APPLICATION NOTE

# Atmel AT03258: Using Low Power Modes in SAM4E Microcontroller

## Atmel 32-bit Microcontroller

## Features

- Low power modes in SAM4E
- Power supply in SAM4E

## Introduction

The purpose of this application note is to explain how to use low power modes in the Atmel SAM4E. The application note gives a brief description of low power modes in SAM4E and system controller for reducing power consumption. It includes how to enter and exit each low power mode, also give some suggestions on using these low power modes in application.

## Table of Contents

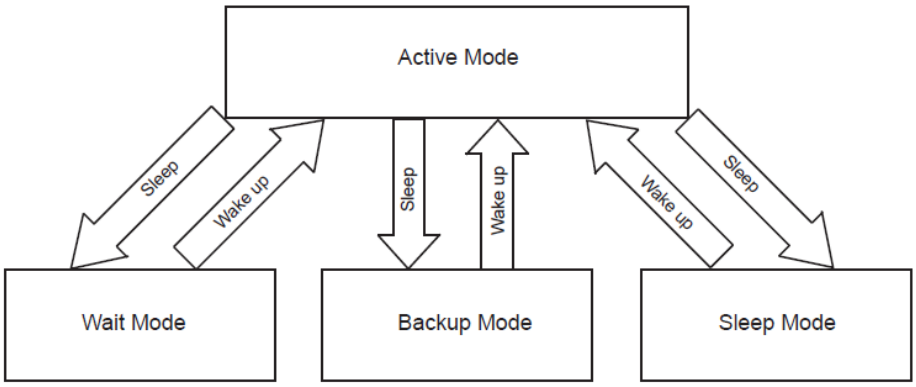
# 1. Low Power Modes in SAM4E

SAM4E microcontroller family has various low power modes, including sleep mode, wait mode and backup mode.

Various low power modes are shown in Figure 1-1. Each low power mode can be entered from active mode and wake up by related enabled wake-up event. The various low power modes will be described below in detail.

The Atmel Software Framework (ASF) provides API to implement low power mode functions.

**Figure 1-1. Various Low Power Modes**




# 2. Power Supply

For reducing power consumption, the system can use supply controller (SUPC) to control the supply voltage and use power management controller (PMC) to control the clock of the system.

## 2.1 Supply Controller (SUPC)

The SUPC controls the supply voltage of the system and manages the backup mode by controlling the embedded voltage regulator.

The SUPC starts up the device by enabling the voltage regulator. Then it generates the proper reset signals to the core power supply.

The SUPC integrates the Slow Clock generator which is based on a 32 kHz crystal oscillator and an embedded 32 kHz RC oscillator. The Slow Clock defaults to the RC oscillator, but the software can enable the crystal oscillator and select it as the Slow Clock source.

The SUPC embeds a supply monitor which is located in the VDDIO power supply and which monitors VDDIO power supply. The supply monitor can be used to prevent the processor from falling into an unpredictable state if the main power supply drops below a certain level.

Figure 2-1 gives an overview of the Supply Controller of SAM4E.

There are several types of power supply pins:

VDDCORE: powers the core, the first flash rail, the embedded memories and the peripherals

VDDIO: powers the peripherals I/O lines (Input/Output Buffers), USB transceiver, the second flash rail, the backup part, 32 kHz crystal oscillator and oscillator pads

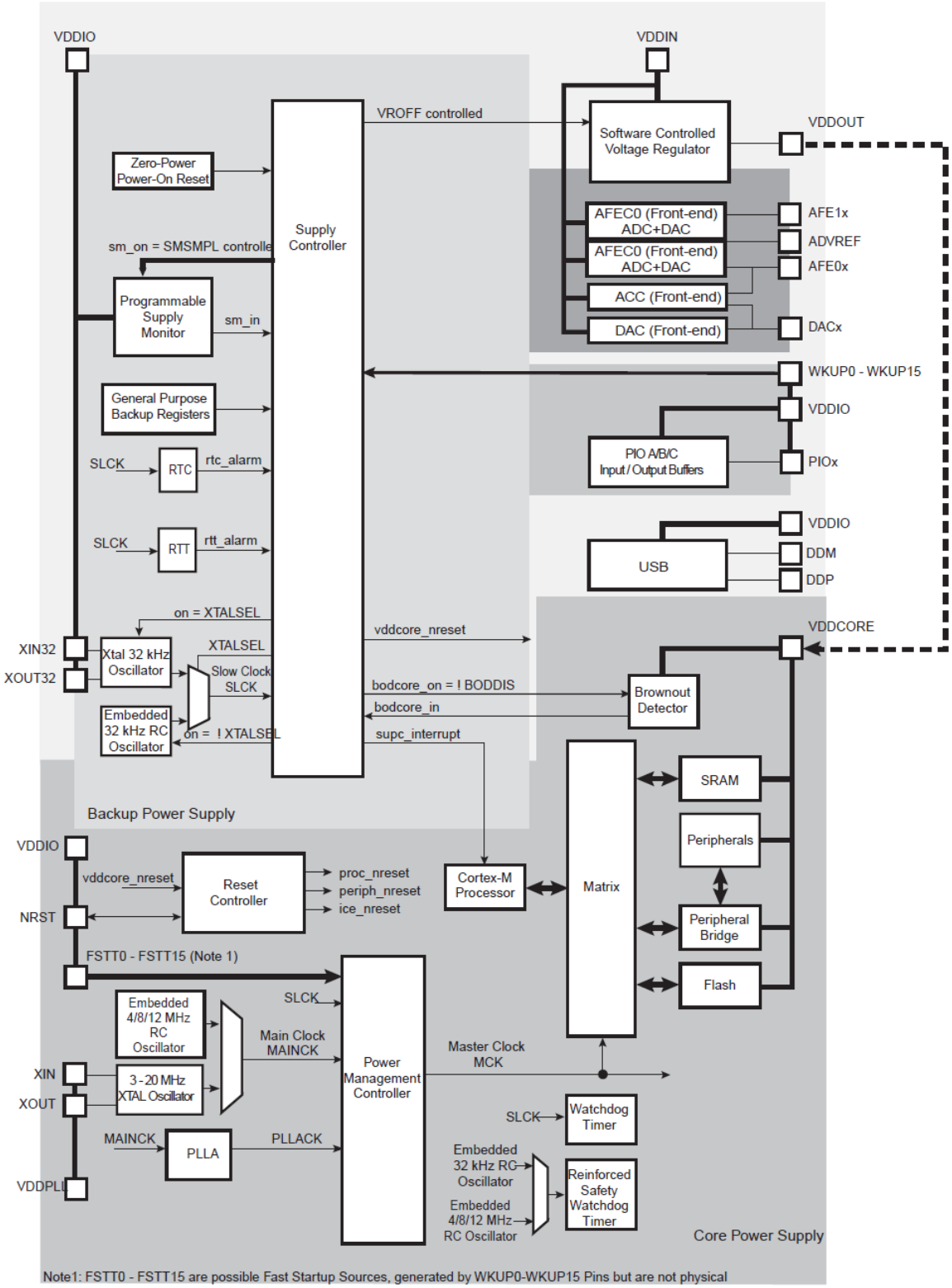
VDDIN: voltage regulator input, ADC, DAC and analog comparator power supply

VDDOUT: the output of the voltage regulator, intends to supply the core of the device

VDDPLL: powers the PLL, the Fast RC and the 3 to 20 MHz oscillator

The device can be divided into two power supply areas as shown in Figure 2-1: Backup Power Supply and Core Power Supply.

**Figure 2-1. Supply Controller of SAM4E**



## 2.2 General Purpose Backup Registers (GPBR)

The device embeds twenty 32-bit general-purpose backup registers which can store 80 bytes of user application data. When system enters backup mode, the voltage regulator is disabled, and VDDCORE is off but the backup registers remain powered by VDDIO. They retain the content when the system wakes up from backup mode.

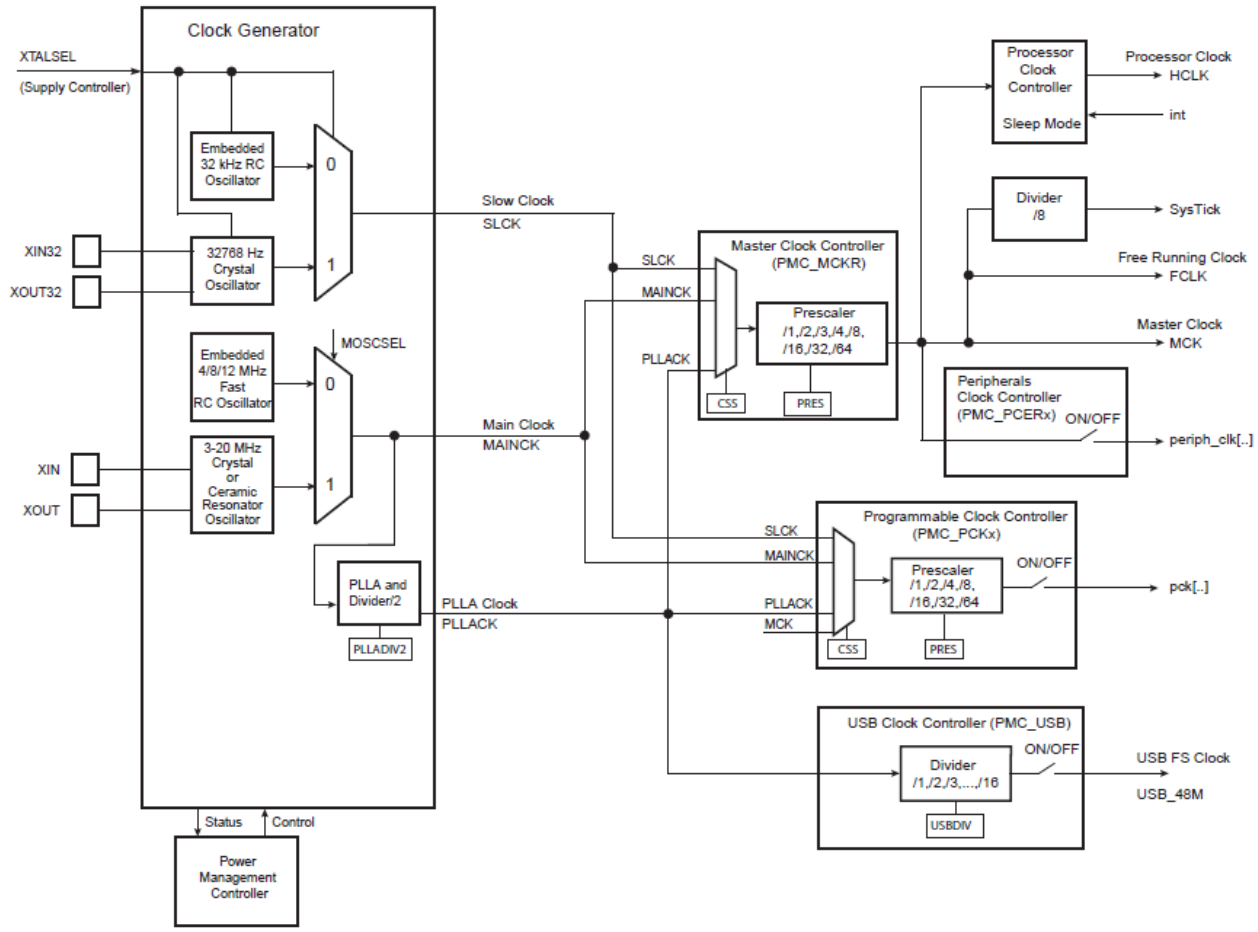
It is possible to generate an immediate clear of the content of general purpose backup registers 0 to 9 (first half), if a low power debounce event is detected on a wakeup pin, WKUP0 or WKUP1. The content of the other general-purpose backup registers (second half) remains unchanged.

If a tamper event has been detected, it is not possible to write into general purpose backup registers while the LPDBCS0 or LPDBCS1 flags are not cleared in supply controller status register (SUPC_SR).

## 2.3 Power Management Controller (PMC)

The PMC controls all system and peripheral clocks as shown in Figure 2.2. It can be used to adapt the system frequency and to enable/disable the peripheral clocks. It acts an important role for optimizing power consumption.

**Figure 2-2. General Clock Block Diagram of SAM4E**



### 2.3.2 Clock Sources

Normally, the system does not need to run at device's maximum frequency. According to requirement of system performance, proper Master Clock (MCK) needs to be found for optimizing power consumption. The user can choose one of the following clocks as master clock source from clock generator.

- Slow clock
- Main clock
- PLL clock

#### 2.3.2.1 Slow Clock

The Slow Clock is generated either by the Slow Clock Crystal Oscillator or by the Slow Clock RC Oscillator.

The selection between the RC and the crystal oscillator is made by writing the XTALSEL bit in the SUPC Control Register (SUPC_CR).

#### 2.3.2.2 Main Clock

The Main Clock has two sources:

- 4/8/12 MHz Fast RC Oscillator, which starts very quickly and is used at startup.
- 3 to 20 MHz Crystal or Ceramic Resonator-based Oscillator, which can be bypassed.

The user can select the source of Main Clock by writing the MOSCSEL bit in the Clock Generator Main Oscillator Register (CKGR_MOR).

The software can enable or disable the 4/8/12 MHz Fast RC Oscillator with the MOSCRCEN bit in the CKGR_MOR. The user can also select the output frequency of the Fast RC Oscillator through MOSCRCF bits in CKGR_MOR, either 4 MHz, 8 MHz or 12 MHz are available.

The software can enables or disable the 3 to 20 MHz Crystal or Ceramic Resonator-based oscillator with the MOSCXTEN bit in the CKGR_MOR. The Main Crystal Oscillator can be bypassed by setting MOSCXTBY and an external clock must be connected on XIN.

#### 2.3.2.3 PLL Clock

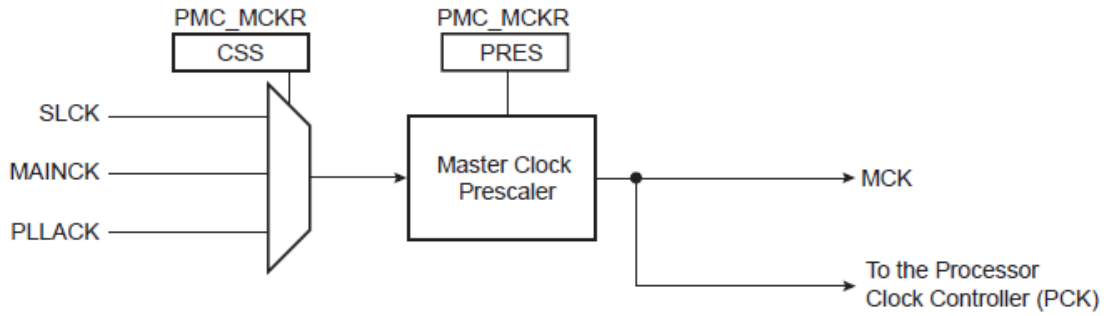
The PLL Clock signal has a frequency that depends on the Main Clock frequency and on the parameters DIV and MUL. The factor applied to the source signal frequency is (MUL + 1) / DIV. For example, the Main Clock frequency is 12MHz, and MUL = 19, DIV = 1, then the PLL Clock frequency is 12 * (19 + 1) / 1 = 240MHz.

### 2.3.3 Clock Configurations

#### 2.3.3.1 Master Clock

The Master Clock Controller can configure the master clock (MCK) as shown in Figure 2-3.

**Figure 2-3. Master Clock Controller**



The Master Clock is selected from one of the Clock Source: Slow Clock, Main Clock and PLL Clock.

The Master Clock selection is made by writing the Clock Source Selection field (CSS) in Master Clock Register (PMC_MCKR). The prescaler supports the division by a power of 2 of the selected clock between 1 and 64, and the division by 3. The PRES field in PMC_MCKR programs the prescaler.

#### 2.3.3.2 Processor Clock

The Processor Clock can be disabled by executing the Wait For Interrupt (WFI) processor instruction.

The Processor Clock HCLK is enabled after a reset and is automatically re-enabled by any enabled interrupt.

The Processor Sleep Mode is achieved by disabling the Processor Clock, which is automatically re-enabled by any enabled fast or normal interrupt, or by the reset of the product.

#### 2.3.3.3 Peripheral Clock

The peripheral clocks are automatically disabled after a reset. The user can individually enable and disable the clock on the peripherals by writing into the Peripheral Clock Enable 0 (PMC_PCER0), Peripheral Clock Enable 1 (PMC_PCER1) and Peripheral Clock Disable 0 (PMC_PCDR0), Peripheral Clock Disable 1 (PMC_PCDR1) registers. In order to reduce power consumption, it is recommended to disable the clock of unused peripherals.

#### 2.3.3.4 Programmable Clock Output

The PMC controls 3 signals to be output on external pins, PCKx. Each signal can be independently programmed via the Programmable Clock Registers (PMC_PCKx).

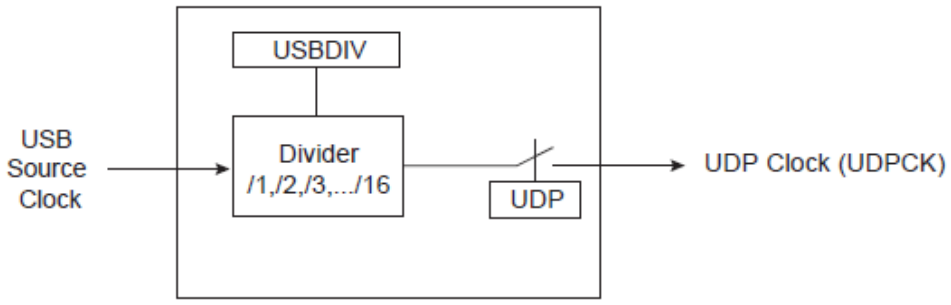
PCKx can be independently selected between the Slow Clock (SLCK), the Main Clock (MAINCK), the PLLA Clock (PLLACK), and the Master Clock (MCK) by writing the CSS field in PMC_PCKx. Each output signal can also be divided by a power of 2 between 1 and 64 by writing the PRES (Prescaler) field in PMC_PCKx.

Each output signal can be enabled or disabled by writing 1 in the corresponding bit, PCKx of PMC_SCER and PMC_SCDR, respectively.

#### 2.3.3.5 USB Clock

The USB Clock Controller can configure the USB clock as shown in Figure 2-4.

**Figure 2-4. USB Clock Controller**




The PLL is the USB source clock, so the user must program the PLL to generate an appropriate frequency depending on the USBDIV bit in PMC_USB.

The USB device clock can be enabled by setting the UDP bit in PMC_SCER. To save power on this peripheral when it is not used, the user can set the UDP bit in PMC_SCDR.

# 3. Sleep Mode

In Sleep mode, the execution of instructions is suspended because the core clock is stopped while the peripheral clocks can be enabled. The purpose of sleep mode is to optimize power consumption of the device versus response time. The core needs lowest wakeup time in this mode.

## 3.1 Enter Sleep Mode

Sleep mode is entered via Wait For Interrupt (WFI) instructions. This mode is achieved by disabling the core clock using PMC. Following API could be used to enter sleep mode in ASF:

```
pmc_enable_sleepmode()
```

When Sleep mode is entered, the current instruction is finished before the clock is stopped, but this does not prevent data transfers from other masters of the system bus.

## 3.2 Exit Sleep Mode

Sleep mode is exited from an interrupt if WFI instruction is used to enter this mode, or by the reset of the device. The PMC automatically re-enables the processor clock and the core executes the next instruction of the program counter.

## 3.3 Using Sleep Mode

In most applications, there are some polling loops for waiting for operation finish. For example, you need to send a large amount of data via USART, the pseudo code might like this:

```
SendDataViaUSART()
{
    Begin to send data via USART by PDC;
    Wait for transfer finish;
    Continue to execute next operation;
}
```

There are two ways for waiting for transfer finish:

- Polling loops, like while (transfer not finish) {1}
- Enter sleep mode and set the core be woken up by an interrupt when transfer finish.

By using polling loops, the instructions are still executed thereby increasing power consumption.

By using Sleep mode, the core clock is stopped thereby reducing power consumption while USART peripheral clock is still working, so data transfer continues. When the transfer finishes, an interrupt will wake up the core immediately.

So in the case of the core is not running while waiting for peripherals transfer data done (with DMA or PDC), it is recommended to use the Sleep mode instead of polling loops.

# 4. Wait Mode

The purpose of the wait mode is to achieve very low power consumption while maintaining the whole device in a powered state for a startup time of less than few hundred µs. Current Consumption in wait mode is typically few µA (total current consumption) if the internal voltage regulator is used.

In this mode, the clocks of the core, peripherals and memories are stopped. However, the core, peripherals and memories power supplies are still powered. From this mode, a fast start-up is available.

## 4.1 Enter Wait Mode

Wait mode is entered via WAITMODE =1 (wait mode bit in CKGR_MOR). The Fast RC Oscillator is disabled by the PMC automatically when entering wait mode. The step of entering Wait mode as below:

- Select the 4/8/12 MHz fast RC oscillator as Main Clock
- Set the FLPM bit field in the PMC Fast Startup Mode Register (PMC_FSMR)
- Set Flash Wait State at 0.
- Set the WAITMODE bit = 1 in the PMC Main Oscillator Register (CKGR_MOR)
- Wait for Master Clock Ready MCKRDY=1 in the PMC Status Register (PMC_SR)

Note: Internal main clock resynchronization cycles are necessary between the writing of MOSCRCEN bit and the effective entry in wait mode. Depending on the user application, waiting for MOSCRCEN bit to be cleared is recommended to ensure that the core will not execute undesired instructions.

In most applications, the external XTAL oscillator is used instead of on-chip RC oscillator for accurate clock. Before entering wait mode, it needs switch MCK from external XTAL oscillator to Fast RC oscillator. Switching Main Clock source will cause MCK unstable, so switching MCK source to stable Slow Clock is necessary before this operation. After Main Clock becomes stable, it needs switch MCK source back to Main Clock. The implementation is by calling following API:

```
pmc_switch_mck_to_sclk(PMC_MCKR_PRES_CLK_1);
pmc_switch_mainck_to_fastrc(CKGR_MOR_MOSCRCF_4_MHz);
pmc_switch_mck_to_mainck(PMC_PCK_PRES_CLK_1);
```

Note: `PMC_MCKR_PRES_CLK_1` indicates the master clock is equal to selected clock.

`CKGR_MOR_MOSCRCF_4_MHz` indicates the main clock is selected to 4MHz Fast RC oscillator.

The following API could be used to enter wait mode in ASF:

```
pmc_enable_waitmode()
```

## 4.2 Exit Wait Mode

Wait mode can be exited from several asynchronous fast start-up sources that have to be programmed prior to entering this mode, include:

- WKUP0-15 pins
- RTT or RTC alarm
- USB Wake-up

As soon as the fast start-up signal is asserted, the PMC automatically restarts the embedded 4/8/12 MHz fast RC oscillator, switches the master clock on this 4/8/12 MHz clock and re-enables the processor clock, then the core executes the next instruction of the program counter.

## 4.3 Using Wait Mode

Some applications are not running all the time. For example, an alarm application waits for signal from a sensor to wake up the microcontroller to launch a task. In this case, the microcontroller can be suspended during idle time and woken up in time when specific event happens. Both Sleep mode and Wait mode may be used for reducing power consumption. Some main factors should be considered:

- Wake-up source
- Power consumption
- Wake-up time

If the wakeup source is an interrupt in the application, only Sleep mode can be used because Wait mode can only be woken up by programmed events.

For power consumption, in Wait mode, current consumption is typically 50 µA when the internal voltage regulator is used. In Sleep mode, current consumption depends on master clock and peripherals in usage. The current consumption in Sleep mode is as low as Wait mode only when the master clock is running at 500Hz with all peripherals clock off. The higher master clock frequency, the higher power consumption in Sleep mode.

The microcontroller can be woken up in several microseconds in both Sleep and Wait mode. When waking up from Sleep mode, the core clock is re-enabled by PMC automatically. But in Wait mode, it needs more time. When waking up from Wait mode, the PMC automatically restarts the embedded 4/8/12 MHz fast RC oscillator, then switches the master clock to this clock. Normally, user needs to switch master clock to PLL clock to get higher frequency.

In short, the trade-off between wakeup time and power consumption must be taken into account when both Sleep mode and Wait mode can be used in the application. Generally, Wait mode can save more power than Sleep mode but Sleep mode has faster startup time than Wait mode.

# 5. Backup Mode

The purpose of backup mode is to achieve the lowest power consumption in a system but not requiring fast startup time. Backup mode is based on the Cortex-M4 deep-sleep mode with the voltage regulator disabled (no power supply for VDDCORE and VDDPLL). The core power supply domain is powered off and the SRAM, flash memory, PLL and peripherals are also switched off. The backup power supply domain is still powered so the Supply Controller, zero-power power-on reset, RTT, RTC, Backup registers and 32 kHz oscillator are running.

## 5.1 Enter Backup Mode

Backup mode is entered by using the VROFF bit of Supply Controller (SUPC_CR) and with the SLEEPDEEP bit in the Cortex-M4 System Control Register set to 1. The following API could be used to enter Backup Mode in ASF.

```
pmc_enable_backupmode()
```

## 5.2 Exit Backup Mode

Backup mode can be exited from several wake-up sources that must first be programmed prior to entering this mode, including:

- WKUPEN0-15 pins (level transition, configurable debouncing)
- RTC alarm
- RTT alarm
- Supply Monitor alarm

When waking up from Backup mode, the program execution restarts in the same way as system startup except backup region (RTC, RTT, GPBR and Supply Controller) are not reset.

## 5.3 Using Backup Mode

In some applications, very low power consumption is necessary. In this case, the Backup mode can be considered. In Backup mode, RTT and RTC can keep running and some important data can be stored in GPBR. Please make sure that the backup supply domain is still powered when entering Backup mode.

# 6. Revision History

| Doc.Rev | Date | Comments |
|---|---|---|
| 42142A | 06/2013 | Initial revision |


**Atmel Corporation**
2325 Orchard Parkway
San Jose, CA 95131
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 Building
1-6-4 Osaki
Shinagawa-ku, Tokyo 141-0032
JAPAN
**Tel:** (+81)(3) 6417-0300
**Fax:** (+81)(3) 6417-0370

© 2013 Atmel Corporation. All rights reserved. / Rev.: 42142A-SAM–06/2013

Atmel®, Atmel logo and combinations thereof, Enabling Unlimited Possibilities®, and others are registered trademarks or trademarks of Atmel Corporation or its subsidiaries. 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.