APPLICATION NOTE

AT06861: SAM Supply Controller (SUPC)

ASF PROGRAMMERS MANUAL

SAM Supply Controller (SUPC)

This driver for SAM devices provides an interface for the configuration and management of the device's Supply Controller. The Supply Controller controls the supply voltages of the system, the selection of the Slow Clock (SCLK) source and manages the Backup Low Power Mode and the Core's wake-up source(s).

The following peripherals are used by this module:

- SUPC (Supply Controller)

The outline of this documentation is as follows:

Table of Contents

Software License

1. Prerequisites

There are no prerequisites for this module.

2. Module Overview

The Supply Controller controls the Embedded Voltage Regulators used by the Core Power Supply and Backup Low Power Mode.

In Backup Low Power Mode the current consumption is reduced to a few microamps for Backup power retention. Exit from this mode can be achieved via multiple wake-up sources.

The Supply Controller also generates the Slow Clock (SCLK) by selecting either the 22-42kHz Low Power RC oscillator or the 32kHz Low Power Crystal oscillator.

2.1 Backup Low Power Mode

The purpose of Backup Low Power Mode is to achieve the lowest power consumption possible in a system that wakes up periodically in order to perform tasks that do not require a fast startup time.

The supply controller, zero-power power-on reset, RTT, RTC, backup registers, and 32kHz oscillator (RC or crystal oscillator selected by software) are all running whilst the regulator and core supply are off.

3. Special Considerations

The time between a Supply Monitor interrupt being raised and the failure of VDDIO is system specific; care must be taken that all critical Supply Monitor detection events can complete within the amount of time available.

For more information on the Supply Monitor refer to the Supply Monitor section of the device's datasheet.

4. Extra Information

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

- Acronyms
- Dependencies
- Errata
- Module History

5. Examples

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

6. API Overview

6.1 Function Definitions

6.1.1 Function supc_disable_backup_power_on_reset()

Disable the Backup Area Power-On Reset.

```
void supc_disable_backup_power_on_reset(
  Supc * p_supc)
```

Note This function is only available on SAM4C devices.

Table 6-1. Parameters

Data direction	Parameter name	Description
[in, out]	p_supc	Module hardware register base address pointer

6.1.2 Function supc_disable_brownout_detector()

Disable the core brownout detector.

```
void supc_disable_brownout_detector(
  Supc * p_supc)
```

Table 6-2. Parameters

Data direction	Parameter name	Description
[in, out]	p_supc	Module hardware register base address pointer

6.1.3 Function supc_disable_brownout_reset()

Disable the assertion of core reset signal when a brownout detection occurs.

```
void supc_disable_brownout_reset(
  Supc * p_supc)
```

Table 6-3. Parameters

Data direction	Parameter name	Description
[in, out]	p_supc	Module hardware register base address pointer

6.1.4 Function supc_disable_monitor_interrupt()

Disable the assertion of SUPC interrupt signal when a Supply Monitor detection occurs.

```
void supc_disable_monitor_interrupt(
```

```
  Supc * p_supc)
```

Table 6-4. Parameters

Data direction	Parameter name	Description
[in, out]	p_supc	Module hardware register base address pointer

6.1.5 Function supc_disable_monitor_reset()

Disable the assertion of core reset signal when a Supply Monitor detection occurs.

```
void supc_disable_monitor_reset(
  Supc * p_supc)
```

Table 6-5. Parameters

Data direction	Parameter name	Description
[in, out]	p_supc	Module hardware register base address pointer

6.1.6 Function supc_disable_voltage_regulator()

Disable the internal voltage regulator to supply VDDCORE with an external supply.

```
void supc_disable_voltage_regulator(
  Supc * p_supc)
```

Note This function is not available on SAMG devices.

Table 6-6. Parameters

Data direction	Parameter name	Description
[in, out]	p_supc	Module hardware register base address pointer

6.1.7 Function supc_enable_backup_mode()

Switch off the voltage regulator to put the device in backup mode.

```
void supc_enable_backup_mode(
  Supc * p_supc)
```

Note This function is not available on SAMG devices.

Table 6-7. Parameters

Data direction	Parameter name	Description
[out]	p_supc	Module hardware register base address pointer

6.1.8 Function supc_enable_backup_power_on_reset()

Enable the Backup Area Power-On Reset.

```
void supc_enable_backup_power_on_reset(
  Supc * p_supc)
```

Note This function is only available on SAM4C devices.

Table 6-8. Parameters

Data direction	Parameter name	Description
[in, out]	p_supc	Module hardware register base address pointer

6.1.9 Function supc_enable_brownout_detector()

Enable the core brownout detector.

```
void supc_enable_brownout_detector(
  Supc * p_supc)
```

Table 6-9. Parameters

Data direction	Parameter name	Description
[in, out]	p_supc	Module hardware register base address pointer

6.1.10 Function supc_enable_brownout_reset()

Enable the assertion of core reset signal when a brownout detection occurs.

```
void supc_enable_brownout_reset(
  Supc * p_supc)
```

Table 6-10. Parameters

Data direction	Parameter name	Description
[in, out]	p_supc	Module hardware register base address pointer

6.1.11 Function supc_enable_monitor_interrupt()

Enable the assertion of SUPC interrupt signal when a Supply Monitor detection occurs.

```
void supc_enable_monitor_interrupt(
  Supc * p_supc)
```

Table 6-11. Parameters

Data direction	Parameter name	Description
[in, out]	p_supc	Module hardware register base address pointer

6.1.12 Function supc_enable_monitor_reset()

Enable the assertion of the core reset signal when a Supply Monitor detection occurs.

```
void supc_enable_monitor_reset(
  Supc * p_supc)
```

Table 6-12. Parameters

Data direction	Parameter name	Description
[in, out]	p_supc	Module hardware register base address pointer

6.1.13 Function supc_enable_voltage_regulator()

Enable the internal voltage regulator.

```
void supc_enable_voltage_regulator(
  Supc * p_supc)
```

Note This function is not available on SAMG devices.

Table 6-13. Parameters

Data direction	Parameter name	Description
[in, out]	p_supc	Module hardware register base address pointer

6.1.14 Function supc_get_slcd_power_mode()

Get the LCD power mode.

```
enum slcdc_power_mode supc_get_slcd_power_mode(
  Supc * p_supc)
```

Note This function is only available on SAM4C devices.

Table 6-14. Parameters

Data direction	Parameter name	Description
[in]	p_supc	Module hardware register base address pointer

Returns The LCD Voltage Regulator operating mode.

6.1.15 Function supc_get_status()

Get the Supply Controller status.

```
uint32_t supc_get_status(
  Supc * p_supc)
```

Table 6-15. Parameters

Data direction	Parameter name	Description
[in]	p_supc	Module hardware register base address pointer

Returns The status of the Supply Controller.

6.1.16 Function supc_set_monitor_sampling_period()

Set the Supply Monitor sampling period.

```
void supc_set_monitor_sampling_period(
  Supc * p_supc,
  uint32_t ul_period)
```

Table 6-16. Parameters

Data direction	Parameter name	Description
[in, out]	p_supc	Module hardware register base address pointer
[in]	ul_period	Supply Monitor sampling period

Where the input parameter *ul_period* can be one of the following:

Parameter Value	Description
SUPC_SMMR_SMSMPL_SMD	Supply Monitor disabled
SUPC_SMMR_SMSMPL_CSM	Continuous Supply Monitor
SUPC_SMMR_SMSMPL_32SLCK	Supply Monitor enabled one SLCK period every 32 SLCK periods
SUPC_SMMR_SMSMPL_256SLCK	Supply Monitor enabled one SLCK period every 256 SLCK periods
SUPC_SMMR_SMSMPL_2048SLCK	Supply Monitor enabled one SLCK period every 2,048 SLCK periods

6.1.17 Function supc_set_monitor_threshold()

Set the Supply Monitor threshold.

```
void supc_set_monitor_threshold(
```

```
  Supc * p_supc,
  uint32_t ul_threshold)
```

Table 6-17. Parameters

Data direction	Parameter name	Description
[in, out]	p_supc	Module hardware register base address pointer
[in]	ul_threshold	Supply monitor threshold (between 1.9V and 3.4V) Refer to the electrical characteristics section in the device's datasheet for suitable values.

6.1.18 Function supc_set_slcd_power_mode()

Set the LCD power mode.

```
void supc_set_slcd_power_mode(
  Supc * p_supc,
  enum slcdc_power_mode mode)
```

Note This function is only available on SAM4C devices.

Table 6-18. Parameters

Data direction	Parameter name	Description
[in, out]	p_supc	Module hardware register base address pointer
[in]	mode	The LCD Voltage Regulator operating mode

6.1.19 Function supc_set_slcd_vol()

Set the LCD Voltage Regulator output.

```
void supc_set_slcd_vol(
  Supc * p_supc,
  uint32_t vol)
```

Note This function is only available on SAM4C devices.

Table 6-19. Parameters

Data direction	Parameter name	Description
[in, out]	p_supc	Module hardware register base address pointer
[in]	vol	The LCD Voltage Regulator output. Refer to the "LCD Voltage Regulator and LCD Output Buffers" section located in "Electrical

Data direction	Parameter name	Description
		Characteristics" of the device's datasheet for values.

6.1.20 Function supc_set_wakeup_inputs()

Set the Supply Controller wake-up inputs.

```
void supc_set_wakeup_inputs(
  Supc * p_supc,
  uint32_t ul_inputs,
  uint32_t ul_transition)
```

Note This function is not available on SAMG devices.

Table 6-20. Parameters

Data direction	Parameter name	Description
[out]	p_supc	Module hardware register base address pointer
[in]	ul_inputs	Bitmask of wake-up inputs that can force the wake-up of the core power supply.
[in]	ul_transition	Bitmask of level transition of the corresponding wake-up inputs • 0 means a high-to-low level transition forces the wake-up of core power supply • 1 means a low-to-high level transition forces the wake-up of core power supply

Where input parameter *ul_inputs* is a bitmask containing one or more of the following:

Parameter Value	Description
SUPC_WUIR_WKUPEN0	wake-up Input Enable 0
SUPC_WUIR_WKUPEN1	wake-up Input Enable 1
SUPC_WUIR_WKUPEN2	wake-up Input Enable 2
SUPC_WUIR_WKUPEN3	wake-up Input Enable 3
SUPC_WUIR_WKUPEN4	wake-up Input Enable 4
SUPC_WUIR_WKUPEN5	wake-up Input Enable 5
SUPC_WUIR_WKUPEN6	wake-up Input Enable 6
SUPC_WUIR_WKUPEN7	wake-up Input Enable 7
SUPC_WUIR_WKUPEN8	wake-up Input Enable 8
SUPC_WUIR_WKUPEN9	wake-up Input Enable 9
SUPC_WUIR_WKUPEN10	wake-up Input Enable 10
SUPC_WUIR_WKUPEN11	wake-up Input Enable 11

Parameter Value	Description
SUPC_WUIR_WKUPEN12	wake-up Input Enable 12
SUPC_WUIR_WKUPEN13	wake-up Input Enable 13
SUPC_WUIR_WKUPEN14	wake-up Input Enable 14
SUPC_WUIR_WKUPEN15	wake-up Input Enable 15

Where input parameter *ul_transition* is a bitmask containing one or more of the following:

Parameter Value	Description
SUPC_WUIR_WKUPT0	wake-up Input Type 0
SUPC_WUIR_WKUPT1	wake-up Input Type 1
SUPC_WUIR_WKUPT2	wake-up Input Type 2
SUPC_WUIR_WKUPT3	wake-up Input Type 3
SUPC_WUIR_WKUPT4	wake-up Input Type 4
SUPC_WUIR_WKUPT5	wake-up Input Type 5
SUPC_WUIR_WKUPT6	wake-up Input Type 6
SUPC_WUIR_WKUPT7	wake-up Input Type 7
SUPC_WUIR_WKUPT8	wake-up Input Type 8
SUPC_WUIR_WKUPT9	wake-up Input Type 9
SUPC_WUIR_WKUPT10	wake-up Input Type 10
SUPC_WUIR_WKUPT11	wake-up Input Type 11
SUPC_WUIR_WKUPT12	wake-up Input Type 12
SUPC_WUIR_WKUPT13	wake-up Input Type 13
SUPC_WUIR_WKUPT14	wake-up Input Type 14
SUPC_WUIR_WKUPT15	wake-up Input Type 15

6.1.21 Function supc_set_wakeup_mode()

Set the Supply Controller wake-up mode.

```
void supc_set_wakeup_mode(
  Supc * p_supc,
  uint32_t ul_mode)
```

Note This function is not available on SAMG devices.

Table 6-21. Parameters

Data direction	Parameter name	Description
[out]	p_supc	Module hardware register base address pointer
[in]	ul_mode	Bitmask of wake-up mode (refer to "Supply Controller Wake-up Mode Register" in the SUPC section of the device datasheet for more details).

Where input parameter *ul_mode* is a bitmask containing one or more of the following:

Parameter Value	Description
SUPC_WUMR_FWUPEN	Force Wake-up Enable (SAM4C and SAM4E only)
SUPC_WUMR_SMEN	Supply Monitor Wake-up Enable
SUPC_WUMR_RTTEN	Real Time Timer Wake-up Enable
SUPC_WUMR_RTCEN	Real Time Clock Wake-up Enable
SUPC_WUMR_LPDBCEN0	Low power Debouncer ENable WKUP0
SUPC_WUMR_LPDBCEN1	Low power Debouncer ENable WKUP1
SUPC_WUMR_LPDBCCLR	Low power Debouncer Clear

Note User applications that require low power wake-up debouncing or a Tamper detect timestamp (e.g. SAM4C) should refer to the device datasheet for more information.

6.1.22 Function supc_switch_sclk_to_32kxtal()

Switch the Slow Clock (SCLK) source selection to external 32kHz (XTAL or Bypass) oscillator. This function disables the PLLs.

```
void supc_switch_sclk_to_32kxtal(
  Supc * p_supc,
  uint32_t ul_bypass)
```

Note Switching SCLK back to the 32kHz RC oscillator is only possible by shutting down the VDDIO power supply.

Table 6-22. Parameters

Data direction	Parameter name	Description
[in, out]	p_supc	Module hardware register base address pointer
[in]	ul_bypass	0 for XTAL, 1 for bypass

6.2 Enumeration Definitions

6.2.1 Enum slcdc_power_mode

Note This enumerated type is only available on SAM4C devices.

Table 6-23. Members

Enum value	Description
SLCDC_POWER_MODE_LCDOFF	The internal supply source and the external supply source are both deselected.

Enum value	Description
SLCDC_POWER_MODE_LCDON_EXTVR	The external supply source for the LCD is selected.
SLCDC_POWER_MODE_LCDON_INVR	The internal supply source for the LCD is selected.

7. Extra Information for Supply Controller Driver

7.1 Acronyms

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

Acronym	Definition
BOD	Brownout detector
LCD	Liquid Crystal Display
PLL	Phase Locked Loop
QSG	Quick Start Guide
RTC	Real-Time Clock
RTT	Real-Time Timer
SCLK	Slow Clock
SLCDC	Segment Liquid Crystal Display Controller

7.2 Dependencies

This driver has the following dependencies:

- None

7.3 Errata

There are no errata related to this driver.

7.4 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 document release

8. Examples for Supply Controller Driver

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

- Quick Start Guide for Supply Controller - RTC Wakeup

8.1 Quick Start Guide for Supply Controller - RTC Wakeup

In this case we apply the following configuration:

- Set the RTC Alarm for 10 seconds into the future
- Enable the RTC Alarm as a wake-up source

8.1.1 Setup

8.1.1.1 Prerequisites

- RTC year, month, day, and week must be configured
- RTC hour, minute, and second must be configured

8.1.1.2 Code Example

Copy-paste the following setup code towards the top of your application file:

```
struct rtc_date_config {
    uint32_t ul_year;
    uint32_t ul_month;
    uint32_t ul_day;
    uint32_t ul_week;
};

struct rtc_time_config {
    uint32_t ul_hour;
    uint32_t ul_minute;
    uint32_t ul_second;
};

struct rtc_config {
    struct rtc_date_config date;
    struct rtc_time_config time;
    };

#define TIME_INIT_YEAR (2014)
#define TIME_INIT_MONTH (1)
#define TIME_INIT_DAY (1)
#define TIME_INIT_WEEK (1)
#define TIME_INIT_HOUR (12)
#define TIME_INIT_MINUTE (0)
#define TIME_INIT_SECOND (0)

#define ALARM_AT_MONTH (1)
#define ALARM_AT_DAY (1)
#define ALARM_AT_HOUR (12)
#define ALARM_AT_MINUTE (0)
#define ALARM_AT_SECOND (20)

enum rtc_config_error {
    RTC_CONFIG_ERROR_NONE = 0,
```

```
    RTC_CONFIG_ERROR_INVALID_TIME = 1,
    RTC_CONFIG_ERROR_INVALID_DATE = 2,
    RTC_CONFIG_ERROR_INVALID_ALARM_TIME = 3,
    RTC_CONFIG_ERROR_INVALID_ALARM_DATE = 4
};
```

```
void RTC_Handler(void)
{
    uint32_t ul_status = rtc_get_status(RTC);

    /* Time or date alarm */
    if ((ul_status & RTC_SR_ALARM) == RTC_SR_ALARM) {
        rtc_clear_status(RTC, RTC_SCCR_ALRCLR);
    }
}
```

```
static void configure_rtc_interrupt(void)
{

    NVIC_DisableIRQ(RTC_IRQn);
    NVIC_ClearPendingIRQ(RTC_IRQn);
    NVIC_SetPriority(RTC_IRQn, 0);
    NVIC_EnableIRQ(RTC_IRQn);
    rtc_enable_interrupt(RTC,RTC_IER_ALREN);
}
```

```
static enum rtc_config_error configure_time(const struct rtc_config const *conf)
{

    /* Validate parameter(s). */
    Assert(conf);

    /* Default RTC configuration, 24-hour mode. */
    rtc_set_hour_mode(RTC, 0);

    /* Set the RTC's date. */
    if (rtc_set_date(RTC, conf->date.ul_year,
            conf->date.ul_month,
            conf->date.ul_day,
            conf->date.ul_week)) {
        return(RTC_CONFIG_ERROR_INVALID_DATE);
    }

    /* Set the RTC's time. */
    if (rtc_set_time(RTC, conf->time.ul_hour,
            conf->time.ul_minute,
            conf->time.ul_second)) {
        return(RTC_CONFIG_ERROR_INVALID_TIME);
    }

    return(RTC_CONFIG_ERROR_NONE);
}
```

```
static enum rtc_config_error configure_alarm(const struct rtc_config const *conf)
{
```

```
    /* Validate parameter(s). */
    Assert(conf);

    /* Clear the RTC Alarm. */
    rtc_clear_date_alarm(RTC);
    rtc_clear_time_alarm(RTC);
    rtc_clear_status(RTC, RTC_SCCR_ALRCLR);

    /* Set the RTC Alarm date. */
    if (rtc_set_date_alarm(RTC, true, conf->date.ul_month,
            true, conf->date.ul_day)) {
        return(RTC_CONFIG_ERROR_INVALID_ALARM_DATE);
    }

    /* Set the RTC Alarm time. */
    if (rtc_set_time_alarm(RTC, true, conf->time.ul_hour,
            true, conf->time.ul_minute,
            true, conf->time.ul_second)) {
        return(RTC_CONFIG_ERROR_INVALID_ALARM_TIME);
    }

    return(RTC_CONFIG_ERROR_NONE);
}
```

```
static void error_handler(enum rtc_config_error error)
{

    switch (error) {
    case RTC_CONFIG_ERROR_NONE:
        puts("\r\n No error!\n");
        break;

    case RTC_CONFIG_ERROR_INVALID_TIME:
        puts("\r\n Invalid RTC Time setting!\n");
        break;

    case RTC_CONFIG_ERROR_INVALID_DATE:
        puts("\r\n Invalid RTC Date setting!\n");
        break;

    case RTC_CONFIG_ERROR_INVALID_ALARM_TIME:
        puts("\r\n Invalid RTC Alarm Time setting!\n");
        break;

    case RTC_CONFIG_ERROR_INVALID_ALARM_DATE:
        puts("\r\n Invalid RTC Alarm Date setting!\n");
    break;

    default:
        puts("\r\n Unknown error!");
    }
}
```

8.1.1.3 Workflow

Create an RTC time/date configuration struct with initial time/date values and program the RTC Time accordingly:

```
static const struct rtc_config homeTime = { {
    TIME_INIT_YEAR,
    TIME_INIT_MONTH,
```

```
    TIME_INIT_DAY,
    TIME_INIT_WEEK
    }, {
    TIME_INIT_HOUR,
    TIME_INIT_MINUTE,
    TIME_INIT_SECOND
    }
};
if ((error = configure_time(&homeTime)) != RTC_CONFIG_ERROR_NONE) {
    /* Handle any errors. */
    error_handler(error);
}
```

Create an RTC Alarm time/date configuration struct with alarm time/date values and program the RTC Alarm accordingly:

```
static const struct rtc_config alarmTime = { {
    0,      /* Year is not used. */
    ALARM_AT_MONTH,
    ALARM_AT_DAY,
    0       /* Week is not used. */
    }, {
    ALARM_AT_HOUR,
    ALARM_AT_MINUTE,
    ALARM_AT_SECOND
    }
};
if ((error = configure_alarm(&alarmTime)) != RTC_CONFIG_ERROR_NONE) {
    /* Handle any errors. */
    error_handler(error);
}
```

Configure the RTC Alarm interrupt:

```
configure_rtc_interrupt();
```

Enable the RTC Alarm as a wake-up source:

```
supc_set_wakeup_mode(SUPC, SUPC_WUMR_RTCEN);
```

Enter Backup Low Power Mode:

```
pmc_enable_backupmode();
#if (!(SAM4S) && !(SAM4E) && !(SAM4N) && !(SAM4C))
        supc_enable_backup_mode(SUPC);
#endif /* (!(SAM4S) && !(SAM4E) && !(SAM4N) && !(SAM4C)) */
```

After approximately 10 seconds (using the default values above) the RTC Alarm will trigger and the device will wake-up.

Note Without being located inside a loop this code sequence will execute once only.

After the initial setup and configuration the minimum amount of code to wake-up from an RTC Alarm is:

```
if ((error = configure_alarm(&alarmTime)) != RTC_CONFIG_ERROR_NONE) {
```

```
    /* Handle any errors. */
    error_handler(error);
}
```

```
pmc_enable_backupmode();
#if (!(SAM4S) && !(SAM4E) && !(SAM4N) && !(SAM4C))
        supc_enable_backup_mode(SUPC);
#endif /* (!(SAM4S) && !(SAM4E) && !(SAM4N) && !(SAM4C)) */
```

Index

E

F

Document Revision History

Doc. Rev.	Date	Comments
42319A	05/2014	Initial document release