AT11378: SAM System (SYSTEM) Driver

APPLICATION NOTE

Introduction

This driver for Atmel® | SMART ARM®-based microcontrollers provides an interface for the configuration and management of the device's system relation functionality, necessary for the basic device operation. This is not limited to a single peripheral, but extends across multiple hardware peripherals.

The following peripherals are used by this module:

- PM (Power Manager)
- RSTC(Reset Controller)
- SUPC(Supply Controller)

The following devices can use this module:

- Atmel | SMART SAM L21

The outline of this documentation is as follows:

- Prerequisites
- Module Overview
- Special Considerations
- Extra Information
- Examples
- API Overview

Table of Contents

Introduction....1

1. Software License....4

2. Prerequisites....5

3. Module Overview....6

3.1. Voltage Regulator....6
3.2. Battery Backup Power Switch....6
3.3. Voltage References....6
3.4. System Reset Cause....6
3.5. Performance Level....7
3.6. Power Domain Gating....8
3.7. RAMs Low Power Mode....9
3.8. Sleep Modes....10

4. Special Considerations....11

5. Extra Information....12

6. Examples....13

7. API Overview....14

7.1. Structure Definitions....14
7.1.1. Struct system_battery_backup_power_switch_config....14
7.1.2. Struct system_standby_config....14
7.1.3. Struct system_voltage_references_config....14
7.1.4. Struct system_voltage_regulator_config....15
7.2. Function Definitions....15
7.2.1. Voltage Regulator....15
7.2.2. Voltage References....16
7.2.3. Battery Backup Power Switch....17
7.2.4. Output Pins in Backup Mode....18
7.2.5. Device Sleep Control....20
7.2.6. Performance Level Control....21
7.2.7. Standby Configuration....22
7.2.8. I/O Retention....22
7.2.9. Reset Control....23
7.2.10. Backup Exit Control....23
7.2.11. System Debugger....25
7.2.12. System Identification....25
7.2.13. System Initialization....25
7.3. Enumeration Definitions....26
7.3.1. Enum system_backup_pin....26
7.3.2. Enum system_battery_power_switch....26
7.3.3. Enum system_linked_power_domain....26

7.3.4. Enum system_performance_level 26
7.3.5. Enum system_power_domain 27
7.3.6. Enum system_ram_back_bias_mode 27
7.3.7. Enum system_reset_backup_exit_source 27
7.3.8. Enum system_reset_cause 28
7.3.9. Enum system_sleepmode 28
7.3.10. Enum system_voltage_reference 28
7.3.11. Enum system_voltage_references_sel 28
7.3.12. Enum system_voltage_regulator_low_power_efficiency 29
7.3.13. Enum system_voltage_regulator_sel 29
7.3.14. Enum system_wakeup_debounce_count 30

8. Examples for Power Driver 31
8.1. Quick Start Guide for Power Driver 31
8.1.1. Quick Start 31
8.1.2. Use Case 33

9. Extra Information for SYSTEM Driver 37
9.1. Acronyms 37
9.2. Dependencies 37
9.3. Errata 37
9.4. Module History 37

10. Document Revision History 38

1. Software License

2. Prerequisites

There are no prerequisites for this module.

3. Module Overview

The System driver provides a collection of interfaces between the user application logic, and the core device functionality (such as clocks, reset cause determination, etc.) that is required for all applications. It contains a number of sub-modules that control one specific aspect of the device:

- System Core (this module)
- System Clock Control (sub-module)
- System Interrupt Control (sub-module)
- System Pin Multiplexer Control (sub-module)

3.1. Voltage Regulator

The SAM device controls the voltage regulators for the core (VDDCORE) and backup (VDDBU) domains. It sets the voltage regulators according to the sleep modes, the performance level, or the user configuration.

In active mode, the voltage regulator can be chosen on the fly between a LDO or a Buck converter. In standby mode, the low power voltage regulator is used to supply VDDCORE.

3.2. Battery Backup Power Switch

The SAM device supports connection of a battery backup to the VBAT power pin. It includes functionality that enables automatic power switching between main power and battery backup power. This will ensure power to the backup domain, when the main battery or power source is unavailable.

3.3. Voltage References

The various analog modules within the SAM devices (such as AC, ADC, and DAC) require a voltage reference to be configured to act as a reference point for comparisons and conversions.

The SAM devices contain multiple references, including an internal temperature sensor and a fixed band-gap voltage source. When enabled, the associated voltage reference can be selected within the desired peripheral where applicable.

3.4. System Reset Cause

In some applications there may be a need to execute a different program flow based on how the device was reset. For example, if the cause of reset was the Watchdog timer (WDT), this might indicate an error in the application, and a form of error handling or error logging might be needed.

For this reason, an API is provided to retrieve the cause of the last system reset, so that appropriate action can be taken.

There are three groups of reset sources:

- Power supply reset: Resets caused by an electrical issue. It covers POR and BOD reset.
- User reset: Resets caused by the application. It covers external reset, system reset, and watchdog reset.

- Backup reset: Resets caused by a backup mode exit condition.

3.5. Performance Level

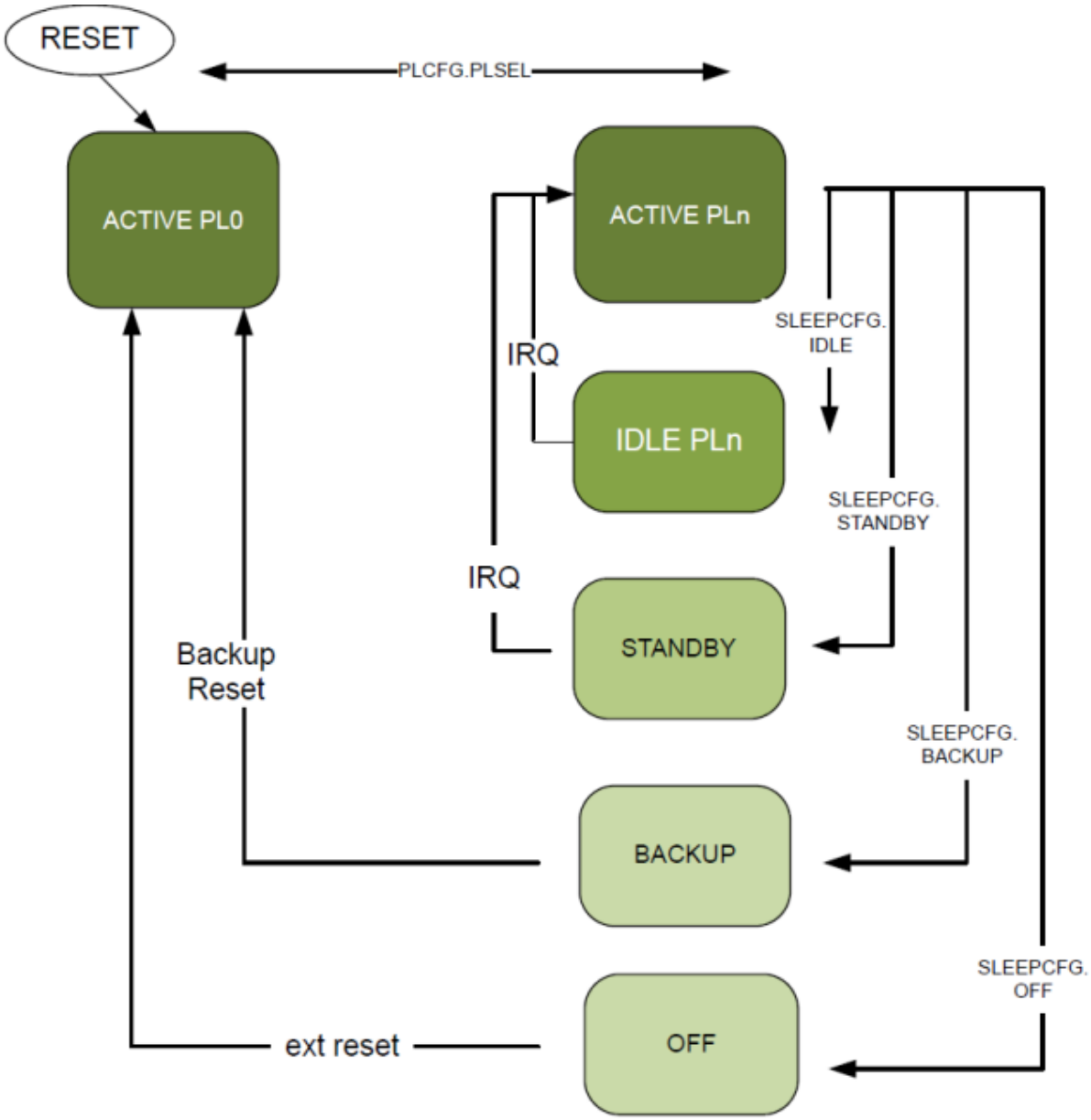
Performance level allows the user to adjust the regulator output voltage to reduce power consumption. The user can on the fly select the most suitable performance level, depending on the application demands.

The SAM device can operate at two different performance levels (PL0 and PL2). When operating at PL0, the voltage applied on the full logic area is reduced by voltage scaling. This voltage scaling technique allows to reduce the active power consumption while decreasing the maximum frequency of the device. When operating at PL2, the voltage regulator supplies the highest voltage, allowing the device to run at higher clock speeds.

Performance level transition is possible only when the device is in active mode. After a reset, the device starts at the lowest performance level (lowest power consumption and lowest max. frequency). The application can then switch to another performance level at any time without any stop in the code execution. As shown in Figure 3-1 Performance Level Transition on page 8.

Note: When scaling down the performance level, the bus frequency should first be scaled down in order to not exceed the maximum frequency allowed for the low performance level. When scaling up the performance level (e.g. from PL0 to PL2), check the performance level status before increasing the bus frequency. It can be increased only when the performance level transition is completed.

Figure 3-1 Performance Level Transition



3.6. Power Domain Gating

Power domain gating allows power saving by reducing the voltage in logic areas in the device to a low-power supply. The feature is available in Standby sleep mode and will reduce the voltage in domains where all peripherals are idle. Internal logic will maintain its content, meaning the corresponding peripherals will not need to be reconfigured when normal operating voltage is returned. Most power domains can be in the following three states:

- Active state: The power domain is powered on.
- Retention state: The main voltage supply for the power domain is switched off, while maintaining a secondary low-power supply for the sequential cells. The logic context is restored when waking up.
- Off state: The power domain is entirely powered off. The logic context is lost.

The SAM L21 device contains three power domains which can be controlled using power domain gating, namely PD0, PD1, and PD2. These power domains can be configured to the following cases:

- Default with no sleepwalking peripherals: A power domain is automatically set to retention state in standby sleep mode if no activity require it. The application can force all power domains to remain in active state during standby sleep mode in order to accelerate wakeup time.
- Default with sleepwalking peripherals: If one or more peripherals are enabled to perform sleepwalking tasks in standby sleep mode, the corresponding power domain (PDn) remains in active state as well as all inferior power domains (<PDn).
- Sleepwalking with dynamic power domain gating: During standby sleep mode, a power domain (PDn) in active can wake up a superior power domain (>PDn) in order to perform a sleepwalking task. The superior power domain is then automatically set to active state. At the end of the sleepwalking task, the device can either be woken up or the superior power domain can return to retention state.

Power domains can be linked to each other, it allows a power domain (PDn) to be kept in active state if the inferior power domain (PDn-1) is in active state too.

Table 3-1 Sleep Mode versus Power Domain State Overview on page 9 illustrates the four cases to consider in standby mode.

Table 3-1 Sleep Mode versus Power Domain State Overview

Sleep mode	PD0	PD1	PD2	PDTOP	PDBACKUP
Idle	active	active	active	active	active
Standby - Case 1	active	active	active	active	active
Standby - Case 2	active	active	retention	active	active
Standby - Case 3	active	retention	retention	active	active
Standby - Case 4	retention	retention	retention	active	active
Backup	off	off	off	off	active
Off	off	off	off	off	off

3.7. RAMs Low Power Mode

By default, in standby sleep mode, RAM is in low power mode (back biased) if its power domain is in retention state. Table 3-2 RAM Back-biasing Mode on page 9 lists RAMs low power mode.

Table 3-2 RAM Back-biasing Mode

RAM mode	Description
Retention Back-biasing mode	RAM is back-biased if its power domain is in retention mode
Standby Back-biasing mode	RAM is back-biased if the device is in standby mode
Standby OFF mode	RAM is OFF if the device is in standby mode
Always OFF mode	RAM is OFF if the device is in RET mode

3.8. Sleep Modes

The SAM devices have several sleep modes. The sleep mode controls which clock systems on the device will remain enabled or disabled when the device enters a low power sleep mode. Table 3-3 SAM Device Sleep Modes on page 10 lists the clock settings of the different sleep modes.

Table 3-3 SAM Device Sleep Modes

Sleep mode	System clock	CPU clock	AHB/AHB clock	GCLK clocks	Oscillators (ONDEMAND = 0)	Oscillators (ONDEMAND = 1)	Regulator mode	RAM mode
Idle	Run	Stop	Run if requested	Run	Run	Run if requested	Normal	Normal
Standby	Stop	Stop	Run if requested	Run if requested	Run if requested or RUNSTDBY = 1	Run if requested	Low pwer	Low pwer
Backup	Stop	Stop	Stop	Stop	Stop	Stop	Backup	Off
Off	Off	Off	Off	Off	Off	Off	Off	Off

Before entering device sleep, one of the available sleep modes must be set. The device will automatically wake up in response to an interrupt being generated or upon any other sleep mode exit condition.

Some peripheral clocks will remain enabled during sleep, depending on their configuration. If desired, the modules can remain clocked during sleep to allow them continue to operate while other parts of the system are powered down to save power.

4. Special Considerations

Most of the functions in this driver have device specific restrictions and caveats; refer to your device datasheet.

5. Extra Information

For extra information, see Extra Information for SYSTEM Driver. This includes:

- Acronyms
- Dependencies
- Errata
- Module History

6. Examples

For SYSTEM module related examples, refer to the sub-modules listed in the system module overview.

For a list of examples related to this driver, see Examples for Power Driver.

7. API Overview

7.1. Structure Definitions

7.1.1. Struct system_battery_backup_power_switch_config

Configuration structure for Battery Backup Power Switch (BBPS).

Table 7-1 Members

Type	Name	Description
enum system_battery_power_switch	battery_power_switch	Battery backup power switch configuration
bool	wake_enabled	Enable device wake up when BBPS switches from battery backup power to main power

7.1.2. Struct system_standby_config

Configuration structure for standby mode.

Table 7-2 Members

Type	Name	Description
bool	disable_avregsd	Automatic VREG switching disable
bool	enable_dpgpd0	Enable dynamic power gating for power domain 0
bool	enable_dpgpd1	Enable dynamic power gating for power domain 1
enum system_ram_back_bias_mode	hmcramchs_back_bias	Back bias for HMCRAMCHS
enum system_ram_back_bias_mode	hmcramclp_back_bias	Back bias for HMCRAMCLP
enum system_linked_power_domain	linked_power_domain	Linked power domain
enum system_power_domain	power_domain	Power domain

7.1.3. Struct system_voltage_references_config

Configuration structure for VREF.

Table 7-3 Members

Type	Name	Description
bool	on_demand	On Demand Control
bool	run_in_standby	Run in standby
enum system_voltage_references_sel	sel	Voltage References Selection

7.1.4. Struct system_voltage_regulator_config

Configuration structure for VREG.

Table 7-4 Members

Type	Name	Description
enum system_voltage_regulator_low_power_efficiency	low_power_efficiency	Low power efficiency
enum system_voltage_regulator_sel	regulator_sel	Voltage Regulator Selection
bool	run_in_standby	Run in standby in standby sleep mode
uint8_t	voltage_scale_period	Voltage scaling period
uint8_t	voltage_scale_step	Voltage scaling voltage step

7.2. Function Definitions

7.2.1. Voltage Regulator

7.2.1.1. Function system_voltage_regulator_get_config_defaults()

Retrieve the default configuration for voltage regulator.

```
void system_voltage_regulator_get_config_defaults(
        struct system_voltage_regulator_config *const config)
```

Fills a configuration structure with the default configuration:

- Voltage scaling period is 1µs
- Voltage scaling voltage step is 2*min_step
- The voltage regulator is in low power mode in Standby sleep mode
- The voltage regulator in active mode is an LDO voltage regulator
- The voltage regulator in Low power mode has the default efficiency

Table 7-5 Parameters

Data direction	Parameter name	Description
[out]	config	Configuration structure to fill with default values

7.2.1.2. Function system_voltage_regulator_set_config()

Configure voltage regulator.

```
void system_voltage_regulator_set_config(
        struct system_voltage_regulator_config *const config)
```

Configures voltage regulator with the given configuration.

Table 7-6 Parameters

Data direction	Parameter name	Description
[in]	config	Voltage regulator configuration structure containing the new config

7.2.1.3. Function system_voltage_regulator_enable()

Enable the selected voltage regulator.

```
void system_voltage_regulator_enable( void )
```

Enables the selected voltage regulator source.

7.2.1.4. Function system_voltage_regulator_disable()

Disable the selected voltage regulator.

```
void system_voltage_regulator_disable( void )
```

Disables the selected voltage regulator.

7.2.2. Voltage References

7.2.2.1. Function system_voltage_reference_get_config_defaults()

Retrieve the default configuration for voltage reference.

```
void system_voltage_reference_get_config_defaults(
        struct system_voltage_references_config *const config)
```

Fill a configuration structure with the default configuration:

- 1.0V voltage reference typical value
- On demand control disabled
- The voltage reference and the temperature sensor are halted during standby sleep mode

Table 7-7 Parameters

Data direction	Parameter name	Description
[out]	config	Configuration structure to fill with default values

7.2.2.2. Function system_voltage_reference_set_config()

Configure voltage reference.

```
void system_voltage_reference_set_config(
        struct system_voltage_references_config *const config)
```

Configures voltage reference with the given configuration.

Table 7-8 Parameters

Data direction	Parameter name	Description
[in]	config	Voltage reference configuration structure containing the new config

7.2.2.3. Function system_voltage_reference_enable()

Enable the selected voltage reference.

```
void system_voltage_reference_enable(
        const enum system_voltage_reference vref)
```

Enables the selected voltage reference source, making the voltage reference available on a pin as well as an input source to the analog peripherals.

Table 7-9 Parameters

Data direction	Parameter name	Description
[in]	vref	Voltage reference to enable

7.2.2.4. Function system_voltage_reference_disable()

Disable the selected voltage reference.

```
void system_voltage_reference_disable(
        const enum system_voltage_reference vref)
```

Disables the selected voltage reference source.

Table 7-10 Parameters

Data direction	Parameter name	Description
[in]	vref	Voltage reference to disable

7.2.3. Battery Backup Power Switch

7.2.3.1. Function system_battery_backup_power_switch_get_config_defaults()

Retrieve the default configuration for battery backup power switch control.

```
void system_battery_backup_power_switch_get_config_defaults(
        struct system_battery_backup_power_switch_config *const config)
```

Fills a configuration structure with the default configuration:

- The main Power Supply OK status is not available on the PSOK pin
- The device is not woken up when switched from battery backup power to main power
- The backup domain is always supplied by main power

Table 7-11 Parameters

Data direction	Parameter name	Description
[out]	config	Configuration structure to fill with default values

7.2.3.2. Function system_battery_backup_power_switch_set_config()

Configure battery backup power switch.

```
void system_battery_backup_power_switch_set_config(
        struct system_battery_backup_power_switch_config *const config)
```

Configures battery backup power switch with the given configuration.

Table 7-12 Parameters

Data direction	Parameter name	Description
[in]	config	Battery backup power switch configuration structure containing the new config

7.2.4. Output Pins in Backup Mode

7.2.4.1. Function system_backup_pin_output_enable()

Enable the backup pin output.

```
void system_backup_pin_output_enable(
        enum system_backup_pin pin)
```

The output is enabled and driven by the SUPC.

Table 7-13 Parameters

Data direction	Parameter name	Description
[in]	pin	Backup pin index

7.2.4.2. Function system_backup_pin_output_disable()

Disable the backup pin output.

```
void system_backup_pin_output_disable(
        enum system_backup_pin pin)
```

The output is not enabled.

Table 7-14 Parameters

Data direction	Parameter name	Description
[in]	pin	Backup pin index

7.2.4.3. Function system_backup_pin_output_is_enabled()

Check if backup pin output is enabled.

```
bool system_backup_pin_output_is_enabled(
        enum system_backup_pin pin)
```

Table 7-15 Parameters

Data direction	Parameter name	Description
[in]	pin	Backup pin index

Returns
The enabled status.

Table 7-16 Return Values

Return value	Description
true	The output is enabled
false	The output is not enabled

7.2.4.4. Function system_backup_pin_output_enable_rtc_toggle()

Enable the backup pin toggle on RTC event.

```
void system_backup_pin_output_enable_rtc_toggle(
        enum system_backup_pin pin)
```

Toggle output on RTC event is enabled.

Table 7-17 Parameters

Data direction	Parameter name	Description
[in]	pin	Backup pin index

7.2.4.5. Function system_backup_pin_output_disable_rtc_toggle()

Disable the backup pin toggle on RTC event.

```
void system_backup_pin_output_disable_rtc_toggle(
        enum system_backup_pin pin)
```

Toggle output on RTC event is disabled.

Table 7-18 Parameters

Data direction	Parameter name	Description
[in]	pin	Backup pin index

7.2.4.6. Function system_backup_pin_output_set()

Set the backup pin.

```
void system_backup_pin_output_set(
        enum system_backup_pin pin)
```

Set the corresponding output pin.

Table 7-19 Parameters

Data direction	Parameter name	Description
[in]	pin	Backup pin index

7.2.4.7. Function system_backup_pin_output_clear()

Clear the backup pin.

```
void system_backup_pin_output_clear(
        enum system_backup_pin pin)
```

Clear the corresponding output.

Table 7-20 Parameters

Data direction	Parameter name	Description
[in]	pin	Backup pin index

7.2.4.8. Function system_backup_pin_output_get()

Get the backup I/O input values.

```
bool system_backup_pin_output_get(
        enum system_backup_pin pin)
```

Get the backup I/O data input values. If the corresponding pin is enabled, the I/O input value is given on the pin.

Table 7-21 Parameters

Data direction	Parameter name	Description
[in]	pin	Backup pin index

Returns
The backup I/O input level value.

7.2.5. Device Sleep Control

7.2.5.1. Function system_set_sleepmode()

Set the sleep mode of the device.

```
void system_set_sleepmode(
        const enum system_sleepmode sleep_mode)
```

Sets the sleep mode of the device; the configured sleep mode will be entered upon the next call of the system_sleep() function.

For an overview of which systems are disabled in sleep for the different sleep modes, see Sleep Modes.

Table 7-22 Parameters

Data direction	Parameter name	Description
[in]	sleep_mode	Sleep mode to configure for the next sleep operation

7.2.5.2. Function system_sleep()

Put the system to sleep waiting for interrupt.

```
void system_sleep( void )
```

Executes a device DSB (Data Synchronization Barrier) instruction to ensure all ongoing memory accesses have completed. Further, a WFI (Wait For Interrupt) instruction is executed to place the device into the sleep mode specified by system_set_sleepmode.

7.2.6. Performance Level Control

7.2.6.1. Function system_switch_performance_level()

Switch performance level.

```
enum status_code system_switch_performance_level(
        const enum system_performance_level performance_level)
```

The bus frequency must be reduced prior to scaling down the performance level, in order to not exceed the maximum frequency allowed for the performance level.

When scaling up the performance level (for example from PL0 to PL2), the bus frequency can be increased first when the performance level transition is completed. Check the performance level status before increasing the frequency.

Table 7-23 Parameters

Data direction	Parameter name	Description
[in]	performance_level	Performance level to switch

Table 7-24 Return Values

Return value	Description
STATUS_ERR_INVALID_ARG	Invalid parameter
STATUS_OK	Successfully

7.2.6.2. Function system_get_performance_level()

Get performance level.

```
enum system_performance_level system_get_performance_level( void )
```

Get performance level.

Returns
Current performance level.

7.2.6.3. Function system_get_performance_level_status()

Get performance level status.

```
uint8_t system_get_performance_level_status( void )
```

Returns
Performance level status: Written to one when the performance level is ready.

7.2.6.4. Function system_clear_performance_level_status()

Clear performance level status.

```
void system_clear_performance_level_status( void )
```

Clear performance level status.

7.2.7. Standby Configuration

7.2.7.1. Function system_standby_get_config_defaults()

Retrieve the default configuration for standby.

```
void system_standby_get_config_defaults(
        struct system_standby_config *const config)
```

Fills a configuration structure with the default configuration for standby:

- Retention back biasing mode for PICOPRAM
- Retention back biasing mode for HMCRAMCLP
- Retention back biasing mode for HMCRAMCHS
- Power domains PD0/PD1/PD2 are not linked
- Automatic VREG switching is used
- Dynamic power gating for power domain 1 is disabled
- Dynamic power gating for power domain 0 is disabled
- All power domains switching are handled by hardware

Table 7-25 Parameters

Data direction	Parameter name	Description
[out]	config	Configuration structure to fill with default values

7.2.7.2. Function system_standby_set_config()

Configure standby mode.

```
void system_standby_set_config(
        struct system_standby_config *const config)
```

Configures standby with the given configuration.

Table 7-26 Parameters

Data direction	Parameter name	Description
[in]	config	Standby configuration structure containing the new config

7.2.8. I/O Retention

7.2.8.1. Function system_io_retension_enable()

Enable I/O retention.

```
void system_io_retension_enable( void )
```

Enable I/O retention. After waking up from Backup mode, I/O lines are held until the bit is written to 0.

7.2.8.2. Function system_io_retension_disable()

Disable I/O retention.

```
void system_io_retension_disable( void )
```

Disable IO retention. After waking up from Backup mode, I/O lines are not held.

7.2.9. Reset Control

7.2.9.1. Function system_reset()

Reset the MCU.

```
void system_reset( void )
```

Resets the MCU and all associated peripherals and registers, except RTC, OSC32KCTRL, RSTC, GCLK (if WRTLOCK is set), and I/O retention state of PM.

7.2.9.2. Function system_get_reset_cause()

Get the reset cause.

```
enum system_reset_cause system_get_reset_cause( void )
```

Retrieves the cause of the last system reset.

Returns
An enum value indicating the cause of the last system reset.

7.2.10. Backup Exit Control

7.2.10.1. Function system_get_backup_exit_source()

Get the backup exit source.

```
enum system_reset_backup_exit_source system_get_backup_exit_source( void )
```

Get the backup exit source when a backup reset occurs.

Returns
An enum value indicating the latest backup exit source.

7.2.10.2. Function system_set_pin_wakeup_debounce_counter()

Set wakeup debounce counter.

```
void system_set_pin_wakeup_debounce_counter(
        const enum system_wakeup_debounce_count wakeup_debounce_count)
```

Set the wakeup debounce counter value with the given count.

Table 7-27 Parameters

Data direction	Parameter name	Description
[in]	wakeup_debounce_count	Wakeup debounce counter value

7.2.10.3. Function system_set_pin_wakeup_polarity_low()

Set low polarity of wakeup input pin.

```
void system_set_pin_wakeup_polarity_low(
        const uint16_t pin_mask)
```

Set low polarity with the given wakeup input pin mask.

Table 7-28 Parameters

Data direction	Parameter name	Description
[in]	pin_mask	Input pin mask

7.2.10.4. Function system_set_pin_wakeup_polarity_high()

Set high polarity of wakeup input pin.

```
void system_set_pin_wakeup_polarity_high(
        const uint16_t pin_mask)
```

Set high polarity with the given wakeup input pin mask.

Table 7-29 Parameters

Data direction	Parameter name	Description
[in]	pin_mask	Input pin mask

7.2.10.5. Function system_enable_pin_wakeup()

Enable wakeup of input pin from the backup mode.

```
void system_enable_pin_wakeup(
        const uint16_t pin_mask)
```

Enable pin wakeup from the backup mode with the given pin mask.

Table 7-30 Parameters

Data direction	Parameter name	Description
[in]	pin	Input pin mask

7.2.10.6. Function system_disable_pin_wakeup()

Disable wakeup of input pin from the backup mode.

```
void system_disable_pin_wakeup(
        const uint16_t pin_mask)
```

Disable pin wakeup from the backup mode with the given pin mask.

Table 7-31 Parameters

Data direction	Parameter name	Description
[in]	pin	Input pin mask

7.2.10.7. Function system_get_pin_wakeup_cause()

Check whether any of the enabled wake up pins are active and caused the wakeup.

```
uint16_t system_get_pin_wakeup_cause( void )
```

Check whether any of the enabled wake up pins are active and caused the wakeup from backup sleep mode when exiting backup mode.

Returns
Pin mask, the corresponding pin is active when its pin mask is 1.

7.2.11. System Debugger

7.2.11.1. Function system_is_debugger_present()

Check if debugger is present.

```
bool system_is_debugger_present( void )
```

Check if debugger is connected to the onboard debug system (DAP).

Returns
A bool identifying if a debugger is present.

Table 7-32 Return Values

Return value	Description
true	Debugger is connected to the system
false	Debugger is not connected to the system

7.2.12. System Identification

7.2.12.1. Function system_get_device_id()

Retrieve the device identification signature.

```
uint32_t system_get_device_id( void )
```

Retrieves the signature of the current device.

Returns
Device ID signature as a 32-bit integer.

7.2.13. System Initialization

7.2.13.1. Function system_init()

Initialize system.

```
void system_init( void )
```

This function will call the various initialization functions within the system namespace. If a given optional system module is not available, the associated call will effectively be a NOP (No Operation).

Currently the following initialization functions are supported:

- System clock initialization (via the SYSTEM CLOCK sub-module)
- Board hardware initialization (via the Board module)
- Event system driver initialization (via the EVSYS module)
- External Interrupt driver initialization (via the EXTINT module)

7.3. Enumeration Definitions

7.3.1. Enum system_backup_pin

List of Backup input and output pins. If enabled (system_backup_pin_output_enable), the pins can be driven by the SUPC.

Table 7-33 Members

Enum value	Description
SYSTEM_BACKUP_PIN_PSOK	Power Supply OK status pin
SYSTEM_BACKUP_PIN_OUT_0	Backup output pin 0
SYSTEM_BACKUP_PIN_OUT_1	Backup output pin 1

7.3.2. Enum system_battery_power_switch

Enum for Battery power switch modes.

Table 7-34 Members

Enum value	Description
SYSTEM_BATTERY_POWER_SWITCH_NONE	The backup domain is always supplied by main power
SYSTEM_BATTERY_POWER_SWITCH_AUTOMATIC	The power switch is handled by the automatic power switch
SYSTEM_BATTERY_POWER_SWITCH_FORCED	The backup domain is always supplied by battery backup power
SYSTEM_BATTERY_POWER_SWITCH_BOD33	The power switch is handled by the BOD33

7.3.3. Enum system_linked_power_domain

List of linked power domains. Power domains can be linked to each other. It allows a power domain (PDn) to be kept in active state if the inferior power domain (PDn-1) is in active state too.

Table 7-35 Members

Enum value	Description
SYSTEM_LINKED_POWER_DOMAIN_DEFAULT	Power domains PD0/PD1/PD2 are not linked
SYSTEM_LINKED_POWER_DOMAIN_PD01	Power domains PD0 and PD1 are linked
SYSTEM_LINKED_POWER_DOMAIN_PD12	Power domains PD1 and PD2 are linked
SYSTEM_LINKED_POWER_DOMAIN_PD012	All Power domains are linked

7.3.4. Enum system_performance_level

List of performance levels. Performance level technique consists of adjusting the regulator output voltage to reduce power consumption.

Table 7-36 Members

Enum value	Description
SYSTEM_PERFORMANCE_LEVEL_0	Performance level 0
SYSTEM_PERFORMANCE_LEVEL_2	Performance level 2

7.3.5. Enum system_power_domain

List of power domains. Power domain gating technique consists of turning on or off power domain voltage to save power while keeping other domains powered up.

Table 7-37 Members

Enum value	Description
SYSTEM_POWER_DOMAIN_DEFAULT	All power domains switching are handled by hardware
SYSTEM_POWER_DOMAIN_PD0	Power domain 0 (PD0) is forced ACTIVE
SYSTEM_POWER_DOMAIN_PD01	Power domain 0 and 1 (PD0 and PD1) are forced ACTIVE
SYSTEM_POWER_DOMAIN_PD012	All power domains are forced ACTIVE

7.3.6. Enum system_ram_back_bias_mode

List of RAM back bias modes. By default, in standby sleep mode, RAM is in low power mode (back biased) if its power domain is in retention state. This behavior can be changed by configuring the Back Bias bit groups in STDBYCFG(STDBYCFG.BBIASxx).

Table 7-38 Members

Enum value	Description
SYSTEM_RAM_BACK_BIAS_RETENTION	Retention Back biasing mode
SYSTEM_RAM_BACK_BIAS_STANDBY	Standby Back Biasing mode
SYSTEM_RAM_BACK_BIAS_STANDBY_OFF	Standby OFF mode
SYSTEM_RAM_BACK_BIAS_OFF	Always OFF mode

7.3.7. Enum system_reset_backup_exit_source

List of possible backup exit source.

Table 7-39 Members

Enum value	Description
SYSTEM_RESET_BACKKUP_EXIT_EXTWAKE	The backup exit source was external wakeup
SYSTEM_RESET_BACKKUP_EXIT_RTC	The backup exit source was RTC interrupt
SYSTEM_RESET_BACKKUP_EXIT_BBPS	The backup exit source was battery backup power switch

7.3.8. Enum system_reset_cause

List of possible reset causes of the system.

Table 7-40 Members

Enum value	Description
SYSTEM_RESET_CAUSE_BACKUP	The system was last reset by a backup reset
SYSTEM_RESET_CAUSE_SOFTWARE	The system was last reset by a software reset
SYSTEM_RESET_CAUSE_WDT	The system was last reset by the watchdog timer
SYSTEM_RESET_CAUSE_EXTERNAL_RESET	The system was last reset because the external reset line was pulled low
SYSTEM_RESET_CAUSE_BOD33	The system was last reset by the BOD33
SYSTEM_RESET_CAUSE_BOD12	The system was last reset by the BOD12
SYSTEM_RESET_CAUSE_POR	The system was last reset by the POR (Power on reset)

7.3.9. Enum system_sleepmode

List of available sleep modes in the device. A table of clocks available in different sleep modes can be found in Sleep Modes.

Table 7-41 Members

Enum value	Description
SYSTEM_SLEEPMODE_IDLE	IDLE sleep mode
SYSTEM_SLEEPMODE_STANDBY	STANDBY sleep mode
SYSTEM_SLEEPMODE_BACKUP	BACKUP sleep mode
SYSTEM_SLEEPMODE_OFF	OFF sleep mode

7.3.10. Enum system_voltage_reference

List of available voltage references (VREF) that may be used within the device.

Table 7-42 Members

Enum value	Description
SYSTEM_VOLTAGE_REFERENCE_TEMPSENSE	Temperature sensor voltage reference
SYSTEM_VOLTAGE_REFERENCE_OUTPUT	Voltage reference output

7.3.11. Enum system_voltage_references_sel

Voltage references selection.

Table 7-43 Members

Enum value	Description
SYSTEM_VOLTAGE_REFERENCE_1V0	1.0V voltage reference typical value
SYSTEM_VOLTAGE_REFERENCE_1V1	1.1V voltage reference typical value
SYSTEM_VOLTAGE_REFERENCE_1V2	1.2V voltage reference typical value
SYSTEM_VOLTAGE_REFERENCE_1V25	1.25V voltage reference typical value
SYSTEM_VOLTAGE_REFERENCE_2V0	2.0V voltage reference typical value
SYSTEM_VOLTAGE_REFERENCE_2V2	2.2V voltage reference typical value
SYSTEM_VOLTAGE_REFERENCE_2V4	2.4V voltage reference typical value
SYSTEM_VOLTAGE_REFERENCE_2V5	2.5V voltage reference typical value

7.3.12. Enum system_voltage_regulator_low_power_efficiency

Low power mode efficiency.

Table 7-44 Members

Enum value	Description
SYSTEM_VOLTAGE_REGULATOR_LOW_POWER_EFFICIENCY_DEFAULT	The voltage regulator in Low power mode has the default efficiency and support the whole VDD range (1.62V to 3.6V)
SYSTEM_VOLTAGE_REGULATOR_LOW_POWER_EFFICIENCY_HIGHTEST	The voltage regulator in Low power mode has the highest efficiency and support the limited VDD range (2.5V to 3.6V)

7.3.13. Enum system_voltage_regulator_sel

Voltage regulators selection. In active mode, the voltage regulator can be chosen on the fly between a LDO or a Buck converter.

Table 7-45 Members

Enum value	Description
SYSTEM_VOLTAGE_REGULATOR_LDO	The voltage regulator in active mode is a LDO voltage regulator
SYSTEM_VOLTAGE_REGULATOR_BUCK	The voltage regulator in active mode is a buck converter

7.3.14. Enum system_wakeup_debounce_count

Wakeup debounce counter value when waking up by external wakeup pin from backup mode.

Table 7-46 Members

Enum value	Description
SYSTEM_WAKEUP_DEBOUNCE_OFF	No debouncing
SYSTEM_WAKEUP_DEBOUNCE_2CK32	Input pin shall be active for at least two 32KHz clock periods
SYSTEM_WAKEUP_DEBOUNCE_3CK32	Input pin shall be active for at least three 32KHz clock periods
SYSTEM_WAKEUP_DEBOUNCE_32CK32	Input pin shall be active for at least 32 32KHz clock periods
SYSTEM_WAKEUP_DEBOUNCE_512CK32	Input pin shall be active for at least 512 32KHz clock periods
SYSTEM_WAKEUP_DEBOUNCE_4096CK32	Input pin shall be active for at least 4096 32KHz clock periods
SYSTEM_WAKEUP_DEBOUNCE_32768CK32	Input pin shall be active for at least 32768 32KHz clock periods

8. Examples for Power Driver

This is a list of the available Quick Start Guides (QSGs) and example applications. QSGs are simple examples with step-by-step instructions to configure and use this driver in a selection of use cases. Note that a QSG can be compiled as a standalone application or be added to the user application.

- Quick Start Guide for Power Driver

8.1. Quick Start Guide for Power Driver

List of supported boards:

- SAM L21 Xplained Pro

This example demonstrates how to use the power driver. BUTTON0 is used to wake up the system from the standby sleep mode and as an external wakeup pin to wake up the system from the backup sleep mode. The wakeup pin level is low. The I/O pins PB22/PB23 are used as GCLK0/GCLK1 outputs so that an oscilloscope can be used to monitor the clock frequencies.

After power-on-reset (POR), GCLK0 and GCLK1 runs at 4MHz and LED0 is turned on. After one second, LED0 is turned off and the system enters standby sleep mode. BUTTON0 can then be used to wake up the system. After the system wakeup, LED0 is turned on, the performance level is switched to PL2, and the GCLK0 is increased to 48MHz. Further LED0 toggles two times and is turned off before the system enters BACKUP.

When BUTTON0 pushes, it connects to low level, system wakes up from the backup sleep mode, LED0 toggles four times. GCLK0/GCLK1 are running at 4MHz.

8.1.1. Quick Start

8.1.1.1. Prerequisites

There are no prerequisites for this use case.

8.1.1.2. Code

Copy-paste the following setup code to your user application:

```
static void performance_level_switch_test(void)
{
    struct system_gclk_gen_config gclk_conf;

    /* Switch to PL2 */
    system_switch_performance_level(SYSTEM_PERFORMANCE_LEVEL_2);
    system_flash_set_waitstates(2);

    /* Switch GCLK0 to 48MHz */
    system_gclk_gen_get_config_defaults(&gclk_conf);
    gclk_conf.source_clock = SYSTEM_CLOCK_SOURCE_DFLL;
    gclk_conf.division_factor = 1;
    gclk_conf.run_in_standby  = false;
    gclk_conf.output_enable   = true;
    system_gclk_gen_set_config(GCLK_GENERATOR_0, &gclk_conf);
}


static void config_clock_output_and_extwake_pin(void)
```

```
{
    struct system_pinmux_config pin_conf;
    system_pinmux_get_config_defaults(&pin_conf);

    pin_conf.mux_position = CONF_GCLK0_OUTPUT_PINMUX;
    pin_conf.direction    = SYSTEM_PINMUX_PIN_DIR_OUTPUT;
    system_pinmux_pin_set_config(CONF_GCLK0_OUTPUT_PIN, &pin_conf);
    pin_conf.mux_position = CONF_GCLK1_OUTPUT_PINMUX;
    system_pinmux_pin_set_config(CONF_GCLK1_OUTPUT_PIN, &pin_conf);

    pin_conf.direction = SYSTEM_PINMUX_PIN_DIR_INPUT;
    pin_conf.input_pull = SYSTEM_PINMUX_PIN_PULL_UP;
    pin_conf.mux_position = CONF_EXT_WAKEUP_PINMUX;
    system_pinmux_pin_set_config(CONF_EXT_WAKEUP_PIN, &pin_conf);
}


static void configure_extint_channel(void)
{

    struct extint_chan_conf config_extint_chan;
    extint_chan_get_config_defaults(&config_extint_chan);
    config_extint_chan.gpio_pin            = BUTTON_0_EIC_PIN;
    config_extint_chan.gpio_pin_mux        = BUTTON_0_EIC_MUX;
    config_extint_chan.gpio_pin_pull       = EXTINT_PULL_UP;
    config_extint_chan.detection_criteria = EXTINT_DETECT_BOTH;
    extint_chan_set_config(BUTTON_0_EIC_LINE, &config_extint_chan);

extint_chan_enable_callback(BUTTON_0_EIC_LINE,EXTINT_CALLBACK_TYPE_DETECT);
    system_interrupt_enable_global();
    while (extint_chan_is_detected(BUTTON_0_EIC_LINE)) {
        extint_chan_clear_detected(BUTTON_0_EIC_LINE);
    }
}

static void led_toggle_indication(uint32_t count)
{
    for (uint32_t i = 0; i < count; i++) {
        port_pin_set_output_level(LED_0_PIN, LED_0_ACTIVE);
        delay_ms(200);
        port_pin_set_output_level(LED_0_PIN, LED_0_INACTIVE);
        delay_ms(200);
    }
}
```

8.1.1.3. Workflow

1. Switch performance level to PL2.

```
static void performance_level_switch_test(void)
{
    struct system_gclk_gen_config gclk_conf;

    /* Switch to PL2 */
    system_switch_performance_level(SYSTEM_PERFORMANCE_LEVEL_2);
    system_flash_set_waitstates(2);

    /* Switch GCLK0 to 48MHz */
    system_gclk_gen_get_config_defaults(&gclk_conf);
    gclk_conf.source_clock = SYSTEM_CLOCK_SOURCE_DFLL;
    gclk_conf.division_factor = 1;
    gclk_conf.run_in_standby  = false;
```

```
    gclk_conf.output_enable    = true;
    system_gclk_gen_set_config(GCLK_GENERATOR_0, &gclk_conf);
}
```

2. Configure GCLK0/GCLK1 output pin and extwakeup pin.

```
static void config_clock_output_and_extwake_pin(void)
{
    struct system_pinmux_config pin_conf;
    system_pinmux_get_config_defaults(&pin_conf);

    pin_conf.mux_position = CONF_GCLK0_OUTPUT_PINMUX;
    pin_conf.direction    = SYSTEM_PINMUX_PIN_DIR_OUTPUT;
    system_pinmux_pin_set_config(CONF_GCLK0_OUTPUT_PIN, &pin_conf);
    pin_conf.mux_position = CONF_GCLK1_OUTPUT_PINMUX;
    system_pinmux_pin_set_config(CONF_GCLK1_OUTPUT_PIN, &pin_conf);

    pin_conf.direction = SYSTEM_PINMUX_PIN_DIR_INPUT;
    pin_conf.input_pull = SYSTEM_PINMUX_PIN_PULL_UP;
    pin_conf.mux_position = CONF_EXT_WAKEUP_PINMUX;
    system_pinmux_pin_set_config(CONF_EXT_WAKEUP_PIN, &pin_conf);
}
```

3. Config external interrupt.

```
static void configure_extint_channel(void)
{

    struct extint_chan_conf config_extint_chan;
    extint_chan_get_config_defaults(&config_extint_chan);
    config_extint_chan.gpio_pin           = BUTTON_0_EIC_PIN;
    config_extint_chan.gpio_pin_mux       = BUTTON_0_EIC_MUX;
    config_extint_chan.gpio_pin_pull      = EXTINT_PULL_UP;
    config_extint_chan.detection_criteria = EXTINT_DETECT_BOTH;
    extint_chan_set_config(BUTTON_0_EIC_LINE, &config_extint_chan);

extint_chan_enable_callback(BUTTON_0_EIC_LINE,EXTINT_CALLBACK_TYPE_DETE
CT);
    system_interrupt_enable_global();
    while (extint_chan_is_detected(BUTTON_0_EIC_LINE)) {
        extint_chan_clear_detected(BUTTON_0_EIC_LINE);
    }
}
```

8.1.2. Use Case

8.1.2.1. Code

Copy-paste the following code to your user application:

```
/* Check if the RESET is caused by external wakeup pin */
if (system_get_reset_cause() == SYSTEM_RESET_CAUSE_BACKUP
    && system_get_backup_exit_source() == SYSTEM_RESET_BACKKUP_EXIT_EXTWAKE
    && (system_get_pin_wakeup_cause() & (1 << CONF_EXT_WAKEUP_PIN))){
    system_init();
    delay_init();
    config_clock_output_and_extwake_pin();
    /* Now GCLK0/GCLK1 are running at 4MHz , using LED0 ON/OFF as an
indication */
    led_toggle_indication(4);
    port_pin_set_output_level(LED_0_PIN, LED_0_ACTIVE);
    while(1);
}
```

```
/* Disable I/O retention*/
system_io_retension_disable();

system_init();
delay_init();

port_pin_set_output_level(LED_0_PIN, LED_0_ACTIVE);

/* Config GCLK0/GCLK1 output pin and extwakeup pin */
config_clock_output_and_extwake_pin();

/* Config external interrupt for wakeup system from standby mode*/
configure_extint_channel();
delay_s(1);

/* Turn off LED0 before enter standby mode */
port_pin_set_output_level(LED_0_PIN, LED_0_INACTIVE);

/* Set and enter standby mode, using default stanby config */
system_set_sleepmode(SYSTEM_SLEEPMODE_STANDBY);
system_sleep();

/* Detect BUTTON pressed and system  wakeup from standby mode,
    turn on led */
port_pin_set_output_level(LED_0_PIN, LED_0_ACTIVE);

/* Disable external interrupt to avoid any impact */
extint_chan_disable_callback(BUTTON_0_EIC_LINE,
            EXTINT_CALLBACK_TYPE_DETECT);
extint_chan_clear_detected(BUTTON_0_EIC_LINE);

/* Switch GCLK0 to 48MHz */
performance_level_switch_test();

/* GCLK0 is runing at 48MHz and GCLK1 is running at 4MHz ,
    use led ON/OFF as an indication */
led_toggle_indication(2);


/* Set external wakeup pin polarity */
system_set_pin_wakeup_polarity_low(1<<CONF_EXT_WAKEUP_PIN);

/* Set external wakeup detector */
system_enable_pin_wakeup(1<<CONF_EXT_WAKEUP_PIN);
system_set_pin_wakeup_debounce_counter(SYSTEM_WAKEUP_DEBOUNCE_2CK32);

/* Enter BACKUP mode */
system_set_sleepmode(SYSTEM_SLEEPMODE_BACKUP);
system_sleep();

/* Now system is in BACKUP mode and wait for extwakeup pin to low */
```

8.1.2.2. Workflow

1. Check if the RESET is caused by external wakeup pin.

```
/* Check if the RESET is caused by external wakeup pin */
if (system_get_reset_cause() == SYSTEM_RESET_CAUSE_BACKUP
    && system_get_backup_exit_source() ==
```

```
SYSTEM_RESET_BACKKUP_EXIT_EXTWAKE
    && (system_get_pin_wakeup_cause() & (1 << CONF_EXT_WAKEUP_PIN))){
    system_init();
    delay_init();
    config_clock_output_and_extwake_pin();
    /* Now GCLK0/GCLK1 are running at 4MHz , using LED0 ON/OFF as an
indication */
    led_toggle_indication(4);
    port_pin_set_output_level(LED_0_PIN, LED_0_ACTIVE);
    while(1);
}
```

2. Check the standby mode and the backup sleep mode.

```

    /* Disable I/O retention*/
    system_io_retension_disable();

    system_init();
    delay_init();

    port_pin_set_output_level(LED_0_PIN, LED_0_ACTIVE);

    /* Config GCLK0/GCLK1 output pin and extwakeup pin */
    config_clock_output_and_extwake_pin();

    /* Config external interrupt for wakeup system from standby mode*/
    configure_extint_channel();
    delay_s(1);

    /* Turn off LED0 before enter standby mode */
    port_pin_set_output_level(LED_0_PIN, LED_0_INACTIVE);

    /* Set and enter standby mode, using default stanby config */
    system_set_sleepmode(SYSTEM_SLEEPMODE_STANDBY);
    system_sleep();

    /* Detect BUTTON pressed and system  wakeup from standby mode,
        turn on led */
    port_pin_set_output_level(LED_0_PIN, LED_0_ACTIVE);

    /* Disable external interrupt to avoid any impact */
    extint_chan_disable_callback(BUTTON_0_EIC_LINE,
                EXTINT_CALLBACK_TYPE_DETECT);
    extint_chan_clear_detected(BUTTON_0_EIC_LINE);

    /* Switch GCLK0 to 48MHz */
    performance_level_switch_test();

    /* GCLK0 is runing at 48MHz and GCLK1 is running at 4MHz ,
        use led ON/OFF as an indication */
    led_toggle_indication(2);


    /* Set external wakeup pin polarity */
    system_set_pin_wakeup_polarity_low(1<<CONF_EXT_WAKEUP_PIN);

    /* Set external wakeup detector */
    system_enable_pin_wakeup(1<<CONF_EXT_WAKEUP_PIN);

system_set_pin_wakeup_debounce_counter(SYSTEM_WAKEUP_DEBOUNCE_2CK32);

    /* Enter BACKUP mode */
```

```
    system_set_sleepmode(SYSTEM_SLEEPMODE_BACKUP);
    system_sleep();

    /* Now system is in BACKUP mode and wait for extwakeup pin to low
*/
```

9. Extra Information for SYSTEM Driver

9.1. Acronyms

Below is a table listing the acronyms used in this module, along with their intended meanings.

Acronym	Definition
PM	Power Manager
SUPC	Supply Controller
RSTC	Reset Controller

9.2. Dependencies

This driver has the following dependencies:

- None

9.3. Errata

There are no errata related to this driver.

9.4. Module History

An overview of the module history is presented in the table below, with details on the enhancements and fixes made to the module since its first release. The current version of this corresponds to the newest version in the table.

Changelog
Initial Release

10. Document Revision History

Doc. Rev.	Date	Comments
42449A	07/2015	Initial document release

Atmel | Enabling Unlimited Possibilities®

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