SAM4 Real-Time Clock (RTC)

This driver for SAM4C/SAM4E/SAMG/SAM4N/SAM4S devices provides an interface for the configuration and management of the device's Real-Time Clock functionality and for the configuration and retrieval of the current time and date as maintained by the RTC module.

The RTC provides a full Binary-Coded Decimal (BCD) clock that includes century (19/20), year (with leap years), month, day, hour, minute, and second for Gregorian or Persian calendars. The time format can be in 24-hour or 12-hour mode with an AM/PM indicator.

The following peripherals are used by this module:

- RTC (Real-Time Clock)

The outline of this documentation is as follows:

- Prerequisites
- Module Overview
- Special Considerations
- Extra Information
- Examples
- API Overview
Table of Contents

SAM4 Real-Time Clock (RTC) ........................................................................ 1
Software License ......................................................................................... 4

1. Prerequisites .......................................................................................... 5

2. Module Overview .................................................................................. 6
   2.1. Time/Date Field Validation ............................................................. 6
   2.2. Timing .......................................................................................... 6
   2.3. Alarm .......................................................................................... 6
   2.4. Reference Clock .......................................................................... 6
   2.5. Accurate Clock Calibration ........................................................... 6

3. Special Considerations ......................................................................... 7
   3.1. Crystal Selection ........................................................................ 7
   3.2. Waveform Generation .................................................................. 7
   3.3. Year Limit .................................................................................. 7
   3.4. The Year 1900 ........................................................................... 7

4. Extra Information ................................................................................ 8

5. Examples ............................................................................................. 9

6. API Overview ........................................................................................ 10
   6.1. Function Definitions ................................................................ 10
       6.1.1. Function rtc_clear_date_alarm() ........................................ 10
       6.1.2. Function rtc_clear_status() .............................................. 10
       6.1.3. Function rtc_clear_time_alarm() ....................................... 10
       6.1.4. Function rtc_disable_interrupt() ....................................... 10
       6.1.5. Function rtc_enable_interrupt() ......................................... 11
       6.1.6. Function rtc_get_date() .................................................. 11
       6.1.7. Function rtc_get_hour_mode() .......................................... 11
       6.1.8. Function rtc_get_interrupt_mask() ................................... 12
       6.1.9. Function rtc_get_status() ................................................ 12
       6.1.10. Function rtc_get_tamper_date() ........................................ 12
       6.1.11. Function rtc_get_tamper_event_counter() ......................... 13
       6.1.12. Function rtc_get_tamper_source() .................................... 13
       6.1.13. Function rtc_get_tamper_time() ....................................... 14
       6.1.14. Function rtc_get_time() .................................................. 15
       6.1.15. Function rtc_is_tamper_occur_in_backup_mode() ............... 15
       6.1.16. Function rtc_set_calendar_mode() ................................... 16
       6.1.17. Function rtc_set_calibration() .......................................... 16
       6.1.18. Function rtc_set_date() ................................................. 16
       6.1.19. Function rtc_set_date_alarm() ......................................... 17
       6.1.20. Function rtc_set_hour_mode() ........................................ 17
       6.1.21. Function rtc_set_pulse_parameter() ................................. 18
       6.1.22. Function rtc_set_time() .................................................. 18
       6.1.23. Function rtc_set_time_alarm() ......................................... 19
       6.1.24. Function rtc_set_waveform() .......................................... 19
       6.1.25. Function rtc_set_writeprotect() ...................................... 20

7. Extra Information for Real-Time Clock Driver ..................................... 21
   7.1. Acronyms .................................................................................. 21
   7.2. Dependencies ............................................................................ 21
   7.3. Errata ....................................................................................... 21
   7.4. Module History ......................................................................... 21

8. Examples for Real-Time Clock Driver ................................................. 22
8.1. Quick Start guide for SAM RTC driver .................................................. 22
  8.1.1. Basic Use Case ........................................................................ 22
  8.1.2. Setup Steps ............................................................................ 22
  8.1.3. Usage Steps ........................................................................... 23

8.2. Real-Time Clock (RTC) Example ....................................................... 23
  8.2.1. Purpose ................................................................................ 23
  8.2.2. Requirements ....................................................................... 23
  8.2.3. Description .......................................................................... 23
  8.2.4. Main Files ........................................................................... 24
  8.2.5. Compilation Information ....................................................... 24
  8.2.6. Usage .................................................................................. 24

8.3. Real-Time Clock (RTC) Tamper Example .......................................... 25
  8.3.1. Purpose ................................................................................ 25
  8.3.2. Requirements ....................................................................... 25
  8.3.3. Description .......................................................................... 25
  8.3.4. Main Files ........................................................................... 25
  8.3.5. Compilation Information ....................................................... 25
  8.3.6. Usage .................................................................................. 25

Index .................................................................................................. 27

Document Revision History ................................................................. 28
Software License

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. The name of Atmel may not be used to endorse or promote products derived from this software without specific prior written permission.

4. This software may only be redistributed and used in connection with an Atmel microcontroller product.

THIS SOFTWARE IS PROVIDED BY ATMEL "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT ARE EXPRESSLY AND SPECIFICALLY DISCLAIMED. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
1. Prerequisites

There are no prerequisites for this module.
2. Module Overview

The RTC provides a full binary-coded decimal (BCD) clock that includes century (19/20), year (with leap years), month, day, hours, minutes, and seconds. The RTC can operate in 24-hour mode or in 12-hour mode with an AM/PM indicator. Automatic corrections for leap years are included (all years divisible by four being leap years).

2.1 Time/Date Field Validation

To avoid unwanted side effects, verification is performed on user application writes to the century, year, month, day, hours, minutes, seconds, and alarms. If any field fails its check the new data is not loaded into the destination register/counter and a flag is set in the validity register. When the user application programs acceptable time/date fields the validity flag is cleared.

Note
Refer to "Error Checking when Programming" located in the RTC section of the device's datasheet for more information on each field's validation criteria.

2.2 Timing

The RTC is updated in real time at one-second intervals in normal mode for the counters of seconds, at one-minute intervals for the counter of minutes and so on. Due to the asynchronous operation of the RTC with respect to the rest of the chip, to be certain that the value read in the RTC registers (century, year, month, date, day, hours, minutes, seconds) are valid and stable, it is necessary to read these registers twice and perform a comparison.

2.3 Alarm

The RTC Alarm has five programmable fields: month, day, hours, minutes, and seconds. Each of these fields can be enabled or disabled individually to match the alarm condition: Depending on the combination of fields enabled, a large number of possibilities are available to the user application ranging from minutes to 365/366 days.

2.4 Reference Clock

The reference clock is Slow Clock (SLCK). It can be driven internally or by an external 32.768kHz crystal and also runs during the low power modes of the processor.

2.5 Accurate Clock Calibration

The crystal oscillator that drives the RTC may not be as accurate as expected mainly due to temperature variation. The RTC module is equipped with circuitry able to correct slow clock crystal drift.
3. Special Considerations

3.1 Crystal Selection
The external crystal selection used by the RTC module in the final system design must take into account:

- Current consumption to achieve the best power savings in low power operating modes
- Frequency drift (due to temperature effects on the circuit) for the best time accuracy

3.2 Waveform Generation
Waveforms can be generated on the RTC module's RTCOUT0 and RTCOUT1 outputs. The low power operating modes of the device do not affect their operation which allows them to be used as periodic or alarm based stimulus to external devices, peripherals or subsystems.

3.3 Year Limit
The RTC module has a year range between 1900 and 2099 in Gregorian mode (1300 to 1499 in Persian mode). Dates outside the calendar mode's specified range will need software adjustment.

3.4 The Year 1900
If the user application attempts to configure the RTC to the 29th of February 1900 the date will fail the RTC module's field validation checks.
4. **Extra Information**

For extra information, see *Extra Information for Real-Time Clock Driver*. This includes:

- Acronyms
- Dependencies
- Errata
- Module History
5. Examples

For a list of examples related to this driver, see Examples for Real-Time Clock Driver.
6. API Overview

6.1 Function Definitions

6.1.1 Function rtc_clear_date_alarm()

Clear the RTC date alarm setting.

```c
void rtc_clear_date_alarm(
    Rtc * p_rtc)
```

Table 6-1. Parameters

<table>
<thead>
<tr>
<th>Data direction</th>
<th>Parameter name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[out]</td>
<td>p_rtc</td>
<td>Module hardware register base address pointer</td>
</tr>
</tbody>
</table>

6.1.2 Function rtc_clear_status()

Set the RTC SCCR to clear status bits.

```c
void rtc_clear_status(
    Rtc * p_rtc,
    uint32_t ul_clear)
```

Table 6-2. Parameters

<table>
<thead>
<tr>
<th>Data direction</th>
<th>Parameter name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[out]</td>
<td>p_rtc</td>
<td>Module hardware register base address pointer</td>
</tr>
<tr>
<td>[in]</td>
<td>ul_clear</td>
<td>Some flag bits which will be cleared</td>
</tr>
</tbody>
</table>

6.1.3 Function rtc_clear_time_alarm()

Clear the RTC time alarm setting.

```c
void rtc_clear_time_alarm(
    Rtc * p_rtc)
```

Table 6-3. Parameters

<table>
<thead>
<tr>
<th>Data direction</th>
<th>Parameter name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[out]</td>
<td>p_rtc</td>
<td>Module hardware register base address pointer</td>
</tr>
</tbody>
</table>

6.1.4 Function rtc_disable_interrupt()

Disable RTC interrupts.

```c
void rtc_disable_interrupt(
    Rtc * p_rtc,
    uint32_t ul_clear)
```
6.1.5 Function rtc_enable_interrupt()

Enable the RTC interrupts.

```c
void rtc_enable_interrupt(
    Rtc * p_rtc,
    uint32_t ul_sources)
```

6.1.6 Function rtc_get_date()

Get the RTC date value.

```c
void rtc_get_date(
    Rtc * p_rtc,
    uint32_t * pul_year,
    uint32_t * pul_month,
    uint32_t * pul_day,
    uint32_t * pul_week)
```

6.1.7 Function rtc_get_hour_mode()

Get the RTC hour mode.

```c
uint32_t rtc_get_hour_mode()
```
6.1.8 Function rtc_get_interrupt_mask()

_read RTC interrupt mask_.

```c
uint32_t rtc_get_interrupt_mask(
    Rtc * p_rtc)
```

<table>
<thead>
<tr>
<th>Return value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>RTC is in 24-hour mode</td>
</tr>
<tr>
<td>1</td>
<td>RTC is in 12-hour mode</td>
</tr>
</tbody>
</table>

6.1.9 Function rtc_get_status()

_get the RTC status_.

```c
uint32_t rtc_get_status(
    Rtc * p_rtc)
```

<table>
<thead>
<tr>
<th>Data direction</th>
<th>Parameter name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in]</td>
<td>p_rtc</td>
<td>Module hardware register base address pointer</td>
</tr>
</tbody>
</table>

| Returns | Status of the RTC. |
Get the RTC tamper date.

```c
void rtc_get_tamper_date(
    Rtc * p_rtc,
    uint32_t * pul_year,
    uint32_t * pul_month,
    uint32_t * pul_day,
    uint32_t * pul_week,
    uint8_t  reg_num)
```

**Note**
This function should be called before `rtc_get_tamper_source()` function call. Otherwise, the tamper date will be cleared.
This function is only available on SAM4C devices.

<table>
<thead>
<tr>
<th>Data direction</th>
<th>Parameter name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in]</td>
<td>p_rtc</td>
<td>Module hardware register base address pointer</td>
</tr>
<tr>
<td>[out]</td>
<td>pul_year</td>
<td>Current year</td>
</tr>
<tr>
<td>[out]</td>
<td>pul_month</td>
<td>Current month</td>
</tr>
<tr>
<td>[out]</td>
<td>pul_day</td>
<td>Current day</td>
</tr>
<tr>
<td>[out]</td>
<td>pul_week</td>
<td>Current day in current week</td>
</tr>
<tr>
<td>[in]</td>
<td>reg_num</td>
<td>Current tamper register set number</td>
</tr>
</tbody>
</table>

### 6.1.11 Function rtc_get_tamper_event_counter()

Get the RTC tamper event counter.

```c
uint32_t rtc_get_tamper_event_counter(
    Rtc * p_rtc)
```

**Note**
This function should be called before `rtc_get_tamper_source()` function call. Otherwise, the tamper event counter will be cleared.
This function is only available on SAM4C devices.

<table>
<thead>
<tr>
<th>Data direction</th>
<th>Parameter name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in]</td>
<td>p_rtc</td>
<td>Module hardware register base address pointer</td>
</tr>
</tbody>
</table>

**Returns**
The RTC Tamper event counter.

### 6.1.12 Function rtc_get_tamper_source()
Get the RTC tamper source.

```c
uint32_t rtc_get_tamper_source(
    Rtc * p_rtc,
    uint8_t reg_num)
```

**Note**
This function is only available on SAM4C devices.

**Table 6-13. Parameters**

<table>
<thead>
<tr>
<th>Data direction</th>
<th>Parameter name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in]</td>
<td>p_rtc</td>
<td>Module hardware register base address pointer</td>
</tr>
<tr>
<td>[in]</td>
<td>reg_num</td>
<td>Current tamper register set number</td>
</tr>
</tbody>
</table>

**Returns**
The RTC Tamper source.

**Table 6-14. Return Values**

<table>
<thead>
<tr>
<th>Return value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Tamper occurred on tamper pin TMP0</td>
</tr>
<tr>
<td>1</td>
<td>Tamper occurred on tamper pin TMP1</td>
</tr>
<tr>
<td>2</td>
<td>Tamper occurred on tamper pin TMP2</td>
</tr>
<tr>
<td>3</td>
<td>Tamper occurred on tamper pin TMP3</td>
</tr>
</tbody>
</table>

### 6.1.13 Function rtc_get_tamper_time()

Get the RTC tamper time value.

```c
void rtc_get_tamper_time(
    Rtc * p_rtc,
    uint32_t * pul_hour,
    uint32_t * pul_minute,
    uint32_t * pul_second,
    uint8_t reg_num)
```

**Note**
This function should be called before `rtc_get_tamper_source()` function call. Otherwise, the tamper time will be cleared.
This function is only available on SAM4C devices.

**Table 6-15. Parameters**

<table>
<thead>
<tr>
<th>Data direction</th>
<th>Parameter name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in]</td>
<td>p_rtc</td>
<td>Module hardware register base address pointer</td>
</tr>
<tr>
<td>[out]</td>
<td>pul_hour</td>
<td>Current hour, 24-hour mode</td>
</tr>
<tr>
<td>[out]</td>
<td>pul_minute</td>
<td>Current minute</td>
</tr>
</tbody>
</table>
### 6.1.14 Function rtc_get_time()

*Get the RTC time value.*

```c
void rtc_get_time(
    Rtc * p_rtc,
    uint32_t * pul_hour,
    uint32_t * pul_minute,
    uint32_t * pul_second)
```

#### Table 6-16. Parameters

<table>
<thead>
<tr>
<th>Data direction</th>
<th>Parameter name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in, out]</td>
<td>p_rtc</td>
<td>Module hardware register base address pointer</td>
</tr>
<tr>
<td>[out]</td>
<td>pul_hour</td>
<td>Current hour, 24-hour mode</td>
</tr>
<tr>
<td>[out]</td>
<td>pul_minute</td>
<td>Current minute</td>
</tr>
<tr>
<td>[out]</td>
<td>pul_second</td>
<td>Current second</td>
</tr>
</tbody>
</table>

### 6.1.15 Function rtc_is_tamper_occur_in_backup_mode()

*Check the system is in backup mode when RTC tamper event happen.*

```c
bool rtc_is_tamper_occur_in_backup_mode(
    Rtc * p_rtc,
    uint8_t  reg_num)
```

#### Note

This function should be called before the function `rtc_get_tamper_source()` is called. Otherwise, the flag that indicates a tamper occurred in backup mode will be cleared.

This function is only available on SAM4C devices.

#### Table 6-17. Parameters

<table>
<thead>
<tr>
<th>Data direction</th>
<th>Parameter name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in]</td>
<td>p_rtc</td>
<td>Module hardware register base address pointer</td>
</tr>
<tr>
<td>[in]</td>
<td>reg_num</td>
<td>Current tamper register set number</td>
</tr>
</tbody>
</table>

#### Returns

RTC Tamper occurred in backup mode.

#### Table 6-18. Return Values

<table>
<thead>
<tr>
<th>Return value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>The system is different from backup mode</td>
</tr>
</tbody>
</table>
6.1.16 Function rtc_set_calendar_mode()

Set the RTC calendar mode.

```c
void rtc_set_calendar_mode(
    Rtc * p_rtc,
    uint32_t ul_mode)
```

<table>
<thead>
<tr>
<th>Data direction</th>
<th>Parameter name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in, out]</td>
<td>p_rtc</td>
<td>Module hardware register base address pointer</td>
</tr>
<tr>
<td>[in]</td>
<td>ul_mode</td>
<td>1 for Persian mode, 0 for Gregorian mode</td>
</tr>
</tbody>
</table>

6.1.17 Function rtc_set_calibration()

Set the RTC calibration.

```c
void rtc_set_calibration(
    Rtc * p_rtc,
    uint32_t ul_direction_ppm,
    uint32_t ul_correction,
    uint32_t ul_range_ppm)
```

<table>
<thead>
<tr>
<th>Data direction</th>
<th>Parameter name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in, out]</td>
<td>p_rtc</td>
<td>Module hardware register base address pointer</td>
</tr>
<tr>
<td>[in]</td>
<td>ul_direction_ppm</td>
<td>Positive/negative correction</td>
</tr>
<tr>
<td>[in]</td>
<td>ul_correction</td>
<td>Correction value</td>
</tr>
<tr>
<td>[in]</td>
<td>ul_range_ppm</td>
<td>Low/high range correction</td>
</tr>
</tbody>
</table>

6.1.18 Function rtc_set_date()

Set the RTC date.

```c
uint32_t rtc_set_date(
    Rtc * p_rtc,
    uint32_t ul_year,
    uint32_t ul_month,
    uint32_t ul_day,
    uint32_t ul_week)
```
Table 6-21. Parameters

<table>
<thead>
<tr>
<th>Data direction</th>
<th>Parameter name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in, out]</td>
<td>p_rtc</td>
<td>Module hardware register base address pointer</td>
</tr>
<tr>
<td>[in]</td>
<td>ul_year</td>
<td>Current year</td>
</tr>
<tr>
<td>[in]</td>
<td>ul_month</td>
<td>Current month</td>
</tr>
<tr>
<td>[in]</td>
<td>ul_day</td>
<td>Current day</td>
</tr>
<tr>
<td>[in]</td>
<td>ul_week</td>
<td>Current day in current week</td>
</tr>
</tbody>
</table>

**Returns**

RTC date field validation result.

Table 6-22. Return Values

<table>
<thead>
<tr>
<th>Return value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>RTC date configured correctly</td>
</tr>
<tr>
<td>RTC_VER_NVCAL</td>
<td>An RTC date field was not configured correctly</td>
</tr>
</tbody>
</table>

6.1.19 **Function rtc_set_date_alarm()**

*Set the RTC alarm date value.*

```c
uint32_t rtc_set_date_alarm(
    Rtc * p_rtc,
    uint32_t ul_month_flag,
    uint32_t ul_month,
    uint32_t ul_day_flag,
    uint32_t ul_day)
```

Table 6-23. Parameters

<table>
<thead>
<tr>
<th>Data direction</th>
<th>Parameter name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in, out]</td>
<td>p_rtc</td>
<td>Module hardware register base address pointer</td>
</tr>
<tr>
<td>[in]</td>
<td>ul_month_flag</td>
<td>1 for setting, 0 for not setting</td>
</tr>
<tr>
<td>[in]</td>
<td>ul_month</td>
<td>Alarm month value</td>
</tr>
<tr>
<td>[in]</td>
<td>ul_day_flag</td>
<td>1 for setting, 0 for not setting</td>
</tr>
<tr>
<td>[in]</td>
<td>ul_day</td>
<td>Alarm day value</td>
</tr>
</tbody>
</table>

**Returns**

Alarm field validation result.

Table 6-24. Return Values

<table>
<thead>
<tr>
<th>Return value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Alarm date configured correctly</td>
</tr>
<tr>
<td>RTC_VER_NVCALALR</td>
<td>An alarm date field was not configured correctly</td>
</tr>
</tbody>
</table>

6.1.20 **Function rtc_set_hour_mode()**
Set the RTC hour mode.

```c
void rtc_set_hour_mode(
Rtc * p_rtc,
uint32_t ul_mode)
```

### Table 6-25. Parameters

<table>
<thead>
<tr>
<th>Data direction</th>
<th>Parameter name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in, out]</td>
<td>p_rtc</td>
<td>Module hardware register base address pointer</td>
</tr>
<tr>
<td>[in]</td>
<td>ul_mode</td>
<td>1 for 12-hour mode, 0 for 24-hour mode</td>
</tr>
</tbody>
</table>

#### 6.1.21 Function rtc_set_pulse_parameter()

Set the pulse output waveform parameters.

```c
void rtc_set_pulse_parameter(
Rtc * p_rtc,
uint32_t ul_time_high,
uint32_t ul_period)
```

**Note**

This function is only available on SAM3S8, SAM3SD8, SAM4S, and SAM4C devices.

### Table 6-26. Parameters

<table>
<thead>
<tr>
<th>Data direction</th>
<th>Parameter name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in, out]</td>
<td>p_rtc</td>
<td>Module hardware register base address pointer</td>
</tr>
<tr>
<td>[in]</td>
<td>ul_time_high</td>
<td>High duration of the output pulse</td>
</tr>
<tr>
<td>[in]</td>
<td>ul_period</td>
<td>Period of the output pulse</td>
</tr>
</tbody>
</table>

#### 6.1.22 Function rtc_set_time()

Set the RTC time value.

```c
uint32_t rtc_set_time(
Rtc * p_rtc,
uint32_t ul_hour,
uint32_t ul_minute,
uint32_t ul_second)
```

### Table 6-27. Parameters

<table>
<thead>
<tr>
<th>Data direction</th>
<th>Parameter name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in, out]</td>
<td>p_rtc</td>
<td>Module hardware register base address pointer</td>
</tr>
<tr>
<td>[in]</td>
<td>ul_hour</td>
<td>Current hour, 24-hour mode</td>
</tr>
<tr>
<td>[in]</td>
<td>ul_minute</td>
<td>Current minute</td>
</tr>
</tbody>
</table>
### Data direction | Parameter name | Description
--- | --- | ---
[in] | ul_second | Current second

**Returns**

RTC time field validation result.

### Table 6-28. Return Values

<table>
<thead>
<tr>
<th>Return value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>RTC time configured correctly</td>
</tr>
<tr>
<td>RTC_VER_NVTIM</td>
<td>An Alarm time field was not configured correctly</td>
</tr>
</tbody>
</table>

**6.1.23 Function rtc_set_time_alarm()**

*Set the RTC alarm time value.*

```c
uint32_t rtc_set_time_alarm(
    Rtc * p_rtc,
    uint32_t ul_hour_flag,
    uint32_t ul_hour,
    uint32_t ul_minute_flag,
    uint32_t ul_minute,
    uint32_t ul_second_flag,
    uint32_t ul_second)
```

**Table 6-29. Parameters**

<table>
<thead>
<tr>
<th>Data direction</th>
<th>Parameter name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in, out]</td>
<td>p_rtc</td>
<td>Module hardware register base address pointer</td>
</tr>
<tr>
<td>[in]</td>
<td>ul_hour_flag</td>
<td>1 for hour field update, 0 otherwise</td>
</tr>
<tr>
<td>[in]</td>
<td>ul_hour</td>
<td>Alarm hour value, 24-hour mode</td>
</tr>
<tr>
<td>[in]</td>
<td>ul_minute_flag</td>
<td>1 for minute field update, 0 otherwise</td>
</tr>
<tr>
<td>[in]</td>
<td>ul_minute</td>
<td>Alarm minute value</td>
</tr>
<tr>
<td>[in]</td>
<td>ul_second_flag</td>
<td>1 for second field update, 0 otherwise</td>
</tr>
<tr>
<td>[in]</td>
<td>ul_second</td>
<td>Alarm second value</td>
</tr>
</tbody>
</table>

**Returns**

RTC Alarm field validation result.

### Table 6-30. Return Values

<table>
<thead>
<tr>
<th>Return value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Alarm time configured correctly</td>
</tr>
<tr>
<td>RTC_VER_NVIM</td>
<td>An Alarm time field was not configured correctly</td>
</tr>
</tbody>
</table>

**6.1.24 Function rtc_set_waveform()**
Set the RTC output waveform.

```c
void rtc_set_waveform(
    Rtc * p_rtc,
    uint32_t ul_channel,
    uint32_t ul_value)
```

**Note** This function is only available on SAM3S8, SAM3SD8, SAM4S, SAM4C, and SAMG devices.

<table>
<thead>
<tr>
<th>Data direction</th>
<th>Parameter name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in, out]</td>
<td>p_rtc</td>
<td>Module hardware register base address pointer</td>
</tr>
<tr>
<td>[in]</td>
<td>ul_channel</td>
<td>Output channel selection</td>
</tr>
<tr>
<td>[in]</td>
<td>ul_value</td>
<td>Output source selection value</td>
</tr>
</tbody>
</table>

6.1.25 Function rtc_set_writeprotect()

Enable or disable write protection of RTC registers.

```c
void rtc_set_writeprotect(
    Rtc * p_rtc,
    uint32_t ul_enable)
```

**Note** This function is only available on SAM3N, SAM3U, and SAM3XA devices.

<table>
<thead>
<tr>
<th>Data direction</th>
<th>Parameter name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[out]</td>
<td>p_rtc</td>
<td>Module hardware register base address pointer</td>
</tr>
<tr>
<td>[in]</td>
<td>ul_enable</td>
<td>1 to enable, 0 to disable</td>
</tr>
</tbody>
</table>
7. Extra Information for Real-Time Clock Driver

7.1 Acronyms

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

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCD</td>
<td>Binary-Coded Decimal</td>
</tr>
<tr>
<td>QSG</td>
<td>Quick Start Guide</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Changelog</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial document release</td>
</tr>
</tbody>
</table>
8. Examples for Real-Time Clock Driver

This is a list of the available Quick Start guides (QSGs) and example applications for SAM4 Real-Time Clock (RTC). 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 SAM RTC driver
- Real-Time Clock (RTC) Example
- Real-Time Clock (RTC) Tamper Example

8.1 Quick Start guide for SAM RTC driver

This is the quick start guide for the SAM Real-Time Clock (RTC) driver, with step-by-step instructions on how to configure and use the driver in a selection of use cases.

The use cases contain several code fragments. The code fragments in the steps for setup can be copied into a custom initialization function, while the steps for usage can be copied into, e.g., the main application function.

8.1.1 Basic Use Case

In this basic use case, the RTC module is using 32kHz external crystal and configured for 24-hour mode. It will read the current date and time.

8.1.1.1 Prerequisites

- System Clock Management (Sysclock)

8.1.2 Setup Steps

8.1.2.1 Example Code

Add this to the application C-file:

```c
void rtc_setup(void)
{
    pmc_switch_sclk_to_32kxtal(PMC_OSC_XTAL);

    while (!pmc_osc_is_ready_32kxtal());

    rtc_set_hour_mode(RTC, 0);
}
```

8.1.2.2 Workflow

**Note**

Ensure that the external 32kHz crystal is available.

1. Enable the External 32kHz crystal:

   ```c
   pmc_switch_sclk_to_32kxtal(PMC_OSC_XTAL);
   ```

2. Wait for the 32kHz crystal to be ready:

   ```c
   while (!pmc_osc_is_ready_32kxtal());
   ```

3. Set default RTC configuration, 24-hour mode:
8.1.3 Usage Steps

8.1.3.1 Example Code
Add to, e.g., main loop in application C-file:

```c
    uint32_t hour, minute, second;
    uint32_t year, month, day, week;
    
    rtc_get_time(RTC, &hour, &minute, &second);
    rtc_get_date(RTC, &year, &month, &day, &week);
```

8.1.3.2 Workflow
1. Define the variables for the date and time:
   ```c
   uint32_t hour, minute, second;
   uint32_t year, month, day, week;
   ```

2. Read current time:
   ```c
   rtc_get_time(RTC, &hour, &minute, &second);
   ```

3. Read current date:
   ```c
   rtc_get_date(RTC, &year, &month, &day, &week);
   ```

8.2 Real-Time Clock (RTC) Example

8.2.1 Purpose
This basic example shows how to use the Real-Time Clock (RTC) peripheral available on SAM devices. The RTC enables easy time and date management and allows the user to monitor events like configurable alarm, second change, calendar change, and so on.

8.2.2 Requirements
This example can be used with SAM4 evaluation kits such as the SAM4S Xplained, the SAM4N Xplained Pro, and other evaluation kits. Refer to the list of kits available for the actual device on http://www.atmel.com.

8.2.3 Description
Upon startup, the program displays the currently set time and date and a menu to perform the following:

```
* Menu:
*   t - Set time
*   d - Set date
*   i - Set time alarm
*   m - Set date alarm
*   c - Clear the alarm notification (only if it has been triggered)
*   w - Generate Waveform
```
"w" is an additional option for SAM3S8, SAM3SD8, SAM4S, and SAM4C devices. An RTC output can be programmed to generate several waveforms, including a prescaled clock derived from the Slow Clock (SCLK). Setting the time, date, and time alarm is done by using Menu option, and the display is updated accordingly. The time alarm is triggered only when the second, minute, and hour match the preset values; the date alarm is triggered only when the month and date match the preset values. Generating waveform is done by using Menu option "w" and a menu to perform the following:

```
* Menu:
* 0 - No Waveform
* 1 - 1 Hz square wave
* 2 - 32 Hz square wave
* 3 - 64 Hz square wave
* 4 - 512 Hz square wave
* 5 - Toggles when alarm flag rise
* 6 - Copy of the alarm flag
* 7 - Duty cycle programmable pulse
* 8 - Quit
```

Note
The example is using RC oscillator by default. This would generate an accuracy problem for the RTC if not calibrated. It is recommended to use an external 32kHz crystal to get high accuracy. How to initialize RTC with external 32kHz crystal can be found at Quick Start guide for SAM RTC driver.

8.2.4 Main Files
- rtc.c: Real-Time Clock driver
- rtc.h: Real-Time Clock driver header file
- rtc_example.c: Real-Time Clock example application

8.2.5 Compilation Information
This software is written for GNU GCC and IAR Embedded Workbench® for Atmel®. Other compilers may or may not work.

8.2.6 Usage
1. Build the program and download it into the evaluation board.
2. On the computer, open, and configure a terminal application (e.g., HyperTerminal on Microsoft® Windows®) with these settings:
   - 115200 baud
   - 8 bits of data
   - No parity
   - 1 stop bit
   - No flow control
3. Start the application.
4. In the terminal window, the following text should appear:

```
* -- RTC Example xxx --
* -- xxxxxx-xx
* -- Compiled: xxx xx xxxx xx:xx:xx --
*
* Menu:
* t - Set time
* d - Set date
* i - Set time alarm
* m - Set date alarm
```

5. Press one of the keys listed in the menu to perform the corresponding action.

8.3 Real-Time Clock (RTC) Tamper Example

8.3.1 Purpose
This basic example shows how to use the Real-Time Clock (RTC) tamper function in SAM.

8.3.2 Requirements
This example can be used with SAM4C evaluation kits.

8.3.3 Description
Upon startup, the system sets the tamper pin TMP0 and TMP2 as wake-up sources. After any key press the system will enter Backup Low Power Mode. By pressing the BP3(TMP0) or BP6(TMP2) button, the system will wake-up and clear the GPBR0~7 register automatically and display the tamper event occurrence time/date.

**Note**
This example uses the RTCOUT pin which is a common I/O with TDO on the device’s JTAG interface. To debug the application normally select the SWD interface.

8.3.4 Main Files
- rtc.c: Real-Time Clock driver
- rtc.h: Real-Time Clock driver header file
- rtc_tamper_example.c: Real-Time Clock tamper example application

8.3.5 Compilation Information
This software is written for GNU GCC and IAR Embedded Workbench for Atmel. Other compilers may or may not work.

8.3.6 Usage
1. Build the program and download it into the evaluation board.
2. On the computer, open, and configure a terminal application (e.g., HyperTerminal on Microsoft Windows) with these settings:
   - 115200 baud
   - 8 bits of data
   - No parity
   - 1 stop bit
- No flow control

3. Start the application.

4. In the terminal window, the following text should appear:

```
*   -- RTC Tamper Example xxx --
*   -- xxxxxxx-xx
*   -- Compiled: xxx xx xxxx xx:xx:xx --
```
Index

F
Function Definitions
  rtc_clear_date_alarm, 10
  rtc_clear_status, 10
  rtc_clear_time_alarm, 10
  rtc_disable_interrupt, 10
  rtc_enable_interrupt, 11
  rtc_get_date, 11
  rtc_get_hour_mode, 11
  rtc_get_interrupt_mask, 12
  rtc_get_status, 12
  rtc_get_tamper_date, 12
  rtc_get_tamper_event_counter, 13
  rtc_get_tamper_source, 13
  rtc_get_tamper_time, 14
  rtc_get_time, 15
  rtc_is_tamper_occur_in_backup_mode, 15
  rtc_set_calendar_mode, 16
  rtc_set_calibration, 16
  rtc_set_date, 16
  rtc_set_date_alarm, 17
  rtc_set_hour_mode, 17
  rtc_set_pulse_parameter, 18
  rtc_set_time, 18
  rtc_set_time_alarm, 19
  rtc_set_waveform, 19
  rtc_set_writeprotect, 20
## Document Revision History

<table>
<thead>
<tr>
<th>Doc. Rev.</th>
<th>Date</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>42284A</td>
<td>05/2014</td>
<td>Initial document release</td>
</tr>
</tbody>
</table>