Application Note

AT91SAM ARM-based Embedded MPU

Using Low-power Modes on SAMA5D3 Series

1. Introduction

The SAMA5D3 series is a member of the Atmel® 32-bit microprocessor family which is based on the ARM® Cortex™-A5 processor core.

Low-power consumption is an essential consideration in a wide range of applications. This Application Note introduces the Low-power modes embedded on SAMA5D3 Series.

The document is organized in four main sections. The first two sections describe the Power Management Controller and the Low-power modes embedded on SAMA5D3 Series.

The last two sections introduce the Power Management in Linux OS and provide details on the implementation of the Low-power modes in Linux OS based on the SAMA5D3-EK board.

2. Associated Documentation

Before going further into this document, please refer to the latest documentation for the corresponding SAMA5D3 devices available on the Atmel® web site at http//:www.atmel.com:

- SAMA5D3 Series Datasheet: lit° 11121
- SAMA5D3-EK User Guide: lit° 11180
- Documentation/Power/ in Linux distribution

3. SAMA5D3 Power Management Controller (PMC)

3.1 Overview

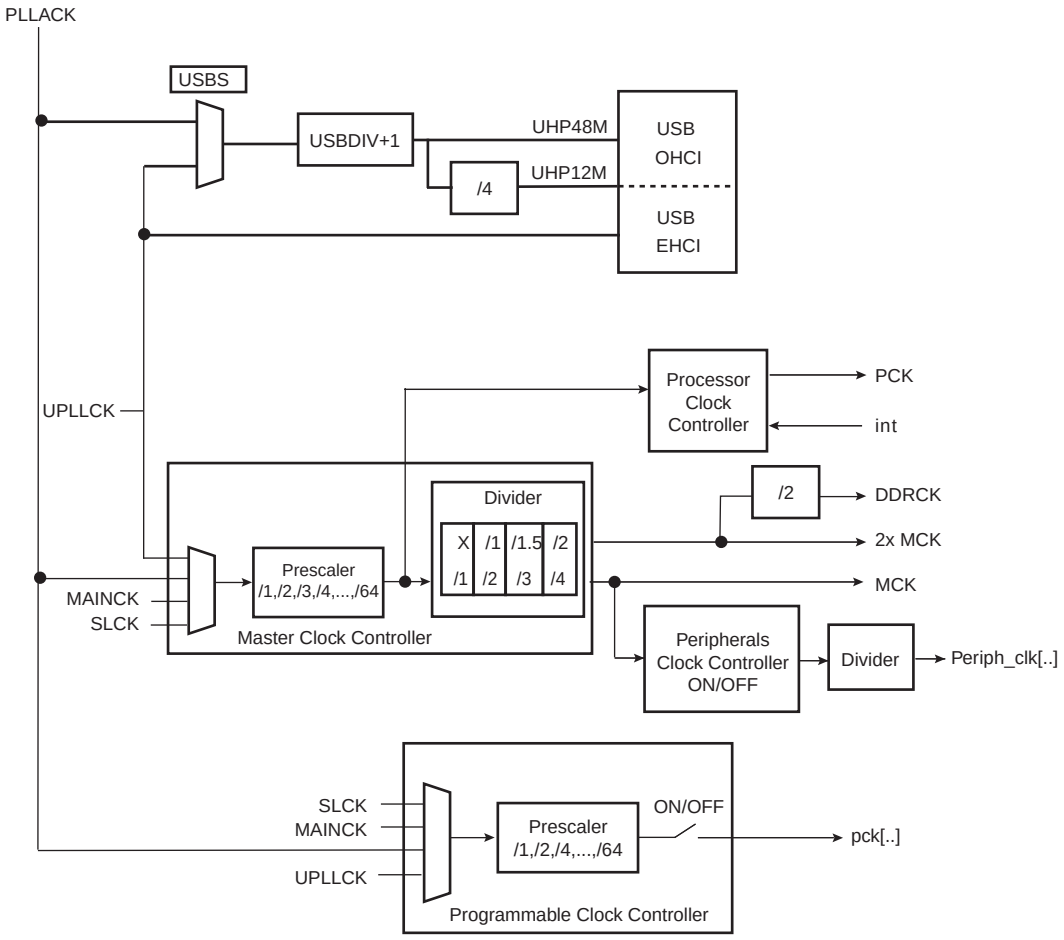
The Power Management Controller (PMC) optimizes power consumption by controlling all system and user peripheral clocks. The PMC enables/disables the clock inputs to many of the peripherals and the Core.

The Power Management Controller provides the following clocks:

- MCK, the Master Clock, programmable from a few hundred Hz to the maximum operating frequency of the device. It is available to the modules running permanently.
- Processor Clock (PCK), must be switched off when entering the processor in Sleep Mode.
- The USB Device HS Clock (UDPCK)
- The Software Modem Clock (SMDCK)
- Peripheral Clocks, typically MCK, provided to the embedded peripherals (USART, SSC, SPI, TWI, TC, HSMCI, etc.) and independently controllable. In order to reduce the number of clock names in a product, the Peripheral Clocks are named MCK in the product datasheet.
- Programmable Clock Outputs can be selected from the clocks provided by the clock generator and driven on the PCKx pins.

3.2 Block Diagram

Figure 3-1. General Clock Block Diagram



3.3 Master Clock Controller

The Master Clock Controller provides selection and division of the Master Clock (MCK). MCK is the clock provided to all the peripherals and the memory controller.

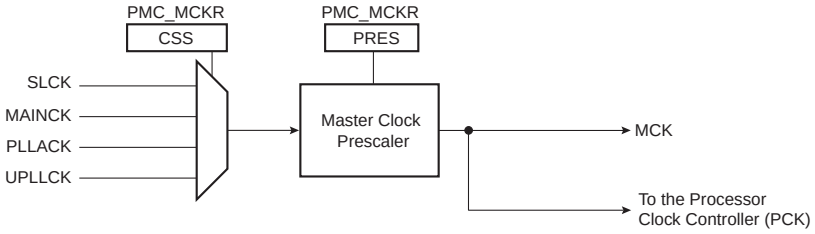
The Master Clock is selected from one of the clocks provided by the Clock Generator. Selecting the Slow Clock provides a Slow Clock signal to the whole device. Selecting the Main Clock saves power consumption of the PLLs.

The Master Clock Controller is made up of a clock selector and a prescaler. It also contains a Master Clock divider which allows the processor clock to be faster than the Master Clock.

The Master Clock selection is made by writing the CSS field (Clock Source Selection) in PMC_MCKR (Master Clock Register). The prescaler supports the division by a power of 2 of the selected clock between 1 and 64, and the division by 6. The PRES field in PMC_MCKR programs the prescaler.

Each time PMC_MCKR is written to define a new Master Clock, the MCKRDY bit is cleared in PMC_SR. It reads 0 until the Master Clock is established. Then, the MCKRDY bit is set and can trigger an interrupt to the processor. This feature is useful when switching from a high-speed clock to a lower one to inform the software when the change is actually done.

Figure 3-2. Master Block Diagram



3.4 Processor Clock Controller

The PMC features a Processor Clock Controller (PCK) that implements the Processor Idle Mode. The Processor Clock can be disabled by writing the System Clock Disable Register (PMC_SCDR). The status of this clock (at least for debug purpose) can be read in the System Clock Status Register (PMC_SCSR).

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

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

3.5 DDR2/LPDDR/LPDDR2 Clock

The Power Management Controller controls the clocks of the DDR memory.

The DDR clock can be enabled and disabled with the DDRCK bit respectively in the PMC_SCER and PMC_SDER registers. At reset, the DDR clock is disabled to save power consumption.

3.6 Peripheral Clock Controller

The Power Management Controller controls the clocks of each embedded peripheral by the way of the Peripheral Clock Controller. The user can individually enable and disable the clock on the peripherals and select a division factor from MCK. This is done with the help of the Peripheral Control Register (PMC_PCR). The peripheral clocks can be enabled and/or disabled via the PMC_PCER and PMC_PCDR registers.

When a peripheral clock is disabled, the clock is immediately stopped. The peripheral clocks are automatically disabled after a reset.

In order to stop a peripheral, it is recommended that the system software wait until the peripheral has executed its last programmed operation before disabling the clock. This is to avoid data corruption or erroneous behavior of the system.

3.7 Programmable Clock Output Controller

The PMC controls 2 signals to be outputs on external pins PCKx. Each signal can be independently programmed via the PMC_PCKx registers.

Each output signal can be enabled and disabled by writing 1 in the corresponding bit, PCKx of PMC_SCER and PMC_SCDR, respectively. Status of the active programmable output clocks are given in the PCKx bits of PMC_SCSR (System Clock Status Register).

3.8 Programming Sequence

The Programming sequence is as follows:

1. Enabling the 12 MHz Main Oscillator
2. Setting PLLA and Divider
3. Setting Bias and High-speed PLL(UPLL) for UTMI
4. Selecting Master Clock and Processor Clock
5. Selecting Programmable Clocks
6. Enabling Peripheral Clocks

4. SAMA5D3x Low-power Modes

The SAMA5D3x low-power modes include Backup Mode, Ultra Low-power Mode and Idle Mode.

4.1 Backup Mode

The purpose of Backup Mode is to achieve the lowest power consumption possible in a system, which is performing periodic wake-ups to perform tasks but not requiring fast startup time. Total current consumption is 1.2 µA typical.

The Zero-power Power-on Reset, RTC, Backup registers and 32 kHz Oscillator (RC or Crystal Oscillator selected by software in the Supply Controller) are running. The core supply is off.

The system can be awakened from this mode through the WKUP0 pin or an RTC wake-up event.

Backup Mode is entered into with the help of the Shutdown Controller that asserts the SHDN output pin. The SHDN pin is to be connected to the Enable of the VDDCORE regulator.

Exit from Backup Mode happens if one of the following enable wake-up events occurs:

- WKUP0 pin (level transition, configurable debouncing)
- RTC alarm

The system will restart for a reset event.

4.2 Ultra Low-power Mode

The purpose of Ultra Low-power Mode is to reduce the power consumption of the device to the minimum without removing VDDCORE power supply. It is a combination of very low frequency operations and Idle Mode.

This mode is entered via the following steps:

1. Set the DDR in Self Refresh Mode
2. Reduce the system clock (PCK and MCK) to the minimum with the help of the PMC:
   - PCK and MCK configuration is to be defined regarding the expected power consumption and wake-up time. Please refer to Table 4-1 for details
   - PLLs are disabled. CKGR_PLLAR (eventually CKGR_PLLBR) is set to 0x3f00. CKGR_UCKR is set to 0.
   - Main Oscillator is disabled. MOSCXTEN is set to 0 in CKGR_MOR.
   - Eventually 12 MHz RC Oscillator is disabled. MOSCRCEN is set to 0 in CKGR_MOR.
3. Enter into Wait for Interrupt (WFI) mode and disable the PCK clock.

The processor can be awakened from an interrupt.

Once woken up, the system must reprogram the system clocks (OSC, PLL,PCK, MCK, DDRCK) to recover the previous state. Data is maintained in the external memory.

4.3 Idle Mode

The purpose of Idle Mode is to optimize power consumption of the device versus response time. In this mode, only the core clock is stopped. The peripheral clocks, including the DDR Controller clock, can be enabled. The current consumption in this mode is application dependent.

This mode is entered via the Wait for Interrupt (WFI) instruction and PCK disabling.

The processor can be awakened from an interrupt. The system will resume where it was before entering into WFI mode.

4.4 Low Power Summary Table

The modes detailed above are the main low-power modes. Each part can be set to ON or OFF separately and wake-up sources can be individually configured. Table 4-1 below shows a summary of the configurations of the low-power modes.

Table 4-1. Low-power Mode Configuration Summary

Mode	32K RC, 12 MHz RC,, 32 kHz Osc, RTC, Backup Registers, POR (Backup Region)	VDDCORE Regulator	Core Memory Peripherals	Mode Entry	Potential Wake-Up Sources	Core at Wake Up	PIO State while in Low Power Mode	PIO State at Wake Up	Consumption(2)	Wake-Up Time(1)
Backup	ON	OFF	OFF (Not powered)	Shutdown Controller	WKUP0 pin RTC alarm	Reset	Reset	Inputs with pull-ups	1.2 µA typ(3)	Start-up time
Idle	ON	ON	Powered (Not clocked)	WFI	Any interrupt	Clocked back at full speed	Previous state saved	Unchanged	19 mA on VDDCORE @ 133 MHz 5 - 8 mA for each PLL(4)	350 ns @ 133 MHz
Ultra Low-power	ON	ON	Powered (Not clocked)	DDR in Self Refresh PMC WFI	Any interrupt	Clocked back at previous one	Previous state saved	Unchanged	1780 µA @ 12 MHz 520 µA @ 750 kHz 455 µA @ 187 kHz 432 µA @ 32 kHz 429 µA @ 512 Hz	3.9 µs @ 12 MHz 60 µs @ 750 kHz 230 µs @ 187 kHz 1.4 ms @ 32 kHz 89 ms @ 512 Hz

Notes:
1. When considering wake-up time, the time required to start the PLL is not taken into account. Once started, the device works with the Main Oscillator. The user has to add the PLL start-up time if it is needed in the system. The wake-up time is defined as the time taken for wake up until the first instruction is fetched.
2. The external loads on PIOs are not taken into account in the calculation.
3. Total Current consumption.
4. Depends on MCK frequency.

5. Power Management in Linux OS

Power management in Linux is performed by the PM core implemented in the OS. The Linux kernel supports three power management states generically, which include Standby state, Suspend-to-RAM state and Suspend-to-disk state.

5.1 Standby State

This state offers minimal but real power savings while providing a very low-latency transition back to the working system. No operating state is lost (the CPU retains powered), so the system easily starts up from where it was left off. The devices are set up in a low-power state, which also offers low-power savings with low resume latency.

A transition from the Standby state to the On state takes about 1-2 seconds.

5.2 Suspend-to-RAM State

This state offers significant power savings as everything in the system is set up into a low-power state, except for memory, which is set in self-refresh mode to retain its contents.

System and device state is saved and kept in memory. All devices are suspended and set up into the low-power state. In many cases, all peripheral buses lose power when entering the Suspend-to-RAM state; thus, devices must be able to handle the transition back to the On state.

A transition from the Suspend-to-RAM state to the On state takes about 3-5 seconds.

5.3 Suspend-to-Disk State

This state offers the greatest power savings, and can be used even in the absence of a low-level platform support for power management. This state operates similarly to the Suspend-to-RAM state, but includes a final step of writing the memory content to disk. On resume, the memory content is read and the memory is restored to its pre-suspend state.

A transition from the Suspend-to-Disk state to the On state takes about 30 seconds; however, it's typically a bit more with the current implementation.

6. Example of Low-power Implementation based on SAMA5D3x-EK

This section presents an example of Low-power Mode implementation in Linux OS, based on the SAMA5D3x-EK board. Te following Linux low-power modes are mainly implemented on SAMA5D3x:

- Standby mode
- Suspend-to-RAM mode

6.1 Implementation Sample Code

The Implementation Sample code is located at: *http://lxr.linux.no/linux+v2.6.39/arch/arm/mach-at91/pm.c#L206*.

Note: The SAM9N12 Linux code is temporarily used here as the Sample code, since the SAMA5D3x Linux code is still not available and the Low-power code is similar for both devices.

Before entering in any low-power modes, the user can print out debug information on all the key conditions, such as GPIOs, interrupts, etc.

For all the GPIOs and interrupts, enable those that are marked as 'wakeup' event sources, and suspend those that are not marked as 'wakeup' event sources.

Example code:

```
at91_gpio_suspend();
at91_irq_suspend();
```

6.1.1 Standby Mode

At Standby Mode, all the drivers are suspended. The system ignores the interrupts that are not marked as 'wakeup' event sources and reduces DRAM power.

Each enabled peripheral increases power consumption. Print out status information of all the peripheral clocks. Disable the clock of the peripheral that is not needed before the system enters into low-power modes, and enable it after the system wakes up.

Sample code:

```
case PM_SUSPEND_STANDBY:
      /*
      * NOTE: the Wait-for-Interrupt instruction needs to be
      * in icache so no SDRAM accesses are needed until the
      * wakeup IRQ occurs and self-refresh is terminated.
      * For ARM 926 based chips, this requirement is weaker
      * as at91sam9 can access a RAM in self-refresh mode.
      */

      printk("PMC_PCSR: 0x%x\n", at91_sys_read(AT91_PMC_PCSR));

      at91_sys_write(AT91_PMC_PCDR, 1 << AT91SAM9N12_ID_ADC);
      at91_sys_write(AT91_PMC_PCDR, 1 << AT91SAM9N12_ID_SSC);
      at91_sys_write(AT91_PMC_PCDR, 1 << AT91SAM9N12_ID_UHPFS);
      at91_sys_write(AT91_PMC_PCDR, 1 << AT91SAM9N12_ID_TCB);

      at91_sys_write(AT91_PMC_SCDR, AT91SAM926x_PMC_UHP);
      asm volatile (  "mov r0, #0\n\t"
                                                "b 1f\n\t"
                                                ".align 5\n\t"
                                                "1: mcr p15, 0, r0, c7, c10,
4\n\t"
                                                : /* no output */
                                                : /* no input */
                                                : "r0");
```

```
        saved_lpr = sdram_selfrefresh_enable(); // Set the DDR in Self Refresh Mode
        my_arch_idle(); // Disable the Processor clock and enter into WFI Mode
        sdram_selfrefresh_disable(saved_lpr); //Once woken up, exit from Self Refresh
Mode

        at91_sys_write(AT91_PMC_SCER, AT91SAM926x_PMC_UHP);

        at91_sys_write(AT91_PMC_PCER, 1 << AT91SAM9N12_ID_ADC);
        at91_sys_write(AT91_PMC_PCER, 1 << AT91SAM9N12_ID_SSC);
        at91_sys_write(AT91_PMC_PCER, 1 << AT91SAM9N12_ID_UHPFS);
        at91_sys_write(AT91_PMC_PCER, 1 << AT91SAM9N12_ID_TCB);

        my_sleep(10000);
        break;
```

6.1.2 Suspend-to-RAM Mode

Suspend-to-RAM mode combines Standby mode and Slow Clock mode: the drivers are suspended more deeply and only the main oscillator may be used by the master clock.

Sample code:

```
case PM_SUSPEND_MEM:
        /*
        * Ensure that clocks are in a valid state.
        */
        if (!at91_pm_verify_clocks())
                goto error;

        /*
        * Enter slow clock mode by switching over to clk32k and
        * turning off the main oscillator; reverse on wakeup.
        */
        if (slow_clock) {
#ifdef CONFIG_AT91_SLOW_CLOCK
                /* copy slow_clock handler to SRAM, and call it */
                memcpy(slow_clock, at91_slow_clock, at91_slow_clock_sz);
#endif
                slow_clock();
                break;
        } else {
                pr_info("AT91: PM - no slow clock mode enabled ...\n");
                /* FALLTHROUGH leaving master clock alone */
        }
```

Revision History

In the table that follows, the most recent version of the document appears first.

“rfo” indicates changes requested during the document review and approval loop.

Doc. Rev	Comments	Change Request Ref.
11185A	First issue.	

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