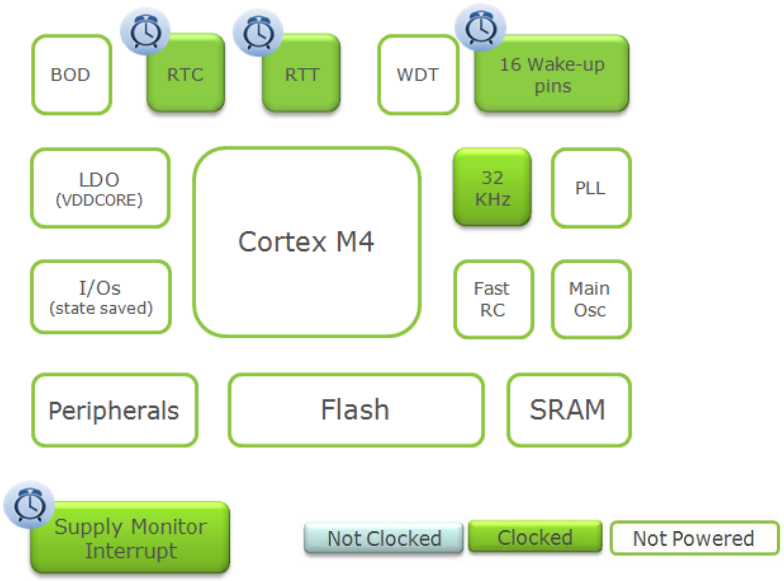
RESULT You have successfully implemented the entry in SLEEP mode and have measured typical power consumption of our SAM4S on that low power mode.

7. Assignment 4: Backup Mode Implementation

The purpose of backup mode is to achieve the lowest possible power consumption while performing periodic wake-up to carry out tasks that do not require very fast startup time.

Only the Supply Controller, power power-on reset, RTT, RTC, backup registers, and 32kHz oscillator are running.

The on-chip voltage regulator (LDO: **L**inear **D**C **O**utput) and so the core power supply (VDDCORE) are powered down.



After wake-up, the core is reset and the IOs are configured as input with internal pull-up enabled.

INFO Concerning our application:

- Entering Sleep Mode (`_EnterBackupMode()` function) is realized by pushing BP2 button
- The system clock (MCK) will run at 1MHz from the Fast RC oscillator
- The LDC Touch Screen Pen Detect pin (PA16) is used as a wake-up pin (WKUP15)

Here are the steps to enter Backup mode using **VROFF** bit:

- Select the Wake Up pin to be used to wake up the core power supply (VDDCORE)
- Set VROFF bit

TO DO Implement `_EnterBackupMode()` function.

- Configure the push button PA16/WKUP15 pin as the wake-up input using `SUPC_WUIR` register. This wake-up event has to be detected on a high-to-low transition. **(TODO4.1)**

TIPS The structure of the `SUPC_WUIR` register is shown below:

Name: SUPC_WUIR

Address: 0x400E1420

Access: Read-write

31	30	29	28	27	26	25	24
WKUPT15	WKUPT14	WKUPT13	WKUPT12	WKUPT11	WKUPT10	WKUPT9	WKUPT8
23	22	21	20	19	18	17	16
WKUPT7	WKUPT6	WKUPT5	WKUPT4	WKUPT3	WKUPT2	WKUPT1	WKUPT0
15	14	13	12	11	10	9	8
WKUPEN15	WKUPEN14	WKUPEN13	WKUPEN12	WKUPEN11	WKUPEN10	WKUPEN9	WKUPEN8
7	6	5	4	3	2	1	0
WKUPEN7	WKUPEN6	WKUPEN5	WKUPEN4	WKUPEN3	WKUPEN2	WKUPEN1	WKUPEN0

- **WKUPEN0 - WKUPEN15: Wake Up Input Enable 0 to 15**

0 (NOT_ENABLE) = the corresponding wake-up input has no wake up effect.

1 (ENABLE) = the corresponding wake-up input forces the wake up of the core power supply.

- **WKUPT0 - WKUPT15: Wake Up Input Transition 0 to 15**

0 (HIGH_TO_LOW) = a high to low level transition on the corresponding wake-up input forces the wake up of the core power supply.

1 (LOW_TO_HIGH) = a low to high level transition on the corresponding wake-up input forces the wake up of the core power supply.

- Set `VROFF` bit in the `SUPC_CR` register **(TODO4.2)**

WARNING The `SUPC_CR` is a key protected register, so the key "**0xA5**" must be entered when modifying VROFF setting.

TIPS The structure of the SUPC_CR register is shown below:

Name: SUPC_CR

Address: 0x400E1410

Access: Write-only

31	30	29	28	27	26	25	24
KEY							
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–		–
7	6	5	4	3	2	1	0
–	–	–	–	XTALSEL	VROFF	–	–

- **VROFF: Voltage Regulator Off**

0 (NO_EFFECT) = no effect.

1 (STOP_VREG) = if KEY is correct, asserts vddcore_nreset and stops the voltage regulator.

- **XTALSEL: Crystal Oscillator Select**

0 (NO_EFFECT) = no effect.

1 (CRYSTAL_SEL) = if KEY is correct, switches the slow clock on the crystal oscillator output.

- **KEY: Password**

Should be written to value 0xA5. Writing any other value in this field aborts the write operation.

RESULT The device is now able to enter Wait mode using `_EnterBackupMode` function.

TO DO Compile and run the program.

- Power down the board
- Re-connect the JTAG probe and JP8
- Power up the board
- Check that the `lowPowerMode` global variable (defined in `main.c`) is set to `BACKUP_MODE` in order to enter in the right low power mode
- Compile and program the demo in internal flash by clicking on the Download & Debug button: 
- Exit IAR Debug mode 

TO DO Measure the power consumption of the board.

- Remove the +5V power supply to switch off the board
- Remove the JTAG probe
- Remove jumper JP8 and plug in the ammeter
- Select the ammeter current range to **200m**
- Power up the board
- Switch the SAM4S to backup mode by pushing BP2
- Change the ammeter to the lowest range

INFO Depending on your ammeter accuracy, you may not observe any current consumption which is normal as we are in the 1µA range for that mode:

Table 44-10. Typical Power Consumption for Backup Mode (SAM4SD32/SD16/SA16 rev A)

Backup Total Consumption	@25°C		@85°C	@105°C	Unit
Conditions	(AMP1) Configuration A	(AMP1) Configuration B	(AMP1) Configuration A	(AMP1) Configuration A	Unit
VDDIO = 3.6V	2.1	2.0	13.9	22.5	µA
VDDIO = 3.3V	1.8	1.8	NA	NA	
VDDIO = 3.0V	1.7	1.6	NA	NA	
VDDIO = 2.5V	1.3	1.3	NA	NA	
VDDIO = 1.8V	0.9	0.9	NA	NA	

RESULT You have successfully implemented the entry in BACKUP mode and have observed typical power consumption of our SAM4S on that low power mode.

8. Conclusion

In this hands-on, you were able to configure the different SAM4S low-power modes and measure their typical current consumption on VDDCORE.

You have been able to confront these measures with the values provided in the SAM4S product datasheet.

9. Revision History

Doc. Rev.	Date	Comments
42272A	03/2014	Initial document release

Atmel | Enabling Unlimited Possibilities®

Atmel Corporation
1600 Technology Drive
San Jose, CA 95110
USA
Tel: (+1)(408) 441-0311
Fax: (+1)(408) 487-2600
www.atmel.com

Atmel Asia Limited
Unit 01-5 & 16, 19F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
HONG KONG
Tel: (+852) 2245-6100
Fax: (+852) 2722-1369

Atmel Munich GmbH
Business Campus
Parkring 4
D-85748 Garching b. Munich
GERMANY
Tel: (+49) 89-31970-0
Fax: (+49) 89-3194621

Atmel Japan G.K.
16F Shin-Osaki Kangyo Bldg.
1-6-4 Osaki, Shinagawa-ku
Tokyo 141-0032
JAPAN
Tel: (+81)(3) 6417-0300
Fax: (+81)(3) 6417-0370