

# RTC

Home

## Apps

Here is a list of all modules:

- [License Terms and Copyright Information](#)
- [Abbreviations and Definitions](#)
- [Overview](#)
- [Architecture Description](#)
- [APP Configuration Parameters](#)
- [Enumerations](#)
- [Data structures](#)
- [Methods](#)
- [Usage](#)
- [Release History](#)



## License Terms and Copyright Information

### License Terms and Copyright Information

Copyright (c) 2015, Infineon Technologies AG All rights reserved.

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

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. 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. Neither the name of the copyright holders nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS 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.

To improve the quality of the software, users are encouraged to share modifications, enhancements or bug fixes with Infineon Technologies AG ([dave@infineon.com](mailto:dave@infineon.com)).

---

# RTC

Home

## Abbreviations and Definitions

### Abbreviations and Definitions

| <b>Abbreviations:</b> |                                      |
|-----------------------|--------------------------------------|
| DAVE™                 | Digital Application Virtual Engineer |
| APP                   | DAVE Application                     |
| API                   | Application Programming Interface    |
| GUI                   | Graphical User Interface             |
| MCU                   | Microcontroller Unit                 |
| SW                    | Software                             |
| HW                    | Hardware                             |
| LLD                   | Low Level Driver                     |
| SCU                   | System Control Unit                  |
| IO                    | Input Output                         |
| RTC                   | Real Time Clock                      |
| NMI                   | Non Maskable Interrupt               |

| <b>Definitions:</b>      |   |
|--------------------------|---|
| Singleton                | Only single instance of the APP is permitted                  |
| Sharable                 | Resource sharing with other APPs is permitted                 |
| initProvider             | Provides the initialization routine                           |
| Physical connectivity    | Hardware inter/intra peripheral (constant) signal connection  |
| Conditional connectivity | Constrained hardware inter/intra peripheral signal connection |
| Aggregation              | Indicates consumption of low level (dependent)                |

APPS

---

---

# RTC

Home

## Overview

### Overview

RTC module tracks time with separate registers for hours, minutes and seconds. The calendar registers track date, time, month and year with automatic leap year correction.

RTC is capable of running from an alternate source of power, so it can continue to keep time while the primary source of power is off or unavailable.

The timer remains operational when core domain is in power down mode.

The kernel part of the RTC keeps running as long as the hibernate domain is powered with an alternate supply source.

The APP allows the user to configure the current date and time, as well as alarm settings, through GUI and at runtime using APIs.

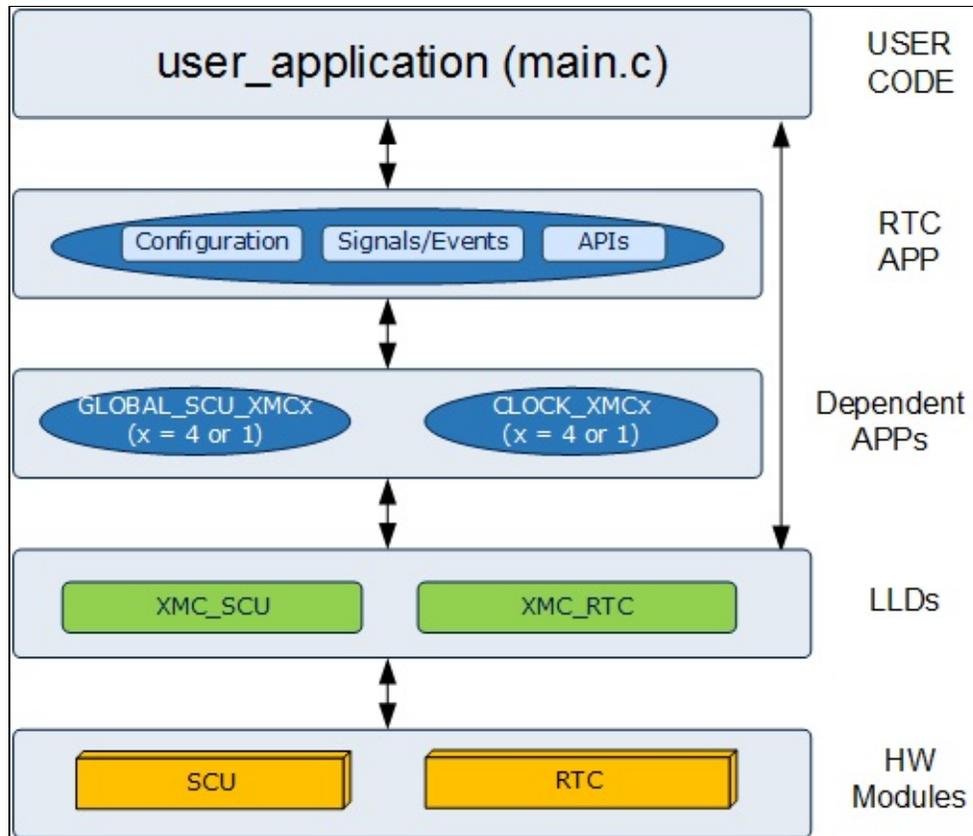
On XMC4000, the timer event and alarm event can be used to trigger the SCU interrupt or NMI(Non Maskable Interrupt).

The features of Real Time Clock module are:

Precise real time keeping with

- 32.768 kHz external crystal clock
- 32.768 kHz high precision internal crystal clock
- Periodic time base interrupt
- Programmable alarm interrupt on time match

Note: In XMC1000, there is only a single power domain (no hibernate domain is present).



**Figure 1 : HW and SW connectivity view of RTC APP**

### Limitations

Wake-up mechanism from hibernate state is not supported in this APP version.

### Supported Devices

1. XMC4800/XMC4700 Series
2. XMC4500 Series
3. XMC4400 Series
4. XMC4300 Series
5. XMC4200 / XMC4100 Series
6. XMC1400 Series
7. XMC1300 Series
8. XMC1200 Series
9. XMC1100 Series

### References

1. XMC4800/XMC4700 Reference Manual
2. XMC4500 Reference Manual
3. XMC4400 Reference Manual
4. XMC4300 Reference Manual
5. XMC4200 / XMC4100 Reference Manual
6. XMC1400 Reference Manual
7. XMC1300 Reference Manual
8. XMC1200 Reference Manual
9. XMC1100 Reference Manual

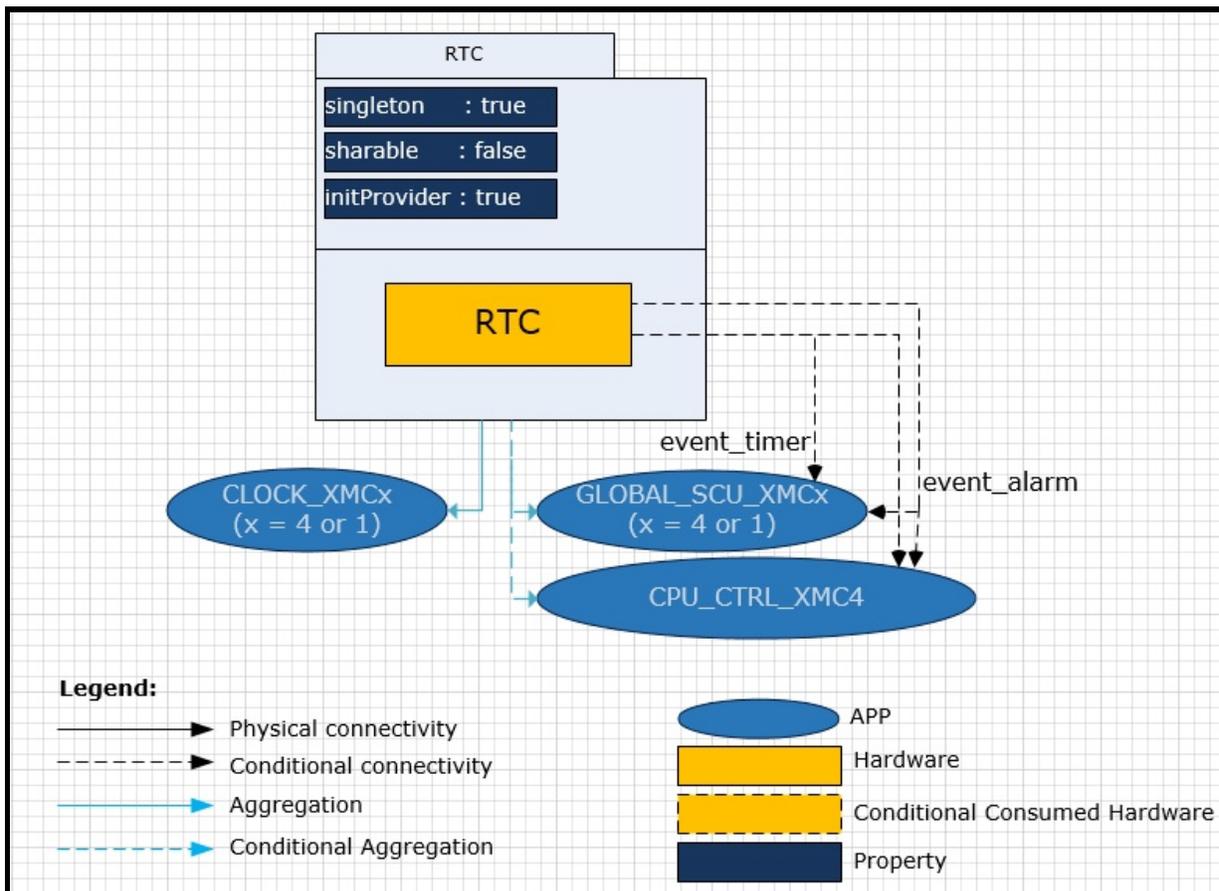


# RTC

Home

## Architecture Description

### Architecture Description



**Figure 1 : Architecture of RTC APP**

This is the software architecture representation of the RTC APP. The GLOBAL\_SCU\_XMC, CPU\_CTRL\_XMC4 and CLOCK\_XMC APPs are aggregated. GLOBAL\_SCU APP will get consumed whenever event trigger is SCU Interrupt and any periodic event / alarm interrupt is enabled in RTC APP. CPU\_CTRL\_XMC4 APP will get consumed whenever event trigger is NMI Interrupt and any periodic event / alarm

interrupt is enabled in RTC APP. A RTC APP instance exists in a DAVE™ project with fixed attributes and provides output signals for inter-peripheral connections. An instantiated APP (after code generation) generates a specific data structure with the GUI configuration. The name of this data structure can be modified by changing the APP instance label (e.g. changing (default label) RTC\_0 to (user defined label) MY\_RTC).

### Signals:

The following table describes the list of IO signals for RTC APP.

**Table 1:** APP Input Output signals

| Signal Name | Input/Output | Availability | Description   |
|-------------|--------------|--------------|---|
| event_timer | Output       | Conditional  | RTC timer interrupt signal. This signal gets connected to an NVIC node if any of the periodic events is selected in UI (e.g. Days, Months, Years, Hours, Minutes, Seconds). |
| event_alarm | Output       | Conditional  | RTC alarm interrupt signal. This signal gets connected to an NVIC   |

node if alarm  
interrupt is  
enabled in UI

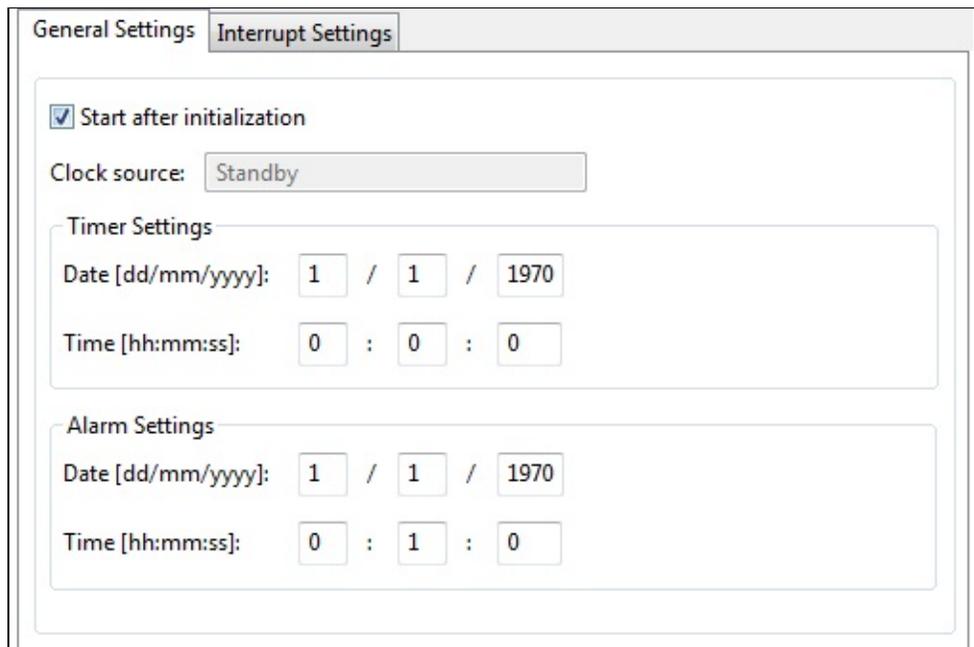


# RTC

Home

## APP Configuration Parameters

### App Configuration Parameters



The screenshot shows a configuration window for the RTC. It has two tabs: 'General Settings' (selected) and 'Interrupt Settings'. Under 'General Settings', there is a checked checkbox for 'Start after initialization'. Below that is a 'Clock source' dropdown menu set to 'Standby'. There are two sections: 'Timer Settings' and 'Alarm Settings'. Each section has a 'Date [dd/mm/yyyy]' field with values '1 / 1 / 1970' and a 'Time [hh:mm:ss]' field with values '0 : 0 : 0' for the timer and '0 : 1 : 0' for the alarm.

| Section          | Field                      | Value                               |
|------------------|----------------------------|-------------------------------------|
| General Settings | Start after initialization | <input checked="" type="checkbox"/> |
|                  | Clock source               | Standby                             |
| Timer Settings   | Date [dd/mm/yyyy]          | 1 / 1 / 1970                        |
|                  | Time [hh:mm:ss]            | 0 : 0 : 0                           |
| Alarm Settings   | Date [dd/mm/yyyy]          | 1 / 1 / 1970                        |
|                  | Time [hh:mm:ss]            | 0 : 1 : 0                           |

Figure 1: General Settings



General Settings | **Interrupt Settings**

Timer Event Settings

Periodic Events

Days     Months     Years

Hours     Minutes     Seconds

Event trigger: SCU Interrupt    Triggers SCU Interrupt

Event handler: Time\_Handler

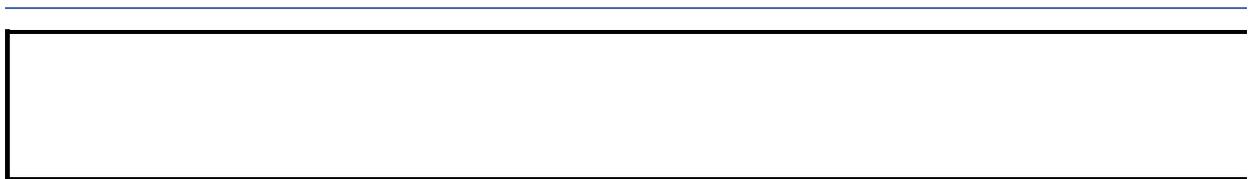
Alarm Event Settings

Enable alarm interrupt

Event trigger: SCU Interrupt    Triggers SCU Interrupt

Event handler: Alarm\_Handler

**Figure 2: Interrupt Settings**



# RTC

[Home](#)

## Enumerations

enum **RTC\_STATUS** { **RTC\_STATUS\_SUCCE**  
= 0U, **RTC\_STATUS\_FAILURE** = 1U }

enum **RTC\_START** { **RTC\_START\_DISABLE**  
0x0U, **RTC\_START\_ENABLE** = 0x1U }

enum **RTC\_INT** {  
    **RTC\_INT\_PERIODIC\_SEC\_DISABLE**  
    0x0U,  
    **RTC\_INT\_PERIODIC\_SEC\_ENABLE** =  
    0x1U,  
    **RTC\_INT\_PERIODIC\_MIN\_DISABLE** =  
    0x0U,  
    **RTC\_INT\_PERIODIC\_MIN\_ENABLE** =  
    0x1U,  
    **RTC\_INT\_PERIODIC\_HOUR\_DISABLE**  
    = 0x0U,  
    **RTC\_INT\_PERIODIC\_HOUR\_ENABLE**  
    0x1U,  
    **RTC\_INT\_PERIODIC\_DAY\_DISABLE** :  
    0x0U,  
    **RTC\_INT\_PERIODIC\_DAY\_ENABLE** =  
    0x1U,  
    **RTC\_INT\_PERIODIC\_MONTH\_DISABLE**  
    = 0x0U,  
    **RTC\_INT\_PERIODIC\_MONTH\_ENABLE**  
    0x1U,  
    **RTC\_INT\_PERIODIC\_YEAR\_DISABLE**  
    0x0U,  
    **RTC\_INT\_PERIODIC\_YEAR\_ENABLE**  
    0x1U,  
    **RTC\_INT\_ALARM\_DISABLE** = 0x0U,

```
RTC_INT_ALARM_ENABLE = 0x1U  
}
```

```
typedef enum RTC_START RTC_START_t
```

```
typedef enum RTC_INT RTC_INT_t
```

```
typedef struct RTC_CONFIG RTC_CONFIG_t
```

```
typedef struct RealTimeClock RTC_t
```

---

## Typedef Documentation

**typedef struct `RTC_CONFIG` `RTC_CONFIG_t`**

RTC APP Handle structure. Stores configured parameters in APP UI.

**typedef enum `RTC_INT` `RTC_INT_t`**

This enumeration defines RTC Module Interrupt Related Events Enable and Disable values.

**typedef enum `RTC_START` `RTC_START_t`**

This enumeration defines RTC Module Enable and Disable values.

**typedef struct `RealTimeClock` `RTC_t`**

Runtime RTC handler.

---

## Enumeration Type Documentation

### enum `RTC_INT`

This enumeration defines RTC Module Interrupt Related Events Enable and Disable values.

#### Enumerator:

|  |  |
|--|--|
| <code>RTC_INT_PERIODIC_SEC_DISABLE</code>  | Timer Periodic<br>Seconds Interrupt<br>Disable |
| <code>RTC_INT_PERIODIC_SEC_ENABLE</code>   | Timer Periodic<br>Seconds Interrupt<br>Enable  |
| <code>RTC_INT_PERIODIC_MIN_DISABLE</code>  | Timer Periodic<br>Minutes Interrupt<br>Disable |
| <code>RTC_INT_PERIODIC_MIN_ENABLE</code>   | Timer Periodic<br>Minutes Interrupt<br>Enable  |
| <code>RTC_INT_PERIODIC_HOUR_DISABLE</code> | Timer Periodic<br>Hours Interrupt<br>Disable   |
| <code>RTC_INT_PERIODIC_HOUR_ENABLE</code>  | Timer Periodic<br>Hours Interrupt<br>Enable    |

|                                       |  |
|---------------------------------------|--|
| <i>RTC_INT_PERIODIC_DAY_DISABLE</i>   | Timer Periodic<br>Days Interrupt<br>Disable  |
| <i>RTC_INT_PERIODIC_DAY_ENABLE</i>    | Timer Periodic<br>Days Interrupt<br>Enable   |
| <i>RTC_INT_PERIODIC_MONTH_DISABLE</i> | Timer Periodic<br>Month Interrupt<br>Disable |
| <i>RTC_INT_PERIODIC_MONTH_ENABLE</i>  | Timer Periodic<br>Month Interrupt<br>Enable  |
| <i>RTC_INT_PERIODIC_YEAR_DISABLE</i>  | Timer Periodic<br>Years Interrupt<br>Disable |
| <i>RTC_INT_PERIODIC_YEAR_ENABLE</i>   | Timer Periodic<br>Years Interrupt<br>Enable  |
| <i>RTC_INT_ALARM_DISABLE</i>          | Alarm Interrupt<br>Disable                   |
| <i>RTC_INT_ALARM_ENABLE</i>           | Alarm Interrupt<br>Enable                    |

Definition at line **104** of file **RTC.h**.

### **enum RTC\_START**

This enumeration defines RTC Module Enable and Disable values.

#### **Enumerator:**

*RTC\_START\_DISABLE* Disable RTC start

*RTC\_START\_ENABLE* Enable RTC start

Definition at line **92** of file **RTC.h**.

### **enum RTC\_STATUS**

#### **Enumerator:**

*RTC\_STATUS\_SUCCESS* APP initialization status success

*RTC\_STATUS\_FAILURE* APP initialization status failure

Definition at line **82** of file **RTC.h**.



# RTC

[Home](#)

[Data Structures](#)

## Data structures

## Data Structures

```
struct RTC_CONFIG  
typedef struct RTC_CONFIG RTC_CONFIG_t
```

## Typedef Documentation

**typedef struct [RTC\\_CONFIG](#) [RTC\\_CONFIG\\_t](#)**

RTC APP Handle structure. Stores configured parameters in APP UI.



# RTC

[Home](#)

## Methods

|                      |   |
|----------------------|---|
| DAVE_APP_VERSION_t   | <b>RTC_GetAppVersion</b> (void)<br>Get RTC APP version.   |
| RTC_STATUS_t         | <b>RTC_Init</b> (RTC_t *const handle)<br>Initializes RTC module.                                      |
| RTC_STATUS_t         | <b>RTC_Time</b> (time_t *time_value)<br>Returns the time in seconds since the epoch time(01/01/1970). |
| __STATIC_INLINE void | <b>RTC_Start</b> (void)<br>Starts the RTC running.  |
| __STATIC_INLINE void | <b>RTC_Stop</b> (void)<br>Stops the RTC running.  |
| RTC_STATUS_t         | <b>RTC_SetTime</b> (XMC_RTC_TIME_t *time)<br>Sets the time.   |
| RTC_STATUS_t         | <b>RTC_SetAlarmTime</b><br>(XMC_RTC_ALARM_t *alarm)<br>Sets the alarm time.                           |
| void                 | <b>RTC_GetTime</b> (XMC_RTC_TIME_t *time)<br>Gets the time.   |
| void                 | <b>RTC_GetAlarmTime</b><br>(XMC_RTC_ALARM_t *alarm)<br>Gets the alarm time.                           |
| uint32_t             | <b>RTC_GetFlagStatus</b> (void)<br>Gets the flag status.  |

## Methods

## Function Documentation

**void** `RTC_GetAlarmTime` (`XMC_RTC_ALARM_t` \* `alarm` )

Gets the alarm time.

**Parameters:**

**time** Pointer to structure of type `XMC_RTC_ALARM_t`

**Returns:**

None

**Example:** Pre-requisite: Instantiate RTC APP and generate code before copying below code snippet.

```
#include <DAVE.h>
XMC_RTC_ALARM_t alarm =
{
    .seconds = 0U,
    .minutes = 1U,
    .hours = 0U,
    .days = 1U,
    .month = 1,
    .year = 1970U
};
int main(void)
{
    RTC_STATUS_t status = RTC_STATUS_FAILURE;
    DAVE_Init();
    RTC_Stop();
    status = RTC_SetAlarmTime(&alarm);
    if(status == RTC_STATUS_SUCCESS)
    {
        RTC_GetAlarmTime(&alarm);
    }
    // ... infinite loop ...
}
```

```
while(1)
{
}
```

Definition at line **354** of file **RTC.c**.

## **DAVE\_APP\_VERSION\_t** **RTC\_GetAppVersion ( void )**

Get RTC APP version.

### **Returns:**

DAVE\_APP\_VERSION\_t APP version information (major, minor and patch number)

### **Description:**

The function can be used to check application software compatibility with a specific version of the APP.

```
#include <DAVE.h>
#include <DAVE_common.h>
int main(void)
{
    DAVE_APP_VERSION_t version;
    DAVE_Init();
    version = RTC_GetAppVersion();
    if(version.major != 4U)
    {
    }
    return 0;
}
```

Definition at line **110** of file **RTC.c**.

## **uint32\_t RTC\_GetFlagStatus ( void )**

Gets the flag status.

### **Parameters:**

**None**

### **Returns:**

uint32\_t flag status

**Example:** Pre-requisite: Instantiate RTC APP and generate code before copying below code snippet.

```
#include <DAVE.h>
int main(void)
{
    DAVE_Init(); // RTC_Init(&RTC_0); is called in
side DAVE_Init()
    if(RTC_GetFlagStatus() & RTC_STSSR_SPSE_Msk) //
check if the periodic seconds event has occurred.

    {
    }
    while(1)
    {}
}
```

Definition at line **367** of file **RTC.c**.

## **void RTC\_GetTime ( XMC\_RTC\_TIME\_t \* time )**

Gets the time.

**Parameters:**

**time** Pointer to structure of type XMC\_RTC\_TIME\_t

**Returns:**

None

**Example:** Pre-requisite: Instantiate RTC APP and generate code before copying below code snippet.

```
#include <DAVE.h>
XMC_RTC_TIME_t timeval =
{
    .seconds = 0U,
    .minutes = 0U,
    .hours = 0U,
    .days = 1U,
    .month = 1,
    .year = 1970U
};
int main(void)
{
    RTC_STATUS_t status = RTC_STATUS_FAILURE;
    DAVE_Init();
    RTC_Stop();
    status = RTC_SetTime(&timeval);
    if(status == RTC_STATUS_SUCCESS)
    {
        RTC_GetTime(&timeval);
    }
    // ... infinite loop ...
    while(1)
    {}
}
```

Definition at line **307** of file **RTC.c**.

## **RTC\_STATUS\_t RTC\_Init ( RTC\_t \*const handle )**

Initializes RTC module.

### **Parameters:**

**handle** Constant pointer to RTC structure of type **RTC\_t**

### **Returns:**

RTC\_STATUS\_t

### **Description:**

Configures the RTC module registers as per settings updated in UI and enables the RTC module.

**Example:** Pre-requisite: Instantiate RTC APP and generate code before copying below code snippet.

```
#include <DAVE.h>
int main(void)
{
    DAVE_Init(); // RTC_Init(&RTC_0) will be called within DAVE_Init()
    // ... infinite loop ...
    while(1)
    {}
}
```

Definition at line **125** of file **RTC.c**.

References **RealTimeClock::config**, **RealTimeClock::initialized**, **RTC\_Start()**, **RTC\_START\_ENABLE**, **RTC\_STATUS\_FAILURE**, **RTC\_STATUS\_SUCCESS**, and **RTC\_CONFIG::start**.

## RTC\_STATUS\_t RTC\_SetAlarmTime (XMC\_RTC\_ALARM\_t\* alarm

Sets the alarm time.

### Parameters:

**alarm** Pointer to structure of type XMC\_RTC\_ALARM\_t

### Returns:

RTC\_STATUS\_t

**Example:** Pre-requisite: Instantiate RTC APP and generate code before copying below code snippet.

```
#include <DAVE.h>
XMC_RTC_ALARM_t alarm =
{
    .seconds = 0U,
    .minutes = 1U,
    .hours = 0U,
    .days = 1U,
    .month = 1,
    .year = 1970U
};
int main(void)
{
    RTC_STATUS_t status = RTC_STATUS_FAILURE;
    DAVE_Init();
    RTC_Stop();
    status = RTC_SetAlarmTime(&alarm);
    if(status != RTC_STATUS_SUCCESS)
    {
        //error
    }
    // ... infinite loop ...
    while(1)
    {}
}
```

Definition at line [320](#) of file [RTC.c](#).

References [RTC\\_STATUS\\_FAILURE](#), and [RTC\\_STATUS\\_SUCCESS](#).

### **RTC\_STATUS\_t** [RTC\\_SetTime](#) ( XMC\_RTC\_TIME\_t \* **time** )

Sets the time.

#### **Parameters:**

**time** Pointer to structure of type XMC\_RTC\_TIME\_t

#### **Returns:**

RTC\_STATUS\_t

**Example:** Pre-requisite: Instantiate RTC APP and generate code before copying below code snippet.

```
#include <DAVE.h>
XMC_RTC_TIME_t timeval =
{
    .seconds = 0U,
    .minutes = 0U,
    .hours = 0U,
    .days = 1U,
    .month = 1,
    .year = 1970U
};
int main(void)
{
    RTC_STATUS_t status = RTC_STATUS_FAILURE;
    DAVE_Init();
    RTC_Stop();
    status = RTC_SetTime(&timeval);
}
```

```
if(status != RTC_STATUS_SUCCESS)
{
    //error
}
// ... infinite loop ...
while(1)
{}
}
```

Definition at line [274](#) of file [RTC.c](#).

References [RTC\\_STATUS\\_FAILURE](#), and [RTC\\_STATUS\\_SUCCESS](#).

## `__STATIC_INLINE void RTC_Start (void )`

Starts the RTC running.

### **Returns:**

None

### **Description:**

Before verifying this API it is required to deselect "Start after initialization" checkbox in UI. Thereby this API shall be used to start the RTC module.

**Example:** Pre-requisite: 1. Instantiate RTC APP  
2. Unselect "Start after initialization checkbox in UI"  
3. Generate code before copying below code snippet.

```
#include <DAVE.h>
int main(void)
{
    DAVE_Init();// RTC_Init(&RTC_0) will be called
    within DAVE_Init()
```

```
RTC_Stop();
if (XMC_RTC_IsRunning() == false)
{
    RTC_Start();
}
// ... infinite loop ...
while(1)
{}
}
```

Definition at line **288** of file **RTC.h**.

Referenced by **RTC\_Init()**.

### **\_\_STATIC\_INLINE void RTC\_Stop ( void )**

Stops the RTC running.

**Returns:**

None

**Example:** Pre-requisite: Instantiate RTC APP  
Enable periodic seconds event in "Interrupt Settings" tab.  
Generate code before copying below code snippet.

```
#include <DAVE.h>

void Time_Handler(void)
{
    static uint32_t seconds = 0;
    seconds++;

    if(seconds == 30)
    {
        RTC_Stop();
    }
}
```

```

    }
}
int main(void)
{
    DAVE_Init(); // RTC_Init(&RTC_0) will be called within DAVE_Init()
    // ... infinite loop ...
    while(1)
    {}
}

```

Definition at line [324](#) of file [RTC.h](#).

### **RTC\_STATUS\_t** [RTC\\_Time](#) (time\_t \* time\_value)

Returns the time in seconds since the epoch time(01/01/1970).

#### **Parameters:**

**time\_value** Pointer to structure of type time\_t

#### **Returns:**

RTC\_STATUS\_t

**Example:** Pre-requisite: Instantiate RTC APP and generate code before copying below code snippet.

```

#include <DAVE.h>
int main(void)
{
    RTC_STATUS_t status = RTC_STATUS_FAILURE;
    time_t Time_Sec;
    DAVE_Init(); // RTC_Init(&RTC_0) will be called within DAVE_Init()
    status = RTC_Time(&Time_Sec);
}

```

```
if(status != RTC_STATUS_SUCCESS)
{
    //error
}
// ... infinite loop ...
while(1)
{}
}
```

Definition at line **380** of file **RTC.c**.

References **RTC\_STATUS\_FAILURE**, and **RTC\_STATUS\_SUCCESS**.

# RTC

Home

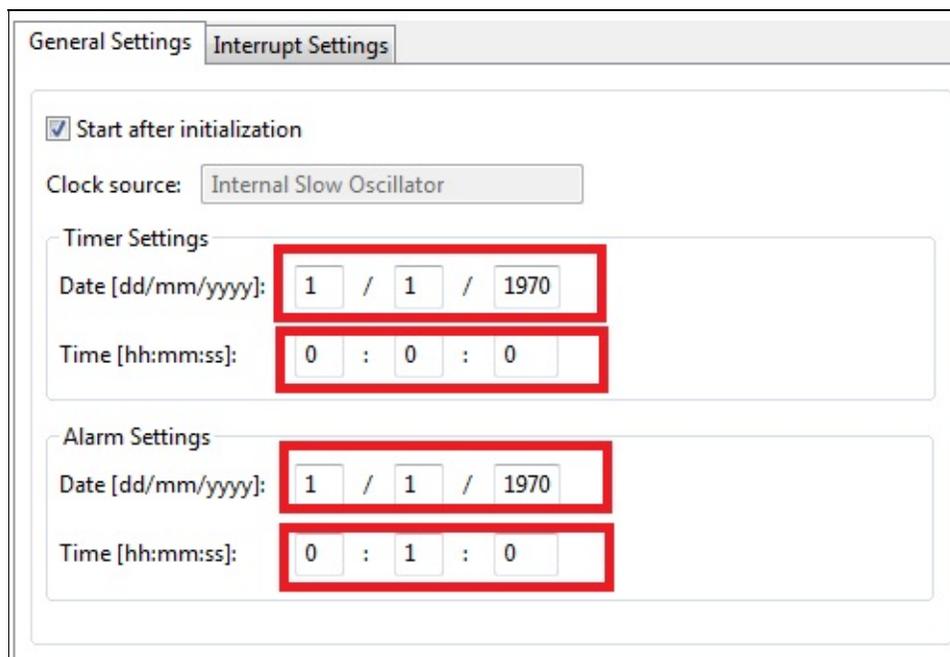
## Usage

### Usage

**Example1:** This example demonstrates how to use the RTC APP to generate a periodic (second) interrupt and an alarm interrupt using XMC4500 hexagon board.

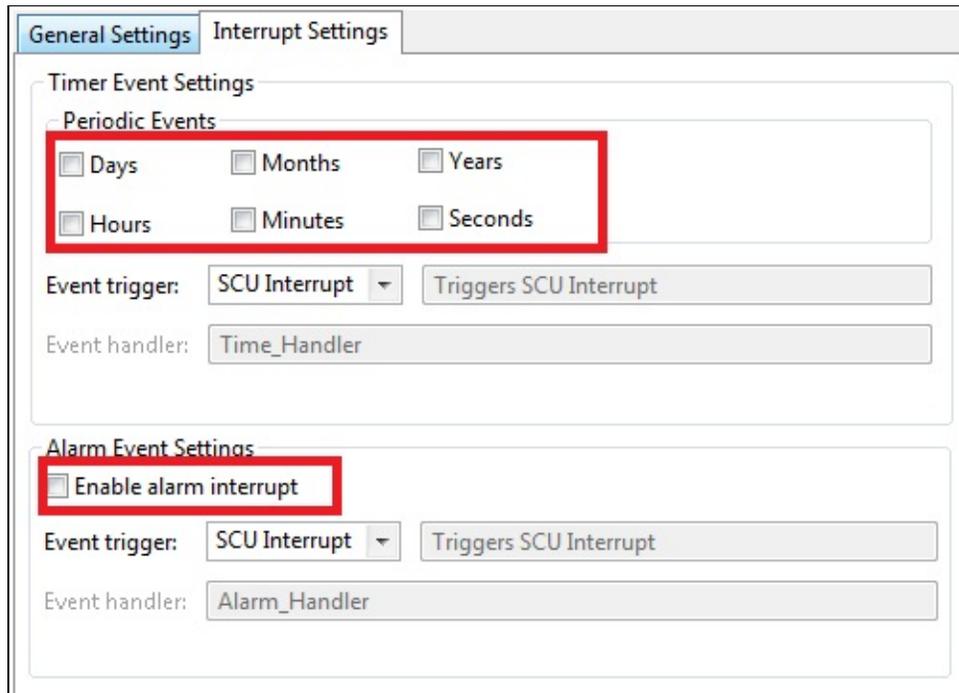
Instantiate the RTC APP and update its GUI with the following configuration

### Configure the APP



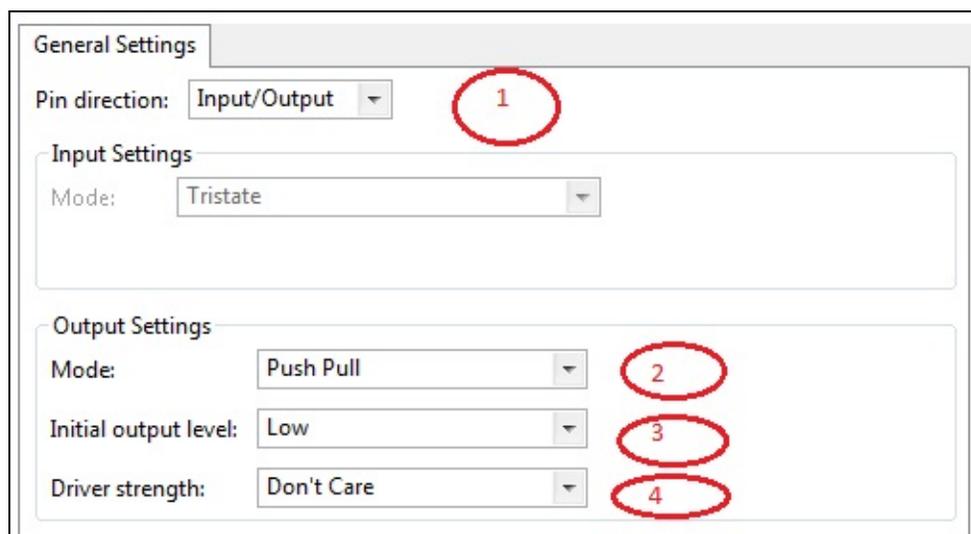
The screenshot shows the configuration window for the RTC APP, with the 'Interrupt Settings' tab selected. The 'Start after initialization' checkbox is checked. The 'Clock source' is set to 'Internal Slow Oscillator'. The 'Timer Settings' section has a date of 1/1/1970 and a time of 00:00:00. The 'Alarm Settings' section has a date of 1/1/1970 and a time of 00:01:00. Red boxes highlight the date and time input fields for both the timer and alarm settings.

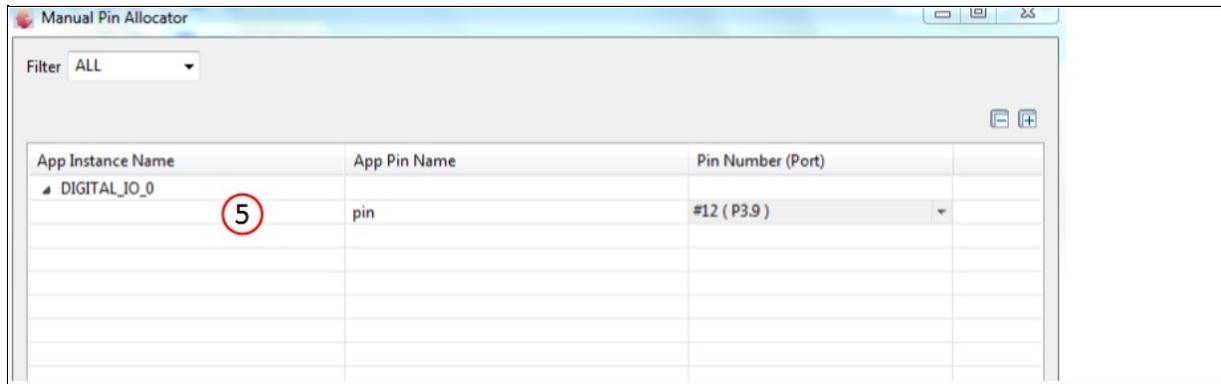
1. Enter date and time  
**Note:** By default date is 01/01/1970 and time is 00:00:00.
2. Enter Alarm date and alarm time  
**Note:** By default date is 01/01/1970 and time is 00:01:00.



1. Select "seconds" check box in the periodic events group
2. Select the check box for "Enable alarm interrupt"

Instantiate DIGITAL\_IO APP and set the following configuration:





## Generate code

Files are generated here: ``<project_name>/Dave/Generated/`' (`<project_name>` is the name chosen by the user during project creation). APP instance definitions and APIs are generated only after code generation.

- **Note:** Code must be explicitly generated for every change in the GUI configuration.

**Important:** Any manual modification to these generated code files will be overwritten by a subsequent code generation operation.

## Sample Application (main.c)

```
#include <DAVE.h>

void Time_Handler(void)
{
    DIGITAL_IO_ToggleOutput(&DIGITAL_IO_0);
}

void Alarm_Handler(void)
{
    XMC_RTC_ALARM_t alarm_time;

    RTC_GetAlarmTime(&alarm_time);    // Read the
current alarm time
```

```

    if(++alarm_time.minutes > 59)
    {
        alarm_time.minutes = 0;
        alarm_time.hours++;
    }

    RTC_Stop();
    RTC_SetAlarmTime(&alarm_time);    // Reconfigure
e alarm time for next minute
    RTC_Start();
}

int main(void)
{
    RTC_Stop();
    DAVE_Init(); // RTC_Init(&RTC_0) is called insi
de DAVE_Init
    while(1)
    {}
    return 0;
}

```

## Build and Run the Project

### Observation

every second, RTC toggles the P3.9 on the XMC4500 hexagon board.  
Keep breakpoint in Alarm\_Handler function.

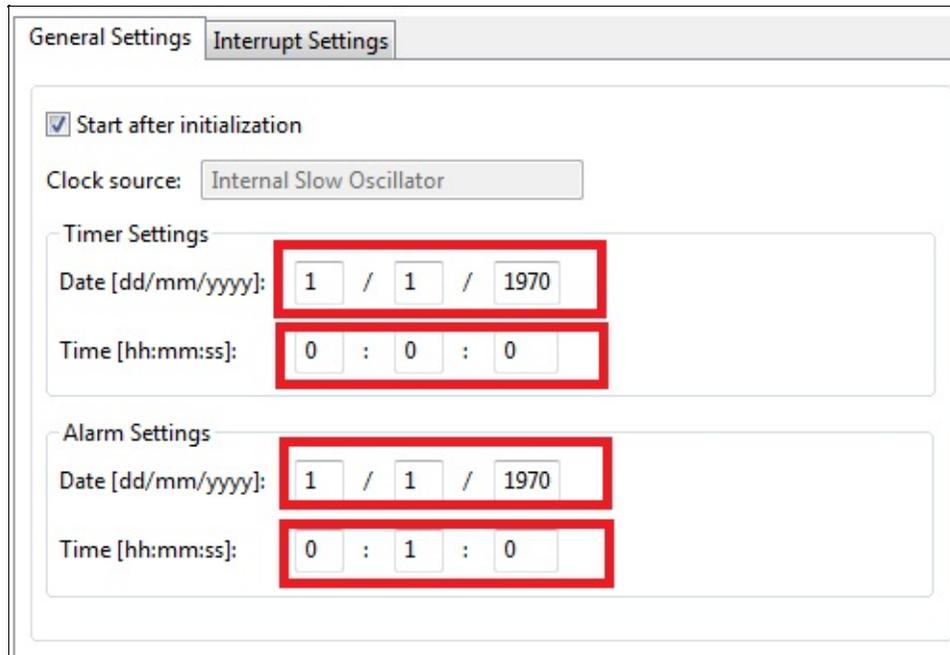
After 60 seconds, program halts in Alarm\_Handler function.

### Example2:

This example demonstrates how to use the RTC alarm event to trigger the NMI(Non Maskable Interrupt) on XMC4500/XMC4800 hexagon board.

Instantiate the RTC APP and update its GUI with the following configuration

## Configure the APP



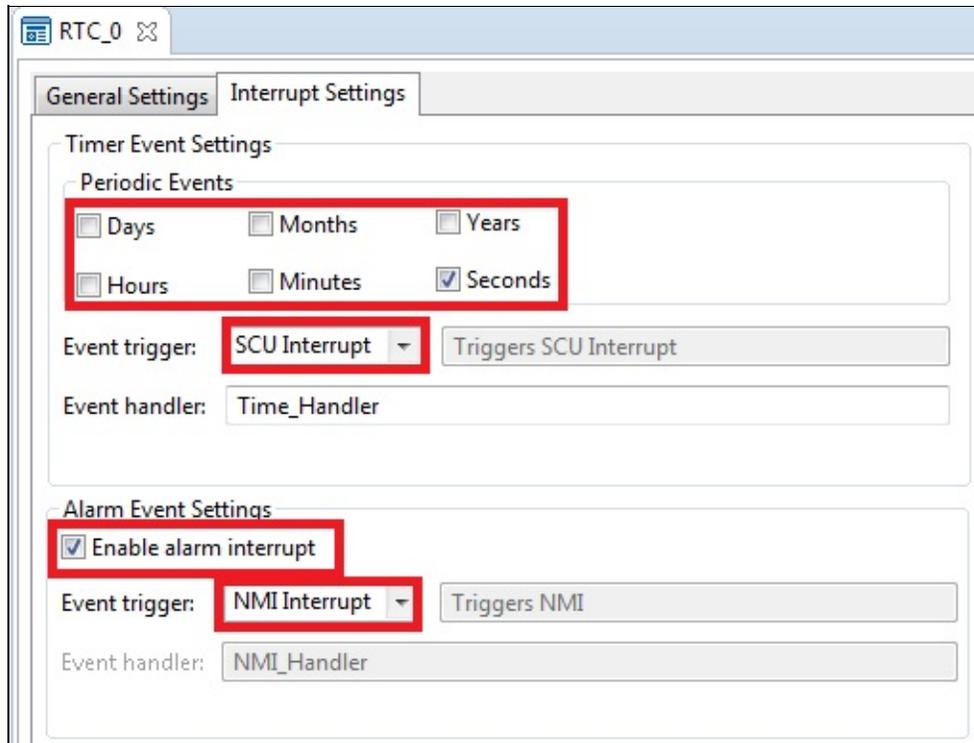
The screenshot shows the configuration interface for the RTC APP, specifically the 'Interrupt Settings' tab. The 'Start after initialization' checkbox is checked. The 'Clock source' is set to 'Internal Slow Oscillator'. Under 'Timer Settings', the 'Date [dd/mm/yyyy]' is set to '1 / 1 / 1970' and the 'Time [hh:mm:ss]' is set to '0 : 0 : 0'. Under 'Alarm Settings', the 'Date [dd/mm/yyyy]' is set to '1 / 1 / 1970' and the 'Time [hh:mm:ss]' is set to '0 : 1 : 0'. Red boxes highlight the date and time input fields for both the timer and alarm settings.

1. Enter date and time

**Note:** By default date is 01/01/1970 and time is 00:00:00.

2. Enter Alarm date and alarm time

**Note:** By default date is 01/01/1970 and time is 00:01:00.



**Timer event settings:**

Select "seconds" check box in the periodic events group  
Select "SCU Interrupt" option for the Event trigger

**Alarm event settings:**

Select the check box for "Enable alarm interrupt"  
Select "NMI Interrupt" option for the Event trigger

Instantiate DIGITAL\_IO APP and set the following configuration:

General Settings

Pin direction: Input/Output 1

Input Settings

Mode: Tristate

Output Settings

Mode: Push Pull 2

Initial output level: Low 3

Driver strength: Don't Care 4

Manual Pin Allocator

Filter: ALL

| App Instance Name  | App Pin Name | Pin Number (Port) |
|--|--------------|-------------------|
| <ul style="list-style-type: none"> <li>▲ DIGITAL_IO_0 <span style="border: 1px solid red; border-radius: 50%; padding: 2px;">5</span></li> </ul> | pin          | #12 ( P3.9)       |
|  |              |                   |
|  |              |                   |
|  |              |                   |
|  |              |                   |
|  |              |                   |
|  |              |                   |
|  |              |                   |
|  |              |                   |
|  |              |                   |

## Generate code

Files are generated here: ``<project_name>/Dave/Generated/`' (`<project_name>` is the name chosen by the user during project creation). APP instance definitions and APIs are generated only after code generation.

- **Note:** Code must be explicitly generated for every change in the GUI configuration.

**Important:** Any manual modification to these generated code files will be overwritten by a subsequent code generation operation.

## Sample Application (main.c)

```
#include <DAVE.h>
```

```

void Time_Handler(void)
{
    DIGITAL_IO_ToggleOutput(&DIGITAL_IO_0);
}

void NMI_Handler(void)
{
    XMC_RTC_ALARM_t alarm_time;

    XMC_SCU_INTERRUPT_ClearEventStatus((XMC_SCU_INTERRUPT_EVENT_t)XMC_SCU_INTERRUPT_EVENT_RTC_ALARM);

    RTC_GetAlarmTime(&alarm_time);    // Read the
current alarm time

    if(++alarm_time.minutes > 59)
    {
        alarm_time.minutes = 0;
        alarm_time.hours++;
    }

    RTC_Stop();
    RTC_SetAlarmTime(&alarm_time);    // Reconfigure
e alarm time for next minute
    RTC_Start();
}

int main(void)
{
    RTC_Stop();
    DAVE_Init(); // RTC_Init(&RTC_0) is called inside DAVE_Init
    while(1)
    {}
    return 0;
}

```

## **Build and Run the Project**

### **Observation**

every second, RTC toggles the P3.9 on the XMC4500 hexagon board.

Keep breakpoint in NMI\_Handler function.

After 60 seconds, program halts in NMI\_Handler function and reconfigures alarm for next minute.

So every minute the NMI hadler gets executed.



# RTC

Home

## Release History

### Release History

|  |
|--|
|  |
|--|

|  |
|--|
|  |
|--|

# RTC

Home

Data Structures

Data Structure Index

Data Fields

## Data Structures

Here are the data structures with brief descriptions:

RealTimeClock

RTC\_CONFIG

|  |
|--|
|  |
|--|

# RTC

[Home](#)

[Data Structures](#)

[Data Structure Index](#)

[Data Fields](#)

[Data Fields](#)

## RealTimeClock Struct Reference

---

## Detailed Description

Runtime RTC handler.

Definition at line **161** of file **RTC.h**.

```
#include <RTC.h>
```

## Data Fields

|  |
|--|
| const <b>RTC_CONFIG_t</b> *const <b>config</b> |
| bool <b>initialized</b>                        |

## Field Documentation

**const RTC\_CONFIG\_t\* const RealTimeClock::config**

Pointer to RTC LLD configuration data

Definition at line **164** of file **RTC.h**.

Referenced by **RTC\_Init()**.

**bool RealTimeClock::initialized**

Pointer to configuration data

Definition at line **165** of file **RTC.h**.

Referenced by **RTC\_Init()**.

---

The documentation for this struct was generated from the following file:

- **RTC.h**



# RTC

[Home](#)

[Data Structures](#)

[Data Structure Index](#)

[Data Fields](#)

[Data Fields](#)

## RTC\_CONFIG Struct Reference

[Data structures](#)

---

## Detailed Description

RTC APP Handle structure. Stores configured parameters in APP UI.

Definition at line [134](#) of file [RTC.h](#).

```
#include <RTC.h>
```

## Data Fields

|                    |                            |
|--------------------|----------------------------|
| <b>RTC_START_t</b> | <b>start</b>               |
| <b>RTC_INT_t</b>   | <b>periodic_sec_intr</b>   |
| <b>RTC_INT_t</b>   | <b>periodic_min_intr</b>   |
| <b>RTC_INT_t</b>   | <b>periodic_hour_intr</b>  |
| <b>RTC_INT_t</b>   | <b>periodic_day_intr</b>   |
| <b>RTC_INT_t</b>   | <b>periodic_month_intr</b> |
| <b>RTC_INT_t</b>   | <b>periodic_year_intr</b>  |
| <b>RTC_INT_t</b>   | <b>alarm_intr</b>          |

## Field Documentation

### **RTC\_INT\_t RTC\_CONFIG::alarm\_intr**

alarm interrupt

Definition at line **145** of file **RTC.h**.

### **RTC\_INT\_t RTC\_CONFIG::periodic\_day\_intr**

periodic days interrupt

Definition at line **141** of file **RTC.h**.

### **RTC\_INT\_t RTC\_CONFIG::periodic\_hour\_intr**

periodic hours interrupt

Definition at line **140** of file **RTC.h**.

### **RTC\_INT\_t RTC\_CONFIG::periodic\_min\_intr**

periodic minutes interrupt

Definition at line **139** of file **RTC.h**.

### **RTC\_INT\_t RTC\_CONFIG::periodic\_month\_intr**

periodic months interrupt

Definition at line **142** of file **RTC.h**.

### **RTC\_INT\_t RTC\_CONFIG::periodic\_sec\_intr**

periodic seconds interrupt

Definition at line **138** of file **RTC.h**.

### **RTC\_INT\_t RTC\_CONFIG::periodic\_year\_intr**

periodic years interrupt

Definition at line **143** of file **RTC.h**.

### **RTC\_START\_t RTC\_CONFIG::start**

rtc start

Definition at line **136** of file **RTC.h**.

Referenced by **RTC\_Init()**.

The documentation for this struct was generated from the following file:

- **RTC.h**



# RTC

Home

Data Structures

Data Structure Index

Data Fields

## Data Structure Index

R

R

RTC\_CONFIG

RealTimeClock

R

# RTC

|                                 |                                      |                             |  |
|---------------------------------|--------------------------------------|-----------------------------|--|
| <a href="#">Home</a>            |                                      |                             |  |
| <a href="#">Data Structures</a> | <a href="#">Data Structure Index</a> | <a href="#">Data Fields</a> |  |
| <a href="#">All</a>             | <a href="#">Variables</a>            |                             |  |

Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- alarm\_intr : [RTC\\_CONFIG](#)
- config : [RealTimeClock](#)
- initialized : [RealTimeClock](#)
- periodic\_day\_intr : [RTC\\_CONFIG](#)
- periodic\_hour\_intr : [RTC\\_CONFIG](#)
- periodic\_min\_intr : [RTC\\_CONFIG](#)
- periodic\_month\_intr : [RTC\\_CONFIG](#)
- periodic\_sec\_intr : [RTC\\_CONFIG](#)
- periodic\_year\_intr : [RTC\\_CONFIG](#)
- start : [RTC\\_CONFIG](#)



# RTC

|                                 |                                      |                             |  |
|---------------------------------|--------------------------------------|-----------------------------|--|
| <a href="#">Home</a>            |                                      |                             |  |
| <a href="#">Data Structures</a> | <a href="#">Data Structure Index</a> | <a href="#">Data Fields</a> |  |
| <a href="#">All</a>             | <a href="#">Variables</a>            |                             |  |

- alarm\_intr : [RTC\\_CONFIG](#)
- config : [RealTimeClock](#)
- initialized : [RealTimeClock](#)
- periodic\_day\_intr : [RTC\\_CONFIG](#)
- periodic\_hour\_intr : [RTC\\_CONFIG](#)
- periodic\_min\_intr : [RTC\\_CONFIG](#)
- periodic\_month\_intr : [RTC\\_CONFIG](#)
- periodic\_sec\_intr : [RTC\\_CONFIG](#)
- periodic\_year\_intr : [RTC\\_CONFIG](#)
- start : [RTC\\_CONFIG](#)



# RTC

|           |         |  |
|-----------|---------|--|
| Home      |         |  |
| File List | Globals |  |

## File List

Here is a list of all documented files with brief descriptions:

|                              |  |
|------------------------------|--|
| <a href="#">RTC.c [code]</a> | Real Time Clock APP Implementation Source File |
| <a href="#">RTC.h [code]</a> |  |

|  |
|--|
|  |
|--|

# RTC

|             |         |  |
|-------------|---------|--|
| <b>Home</b> |         |  |
| File List   | Globals |  |

Functions

## RTC.c File Reference

Real Time Clock APP Implementation Source File. [More...](#)

---

## Detailed Description

Real Time Clock APP Implementation Source File.

**Date:**

2015-11-18

NOTE: This file is generated by DAVE. Any manual modification done to this file will be lost when the code is regenerated.

Definition in file **RTC.c**.

```
#include "rtc.h"
```

## Functions

|                    |   |
|--------------------|---|
| DAVE_APP_VERSION_t | <b>RTC_GetAppVersion</b> (void)<br>Get RTC APP version.   |
| RTC_STATUS_t       | <b>RTC_Init</b> (RTC_t *const handler)<br>Initializes RTC module.                                     |
| RTC_STATUS_t       | <b>RTC_SetTime</b> (XMC_RTC_TIME_t *current_time)<br>Sets the time.                                   |
| void               | <b>RTC_GetTime</b> (XMC_RTC_TIME_t *current_time)<br>Gets the time.                                   |
| RTC_STATUS_t       | <b>RTC_SetAlarmTime</b> (XMC_RTC_ALARM_t *alarm)<br>Sets the alarm time.                              |
| void               | <b>RTC_GetAlarmTime</b> (XMC_RTC_ALARM_t *alarm)<br>Gets the alarm time.                              |
| uint32_t           | <b>RTC_GetFlagStatus</b> (void)<br>Gets the flag status.  |
| RTC_STATUS_t       | <b>RTC_Time</b> (time_t *time_value)<br>Returns the time in seconds since the epoch time(01/01/1970). |

## Function Documentation

**void** `RTC_GetAlarmTime` (`XMC_RTC_ALARM_t` \* `alarm` )

Gets the alarm time.

**Parameters:**

**time** Pointer to structure of type `XMC_RTC_ALARM_t`

**Returns:**

None

**Example:** Pre-requisite: Instantiate RTC APP and generate code before copying below code snippet.

```
#include <DAVE.h>
XMC_RTC_ALARM_t alarm =
{
    .seconds = 0U,
    .minutes = 1U,
    .hours = 0U,
    .days = 1U,
    .month = 1,
    .year = 1970U
};
int main(void)
{
    RTC_STATUS_t status = RTC_STATUS_FAILURE;
    DAVE_Init();
    RTC_Stop();
    status = RTC_SetAlarmTime(&alarm);
    if(status == RTC_STATUS_SUCCESS)
    {
        RTC_GetAlarmTime(&alarm);
    }
    // ... infinite loop ...
}
```

```
while(1)
{
}
```

Definition at line **354** of file **RTC.c**.

## **uint32\_t RTC\_GetFlagStatus ( void )**

Gets the flag status.

### **Parameters:**

**None**

### **Returns:**

uint32\_t flag status

**Example:** Pre-requisite: Instantiate RTC APP and generate code before copying below code snippet.

```
#include <DAVE.h>
int main(void)
{
    DAVE_Init(); // RTC_Init(&RTC_0); is called in
side DAVE_Init()
    if(RTC_GetFlagStatus() & RTC_STSSR_SPSE_Msk) //
check if the periodic seconds event has occurred.

    {
    }
    while(1)
    {}
}
```

Definition at line **367** of file **RTC.c**.

**void RTC\_GetTime ( XMC\_RTC\_TIME\_t \* time )**

Gets the time.

**Parameters:**

**time** Pointer to structure of type XMC\_RTC\_TIME\_t

**Returns:**

None

**Example:** Pre-requisite: Instantiate RTC APP and generate code before copying below code snippet.

```
#include <DAVE.h>
XMC_RTC_TIME_t timeval =
{
    .seconds = 0U,
    .minutes = 0U,
    .hours = 0U,
    .days = 1U,
    .month = 1,
    .year = 1970U
};
int main(void)
{
    RTC_STATUS_t status = RTC_STATUS_FAILURE;
    DAVE_Init();
    RTC_Stop();
    status = RTC_SetTime(&timeval);
    if(status == RTC_STATUS_SUCCESS)
    {
        RTC_GetTime(&timeval);
    }
    // ... infinite loop ...
}
```

```
while(1)
{
}
```

Definition at line **307** of file **RTC.c**.

### **RTC\_STATUS\_t RTC\_Init ( RTC\_t \*const handle )**

Initializes RTC module.

#### **Parameters:**

**handle** Constant pointer to RTC structure of type **RTC\_t**

#### **Returns:**

RTC\_STATUS\_t

#### **Description:**

Configures the RTC module registers as per settings updated in UI and enables the RTC module.

**Example:** Pre-requisite: Instantiate RTC APP and generate code before copying below code snippet.

```
#include <DAVE.h>
int main(void)
{
    DAVE_Init(); // RTC_Init(&RTC_0) will be called within DAVE_Init()
    // ... infinite loop ...
    while(1)
    {}
}
```

Definition at line **125** of file **RTC.c**.

References **RealTimeClock::config**, **RealTimeClock::initialized**, **RTC\_Start()**, **RTC\_START\_ENABLE**, **RTC\_STATUS\_FAILURE**, **RTC\_STATUS\_SUCCESS**, and **RTC\_CONFIG::start**.

**RTC\_STATUS\_t** **RTC\_SetAlarmTime** (**XMC\_RTC\_ALARM\_t** \* **alarm**

Sets the alarm time.

**Parameters:**

**alarm** Pointer to structure of type **XMC\_RTC\_ALARM\_t**

**Returns:**

**RTC\_STATUS\_t**

**Example:** Pre-requisite: Instantiate RTC APP and generate code before copying below code snippet.

```
#include <DAVE.h>
XMC_RTC_ALARM_t alarm =
{
    .seconds = 0U,
    .minutes = 1U,
    .hours = 0U,
    .days = 1U,
    .month = 1,
    .year = 1970U
};
int main(void)
{
    RTC_STATUS_t status = RTC_STATUS_FAILURE;
    DAVE_Init();
    RTC_Stop();
    status = RTC_SetAlarmTime(&alarm);
    if(status != RTC_STATUS_SUCCESS)
```

```
{
    //error
}
// ... infinite loop ...
while(1)
{}
}
```

Definition at line **320** of file **RTC.c**.

References **RTC\_STATUS\_FAILURE**, and **RTC\_STATUS\_SUCCESS**.

**RTC\_STATUS\_t RTC\_SetTime ( XMC\_RTC\_TIME\_t \* time )**

Sets the time.

**Parameters:**

**time** Pointer to structure of type XMC\_RTC\_TIME\_t

**Returns:**

RTC\_STATUS\_t

**Example:** Pre-requisite: Instantiate RTC APP and generate code before copying below code snippet.

```
#include <DAVE.h>
XMC_RTC_TIME_t timeval =
{
    .seconds = 0U,
    .minutes = 0U,
    .hours = 0U,
    .days = 1U,
    .month = 1,
```

```

        .year = 1970U
    };
    int main(void)
    {
        RTC_STATUS_t status = RTC_STATUS_FAILURE;
        DAVE_Init();
        RTC_Stop();
        status = RTC_SetTime(&timeval);
        if(status != RTC_STATUS_SUCCESS)
        {
            //error
        }
        // ... infinite loop ...
        while(1)
        {}
    }

```

Definition at line **274** of file **RTC.c**.

References **RTC\_STATUS\_FAILURE**, and **RTC\_STATUS\_SUCCESS**.

### **RTC\_STATUS\_t RTC\_Time (time\_t \* time\_value )**

Returns the time in seconds since the epoch time(01/01/1970).

**Parameters:**

**time\_value** Pointer to structure of type time\_t

**Returns:**

RTC\_STATUS\_t

**Example:** Pre-requisite: Instantiate RTC APP and generate code before copying below code snippet.

```
#include <DAVE.h>
int main(void)
{
    RTC_STATUS_t status = RTC_STATUS_FAILURE;
    time_t Time_Sec;
    DAVE_Init(); // RTC_Init(&RTC_0) will be called within DAVE_Init()
    status = RTC_Time(&Time_Sec);
    if(status != RTC_STATUS_SUCCESS)
    {
        //error
    }
    // ... infinite loop ...
    while(1)
    {}
}
```

Definition at line **380** of file **RTC.c**.

References **RTC\_STATUS\_FAILURE**, and **RTC\_STATUS\_SUCCESS**.

[Go to the source code of this file.](#)

---

# RTC

|                           |                         |  |
|---------------------------|-------------------------|--|
| <a href="#">Home</a>      |                         |  |
| <a href="#">File List</a> | <a href="#">Globals</a> |  |

[Data Structures](#)

## RTC.h File Reference

---

## Detailed Description

**Date:**

2015-05-18

NOTE: This file is generated by DAVE. Any manual modification done to this file will be lost when the code is regenerated.

Definition in file [RTC.h](#).

```
#include <xmc_scu.h> #include <xmc_rtc.h>
#include "rtc_conf.h"
#include <DAVE_Common.h>
#include "rtc_extern.h"
```

## Data Structures

---

struct **RTC\_CONFIG**

struct **RealTimeClock**

---

## Functions

|                      |  |
|----------------------|--|
| DAVE_APP_VERSION_t   | <b>RTC_GetAppVersion</b> (void)<br>Get RTC APP version.  |
| RTC_STATUS_t         | <b>RTC_Init</b> (RTC_t *const handle)<br>Initializes RTC module.   |
| RTC_STATUS_t         | <b>RTC_Time</b> (time_t *time_value)<br>Returns the time in seconds since the epoch time(01/01/1970).    |
| __STATIC_INLINE void | <b>RTC_Start</b> (void)<br>Starts the RTC running.   |
| __STATIC_INLINE void | <b>RTC_Stop</b> (void)<br>Stops the RTC running.   |
| RTC_STATUS_t         | <b>RTC_SetTime</b> (XMC_RTC_TIME_t *tim)<br>Sets the time.   |
| RTC_STATUS_t         | <b>RTC_SetAlarmTime</b><br>(XMC_RTC_ALARM_t *alarm)<br>Sets the alarm time.                              |
| void                 | <b>RTC_GetTime</b> (XMC_RTC_TIME_t *tim)<br>Gets the time.   |
| void                 | <b>RTC_GetAlarmTime</b><br>(XMC_RTC_ALARM_t *alarm)<br>Gets the alarm time.                              |
| uint32_t             | <b>RTC_GetFlagStatus</b> (void)<br>Gets the flag status.   |
| enum                 | <b>RTC_STATUS</b> { <b>RTC_STATUS_SUCCE</b><br>= 0U, <b>RTC_STATUS_FAILURE</b> = 1U }                    |
| enum                 | <b>RTC_START</b> { <b>RTC_START_DISABLE</b><br>0x0U, <b>RTC_START_ENABLE</b> = 0x1U }                    |
|                      | <b>RTC_INT</b> {<br><b>RTC_INT_PERIODIC_SEC_DISABLE</b><br>0x0U,<br><b>RTC_INT_PERIODIC_SEC_ENABLE</b> = |

```

enum
    0x1U,
    RTC_INT_PERIODIC_MIN_DISABLE =
    0x0U,
    RTC_INT_PERIODIC_MIN_ENABLE =
    0x1U,
    RTC_INT_PERIODIC_HOUR_DISABLE
    = 0x0U,
    RTC_INT_PERIODIC_HOUR_ENABLE
    0x1U,
    RTC_INT_PERIODIC_DAY_DISABLE :
    0x0U,
    RTC_INT_PERIODIC_DAY_ENABLE =
    0x1U,
    RTC_INT_PERIODIC_MONTH_DISABLE
    = 0x0U,
    RTC_INT_PERIODIC_MONTH_ENABLE
    0x1U,
    RTC_INT_PERIODIC_YEAR_DISABLE
    0x0U,
    RTC_INT_PERIODIC_YEAR_ENABLE
    0x1U,
    RTC_INT_ALARM_DISABLE = 0x0U,
    RTC_INT_ALARM_ENABLE = 0x1U
}

```

```

typedef enum RTC_START RTC_START_t
typedef enum RTC_INT RTC_INT_t
typedef struct RTC_CONFIG RTC_CONFIG_t
typedef struct RealTimeClock RTC_t

```

[Go to the source code of this file.](#)



# RTC

|                           |                           |                          |                              |                            |  |
|---------------------------|---------------------------|--------------------------|------------------------------|----------------------------|--|
| <a href="#">Home</a>      |                           |                          |                              |                            |  |
| <a href="#">File List</a> | <a href="#">Globals</a>   |                          |                              |                            |  |
| <a href="#">All</a>       | <a href="#">Functions</a> | <a href="#">Typedefs</a> | <a href="#">Enumerations</a> | <a href="#">Enumerator</a> |  |
| r                         |                           |                          |                              |                            |  |

Here is a list of all documented functions, variables, defines, enums, and typedefs with links to the documentation:

- r -

- [RTC\\_CONFIG\\_t](#) : [RTC.h](#)
- [RTC\\_GetAlarmTime\(\)](#) : [RTC.c](#) , [RTC.h](#)
- [RTC\\_GetAppVersion\(\)](#) : [RTC.c](#) , [RTC.h](#)
- [RTC\\_GetFlagStatus\(\)](#) : [RTC.c](#) , [RTC.h](#)
- [RTC\\_GetTime\(\)](#) : [RTC.c](#) , [RTC.h](#)
- [RTC\\_Init\(\)](#) : [RTC.c](#) , [RTC.h](#)
- [RTC\\_INT](#) : [RTC.h](#)
- [RTC\\_INT\\_ALARM\\_DISABLE](#) : [RTC.h](#)
- [RTC\\_INT\\_ALARM\\_ENABLE](#) : [RTC.h](#)
- [RTC\\_INT\\_PERIODIC\\_DAY\\_DISABLE](#) : [RTC.h](#)
- [RTC\\_INT\\_PERIODIC\\_DAY\\_ENABLE](#) : [RTC.h](#)
- [RTC\\_INT\\_PERIODIC\\_HOUR\\_DISABLE](#) : [RTC.h](#)
- [RTC\\_INT\\_PERIODIC\\_HOUR\\_ENABLE](#) : [RTC.h](#)
- [RTC\\_INT\\_PERIODIC\\_MIN\\_DISABLE](#) : [RTC.h](#)
- [RTC\\_INT\\_PERIODIC\\_MIN\\_ENABLE](#) : [RTC.h](#)
- [RTC\\_INT\\_PERIODIC\\_MONTH\\_DISABLE](#) : [RTC.h](#)
- [RTC\\_INT\\_PERIODIC\\_MONTH\\_ENABLE](#) : [RTC.h](#)
- [RTC\\_INT\\_PERIODIC\\_SEC\\_DISABLE](#) : [RTC.h](#)
- [RTC\\_INT\\_PERIODIC\\_SEC\\_ENABLE](#) : [RTC.h](#)
- [RTC\\_INT\\_PERIODIC\\_YEAR\\_DISABLE](#) : [RTC.h](#)
- [RTC\\_INT\\_PERIODIC\\_YEAR\\_ENABLE](#) : [RTC.h](#)
- [RTC\\_INT\\_t](#) : [RTC.h](#)
- [RTC\\_SetAlarmTime\(\)](#) : [RTC.c](#) , [RTC.h](#)

- `RTC_SetTime()` : **RTC.h** , **RTC.c**
- `RTC_Start()` : **RTC.h**
- `RTC_START` : **RTC.h**
- `RTC_START_DISABLE` : **RTC.h**
- `RTC_START_ENABLE` : **RTC.h**
- `RTC_START_t` : **RTC.h**
- `RTC_STATUS` : **RTC.h**
- `RTC_STATUS_FAILURE` : **RTC.h**
- `RTC_STATUS_SUCCESS` : **RTC.h**
- `RTC_Stop()` : **RTC.h**
- `RTC_t` : **RTC.h**
- `RTC_Time()` : **RTC.c** , **RTC.h**



# RTC

|                           |                           |                          |                              |                            |  |
|---------------------------|---------------------------|--------------------------|------------------------------|----------------------------|--|
| <a href="#">Home</a>      |                           |                          |                              |                            |  |
| <a href="#">File List</a> | <a href="#">Globals</a>   |                          |                              |                            |  |
| <a href="#">All</a>       | <a href="#">Functions</a> | <a href="#">Typedefs</a> | <a href="#">Enumerations</a> | <a href="#">Enumerator</a> |  |

- [RTC\\_GetAlarmTime\(\)](#) : [RTC.c](#) , [RTC.h](#)
- [RTC\\_GetAppVersion\(\)](#) : [RTC.h](#) , [RTC.c](#)
- [RTC\\_GetFlagStatus\(\)](#) : [RTC.c](#) , [RTC.h](#)
- [RTC\\_GetTime\(\)](#) : [RTC.h](#) , [RTC.c](#)
- [RTC\\_Init\(\)](#) : [RTC.c](#) , [RTC.h](#)
- [RTC\\_SetAlarmTime\(\)](#) : [RTC.c](#) , [RTC.h](#)
- [RTC\\_SetTime\(\)](#) : [RTC.c](#) , [RTC.h](#)
- [RTC\\_Start\(\)](#) : [RTC.h](#)
- [RTC\\_Stop\(\)](#) : [RTC.h](#)
- [RTC\\_Time\(\)](#) : [RTC.h](#) , [RTC.c](#)



# RTC

|           |           |          |              |            |  |
|-----------|-----------|----------|--------------|------------|--|
| Home      |           |          |              |            |  |
| File List | Globals   |          |              |            |  |
| All       | Functions | Typedefs | Enumerations | Enumerator |  |

- [RTC\\_CONFIG\\_t](#): [RTC.h](#)
- [RTC\\_INT\\_t](#): [RTC.h](#)
- [RTC\\_START\\_t](#): [RTC.h](#)
- [RTC\\_t](#): [RTC.h](#)



# RTC

|           |           |          |              |            |  |
|-----------|-----------|----------|--------------|------------|--|
| Home      |           |          |              |            |  |
| File List |           | Globals  |              |            |  |
| All       | Functions | Typedefs | Enumerations | Enumerator |  |

- RTC\_INT : [RTC.h](#)
- RTC\_START : [RTC.h](#)
- RTC\_STATUS : [RTC.h](#)



# RTC

|           |           |          |              |            |
|-----------|-----------|----------|--------------|------------|
| Home      |           |          |              |            |
| File List | Globals   |          |              |            |
| All       | Functions | Typedefs | Enumerations | Enumerator |

- [RTC\\_INT\\_ALARM\\_DISABLE](#) : [RTC.h](#)
- [RTC\\_INT\\_ALARM\\_ENABLE](#) : [RTC.h](#)
- [RTC\\_INT\\_PERIODIC\\_DAY\\_DISABLE](#) : [RTC.h](#)
- [RTC\\_INT\\_PERIODIC\\_DAY\\_ENABLE](#) : [RTC.h](#)
- [RTC\\_INT\\_PERIODIC\\_HOUR\\_DISABLE](#) : [RTC.h](#)
- [RTC\\_INT\\_PERIODIC\\_HOUR\\_ENABLE](#) : [RTC.h](#)
- [RTC\\_INT\\_PERIODIC\\_MIN\\_DISABLE](#) : [RTC.h](#)
- [RTC\\_INT\\_PERIODIC\\_MIN\\_ENABLE](#) : [RTC.h](#)
- [RTC\\_INT\\_PERIODIC\\_MONTH\\_DISABLE](#) : [RTC.h](#)
- [RTC\\_INT\\_PERIODIC\\_MONTH\\_ENABLE](#) : [RTC.h](#)
- [RTC\\_INT\\_PERIODIC\\_SEC\\_DISABLE](#) : [RTC.h](#)
- [RTC\\_INT\\_PERIODIC\\_SEC\\_ENABLE](#) : [RTC.h](#)
- [RTC\\_INT\\_PERIODIC\\_YEAR\\_DISABLE](#) : [RTC.h](#)
- [RTC\\_INT\\_PERIODIC\\_YEAR\\_ENABLE](#) : [RTC.h](#)
- [RTC\\_START\\_DISABLE](#) : [RTC.h](#)
- [RTC\\_START\\_ENABLE](#) : [RTC.h](#)
- [RTC\\_STATUS\\_FAILURE](#) : [RTC.h](#)
- [RTC\\_STATUS\\_SUCCESS](#) : [RTC.h](#)



# RTC

|                           |                         |  |
|---------------------------|-------------------------|--|
| <a href="#">Home</a>      |                         |  |
| <a href="#">File List</a> | <a href="#">Globals</a> |  |

## RTC.h

[Go to the documentation of this file.](#)

```
00001
00053 #ifndef RTC_H
00054 #define RTC_H
00055
00056
00057 /*****
*****
*****
*****
00058  * HEADER FILES
00059  *****/
00060 #include <xmc_scu.h>
00061 #include <xmc_rtc.h>
00062 #include "rtc_conf.h"
00063 #include <DAVE_Common.h>
00064
00065 /*****
*****
*****
00066  * MACROS
00067  *****/
00068 #if (!(XMC_LIB_MAJOR_VERSION == 2U) && \
00069       (XMC_LIB_MINOR_VERSION >= 0U) && \
00070       (XMC_LIB_PATCH_VERSION >= 0U))
```

```

00071 #error "RTC APP requires XMC Peripheral Libr
ary v2.0.0 or higher"
00072 #endif
00073
00074 /*****
*****
*****
00075  * ENUMS
00076  *****/
00082 typedef enum RTC_STATUS
00083 {
00084     RTC_STATUS_SUCCESS = 0U,
00085     RTC_STATUS_FAILURE = 1U
00086 } RTC_STATUS_t;
00087
00092 typedef enum RTC_START
00093 {
00094     RTC_START_DISABLE = 0x0U,
00095     RTC_START_ENABLE = 0x1U
00096 } RTC_START_t;
00097
00098
00104 typedef enum RTC_INT
00105 {
00106     RTC_INT_PERIODIC_SEC_DISABLE = 0x0U,
00107     RTC_INT_PERIODIC_SEC_ENABLE = 0x1U,
00108     RTC_INT_PERIODIC_MIN_DISABLE = 0x0U,
00109     RTC_INT_PERIODIC_MIN_ENABLE = 0x1U,
00110     RTC_INT_PERIODIC_HOUR_DISABLE = 0x0U,
00111     RTC_INT_PERIODIC_HOUR_ENABLE = 0x1U,

```

```

00112   RTC_INT_PERIODIC_DAY_DISABLE = 0x0U,
00113   RTC_INT_PERIODIC_DAY_ENABLE = 0x1U,
00114   RTC_INT_PERIODIC_MONTH_DISABLE = 0x0U,
00115   RTC_INT_PERIODIC_MONTH_ENABLE = 0x1U,
00116   RTC_INT_PERIODIC_YEAR_DISABLE = 0x0U,
00117   RTC_INT_PERIODIC_YEAR_ENABLE = 0x1U,
00118   RTC_INT_ALARM_DISABLE = 0x0U,
00119   RTC_INT_ALARM_ENABLE = 0x1U

00120 } RTC_INT_t;
00121
00122 /*****
*****
*****
00123 * DATA STRUCTURES
00124 *****/
00134 typedef struct RTC_CONFIG
00135 {
00136   RTC_START_t start;

00138   RTC_INT_t periodic_sec_intr;

00139   RTC_INT_t periodic_min_intr;

00140   RTC_INT_t periodic_hour_intr;

00141   RTC_INT_t periodic_day_intr;

```

```

00142  RTC_INT_t  periodic_month_intr;

00143  RTC_INT_t  periodic_year_intr;

00145  RTC_INT_t  alarm_intr;

00147  #if ((UC_FAMILY == XMC4) && (RTC_INTERRUPT_E
NABLED == 1))
00148  GLOBAL_SCU_XMC4_EVENT_HANDLER_t pi_listener
;
00149  GLOBAL_SCU_XMC4_EVENT_HANDLER_t ai_listener
;
00150  #endif
00151
00152  #if ((UC_FAMILY == XMC1) && (RTC_INTERRUPT_E
NABLED == 1))
00153  GLOBAL_SCU_XMC1_EVENT_HANDLER_t pi_listener
;
00154  GLOBAL_SCU_XMC1_EVENT_HANDLER_t ai_listener
;
00155  #endif
00156  } RTC_CONFIG_t;
00157
00161  typedef struct RealTimeClock
00162  {
00163      const XMC_RTC_CONFIG_t *const time_alarm_c
onfig;
00164      const RTC_CONFIG_t *const config;

00165      bool initialized;

00166  } RTC_t;
00171  #ifdef __cplusplus
00172  extern "C" {
00173  #endif
00174  /*****

```

```

*****
*****
00175  * API PROTOTYPES
00176  *****
*****
***** /
00177
00206 DAVE_APP_VERSION_t RTC_GetAppVersion(void);
00207
00208
00231 RTC_STATUS_t RTC_Init(RTC_t *const handle);
00232
00258 RTC_STATUS_t RTC_Time(time_t* time_value) ;
00259
00288 __STATIC_INLINE void RTC_Start(void)
00289 {
00290     XMC_RTC_Start();
00291 }
00292
00324 __STATIC_INLINE void RTC_Stop(void)
00325 {
00326     XMC_RTC_Stop();
00327 }
00328
00363 RTC_STATUS_t RTC_SetTime(XMC_RTC_TIME_t *time);
00398 RTC_STATUS_t RTC_SetAlarmTime(XMC_RTC_ALARM_t *alarm);
00434 void RTC_GetTime(XMC_RTC_TIME_t *time);
00469 void RTC_GetAlarmTime(XMC_RTC_ALARM_t *alarm);
00490 uint32_t RTC_GetFlagStatus(void);
00491
00496 #include "rtc_extern.h"
00497
00498 #ifdef __cplusplus
00499 }

```

```
00500 #endif
```

```
00501
```

```
00502 #endif
```

```
00503
```

```
00504
```



# RTC

|           |         |  |
|-----------|---------|--|
| Home      |         |  |
| File List | Globals |  |

## RTC.c

[Go to the documentation of this file.](#)

```
00001
00066 /* ****
****
****
00067  * HEADER FILES
00068  ****
****
**** */
00069 #include "rtc.h"
00070
00071 /* ****
****
****
00072  * MACROS
00073  ****
****
**** */
00074 /* Constant used for the number of days in a
n year */
00075 #define RTC_DAYS_IN_AN_YEAR      (365U)
00076 /* Constant used for the seconds in a day */
00077 #define RTC_SECONDS_IN_A_DAY     (24U * 60U *
60U)
00078 /* Constant used for the seconds in an hour
*/
00079 #define RTC_SECONDS_IN_AN_HOUR   (60U * 60U)
00080 /* Constant used for the seconds in a minute
```

```

*/
00081 #define RTC_SECONDS_IN_A_MINUTE (60U)
00082 /* Constant used for the epoch year */
00083 #define RTC_EPOCH_YEAR          (1970U)
00084
00085 /*****
*****
*****
00086  * LOCAL DATA
00087  *****/
00088 /* Constant array used to store the number o
f days in each month */
00089 const uint32_t RTC_DAYS_IN_MONTH[13] =
00090 {
00091     /* Index from 1, hence skip 0*/
00092     0U,
00093     /*Jan, Feb, Mar, Apr, May, Jun, Jul, Aug,
Sep, Oct, Nov, Dec*/
00094     31U, 28U, 31U, 30U, 31U, 30U, 31U, 31U, 30
U, 31U, 30U, 31U
00095 };
00096
00097 /*****
*****
*****
00098  * LOCAL ROUTINES
00099  *****/
00100 uint8_t RTC_lleapyear(uint16_t year);
00101 bool RTC_lConfigureInterrupts(const RTC_t *c
onst handler);
00102 XMC_RTC_STATUS_t RTC_lRegister_Callbacks(con
st RTC_t *const handler);
00103

```

```

00104 /*****
*****
*****
00105  * API IMPLEMENTATION
00106 *****/
00107 /*
00108  * API to retrieve the version of the RTC APP

00109 */
00110 DAVE_APP_VERSION_t RTC_GetAppVersion(void)
00111 {
00112     DAVE_APP_VERSION_t version;
00113
00114     version.major = RTC_MAJOR_VERSION;
00115     version.minor = RTC_MINOR_VERSION;
00116     version.patch = RTC_PATCH_VERSION;
00117
00118     return (version);
00119 }
00120
00121 /*
00122  Initialization function for the APP. Configures the registers
00123  based on options selected in UI.
00124 */
00125 RTC_STATUS_t RTC_Init(RTC_t *const handler)
00126 {
00127     XMC_RTC_STATUS_t status;
00128     RTC_STATUS_t rtc_initstatus;
00129     bool interrupt_configured;
00130
00131     XMC_ASSERT("RTC_Init: NULL Handler", handler != NULL);
00132
00133     status = XMC_RTC_STATUS_OK;

```

```

00134     rtc_initstatus = RTC_STATUS_FAILURE;
00135
00136 #if (RTC_INTERRUPT_ENABLED == 1)
00137 #if (UC_FAMILY == XMC4)
00138     rtc_initstatus = (RTC_STATUS_t)GLOBAL_SCU_
XMC4_Init(GLOBAL_SCU_HANDLE);
00139 #else
00140     rtc_initstatus = (RTC_STATUS_t)GLOBAL_SCU_
XMC1_Init(GLOBAL_SCU_HANDLE);
00141 #endif
00142     if (rtc_initstatus == RTC_STATUS_SUCCESS)
00143     {
00144 #endif
00145         if (handler->initialized == false)
00146         {
00147             /* Initialize the clock source and p
re-scalar */
00148             status = XMC_RTC_Init(handler->time_
alarm_config);
00149
00150             if (status == XMC_RTC_STATUS_OK)
00151             {
00152                 /* Configure periodic, alarm and h
ibernate periodic interrupts */
00153                 interrupt_configured = RTC_lConfig
ureInterrupts(handler);
00154
00155                 if (interrupt_configured == true)
00156                 {
00157                     status = RTC_lRegister_Callbacks
(handler);
00158                 }
00159
00160                 if (status == XMC_RTC_STATUS_OK)
00161                 {
00162                     /* Check RTC start during in
it is set or not in UI */

```

```

00163         if (handler->config->start == RT
C_START_ENABLE)
00164             {
00165                 RTC_Start();
00166             }
00167             handler->initialized = true;
00168             rtc_initstatus = RTC_STATUS_SUCC
ESS;
00169         }
00170     }
00171     else
00172     {
00173         rtc_initstatus = RTC_STATUS_FAILURE
;
00174     }
00175 }
00176 else
00177 {
00178     rtc_initstatus = RTC_STATUS_SUCCESS;
00179 }
00180 #if (RTC_INTERRUPT_ENABLED == 1)
00181     } /* end of if(rtc_initstatus == GLOBAL_S
CU_XMC4_STATUS_OK) */
00182 #endif
00183
00184     return (rtc_initstatus);
00185 }
00186 /*
00187  * This function configures periodic and al
arm interrupts
00188  */
00189 bool RTC_lConfigureInterrupts(const RTC_t *c
onst handler)
00190 {
00191     uint32_t regval;
00192     bool interrupt_configured = false;
00193

```

```

00194  /* Enable periodic seconds, minutes, hours
      days, months and years interrupts */
00195  regval = (((uint32_t)handler->config->peri
odic_sec_intr << RTC_MSKSR_MPSE_Pos)
00196          | ((uint32_t)handler->config->per
iodic_min_intr << RTC_MSKSR_MPMI_Pos)
00197          | ((uint32_t)handler->config->per
iodic_hour_intr << RTC_MSKSR_MPH0_Pos)
00198          | ((uint32_t)handler->config->per
iodic_day_intr << RTC_MSKSR_MPDA_Pos)
00199          | ((uint32_t)handler->config->per
iodic_month_intr << RTC_MSKSR_MPMO_Pos)
00200          | ((uint32_t)handler->config->per
iodic_year_intr << RTC_MSKSR_MPYE_Pos));
00201
00202  /* Enable RTC periodic interrupt in SCU wh
en any of the periodic interrupts
00203  * are enabled */
00204  if (regval != 0U)
00205  {
00206      XMC_RTC_EnableEvent(regval);
00207  #if ((UC_FAMILY == XMC4) && (RTC_TIMER_EVENT
_TRIG_TO_NMI == 1))
00208      XMC_SCU_INTERRUPT_EnableEven
t((XMC_SCU_INTERRUPT_EVENT_t)XMC_SCU_NMIREQ_RTC_PI
);
00209      XMC_SCU_INTERRUPT_EnableNmiR
equest((uint32_t)XMC_SCU_NMIREQ_RTC_PI);
00210  #endif
00211  #if ((UC_FAMILY == XMC4) && (RTC_TIMER_EVENT
_TRIG_TO_SCU == 1))
00212      GLOBAL_SCU_XMC4_EnableEvent(
(GLOBAL_SCU_XMC4_EVENT_t)GLOBAL_SCU_XMC4_EVENT_RTC
_PERIODIC);
00213  #endif
00214      interrupt_configured = true;
00215  }

```

```

00216
00217
00218     if (handler->config->alarm_intr == R
TC_INT_ALARM_ENABLE)
00219     {
00220         XMC_RTC_EnableEvent((uint32_
t)XMC_RTC_EVENT_ALARM);
00221 #if ((UC_FAMILY == XMC4) && (RTC_ALARM_EVENT
_TRIG_TO_NMI == 1))
00222         XMC_SCU_INTERRUPT_EnableEven
t((XMC_SCU_INTERRUPT_EVENT_t)XMC_SCU_NMIREQ_RTC_AI
);
00223         XMC_SCU_INTERRUPT_EnableNmiR
equest((uint32_t)XMC_SCU_NMIREQ_RTC_AI);
00224 #endif
00225 #if ((UC_FAMILY == XMC4) && (RTC_ALARM_EVENT
_TRIG_TO_SCU == 1))
00226         GLOBAL_SCU_XMC4_EnableEvent(
(GLOBAL_SCU_XMC4_EVENT_t)GLOBAL_SCU_XMC4_EVENT_RTC
_ALARM);
00227 #endif
00228
00229         interrupt_configured = true;
00230     }
00231
00232
00233     return (interrupt_configured);
00234 }
00235
00236 /*
00237  * Interface to register the RTC call backs
00238  */
00239 XMC_RTC_STATUS_t RTC_lRegister_Callbacks(con
st RTC_t *const handler)
00240 {
00241     XMC_RTC_STATUS_t pi_status;
00242     XMC_RTC_STATUS_t ai_status;

```

```

00243
00244     pi_status = XMC_RTC_STATUS_OK;
00245     ai_status = XMC_RTC_STATUS_OK;
00246
00247     #if (RTC_INTERRUPT_ENABLED == 1)
00248         #if (UC_FAMILY == XMC4)
00249             pi_status = (XMC_RTC_STATUS_t)GLOBAL_SCU
00250             _XMC4_RegisterCallback(
00251                 (GLOBAL_SCU_XMC4_EVENT_t)GLO
00252                 BAL_SCU_XMC4_EVENT_RTC_PERIODIC, handler->config->
00253                 pi_listener);
00254             #if (handler->config->alarm_intr == RTC_I
00255             NT_ALARM_ENABLE)
00256                 {
00257                     ai_status = (XMC_RTC_STATUS_t)GLOBAL_S
00258                     CU_XMC4_RegisterCallback(
00259                         (GLOBAL_SCU_XMC4_EVENT_t)GLO
00260                         BAL_SCU_XMC4_EVENT_RTC_ALARM, handler->config->ai_l
00261                         istener);
00262                 }
00263             #else
00264                 pi_status = (XMC_RTC_STATUS_t)GLOBAL_SCU
00265                 _XMC1_RegisterCallback((GLOBAL_SCU_XMC1_EVENT_t)GL
00266                 OBAL_SCU_XMC1_EVENT_RTC_PERIODIC,
00267
00268                                     handler->config->pi_listene
00269                 r);
00270             #if (handler->config->alarm_intr == RTC_I
00271             NT_ALARM_ENABLE)
00272                 {
00273                     ai_status = (XMC_RTC_STATUS_t)GLOBAL_S
00274                     CU_XMC1_RegisterCallback((GLOBAL_SCU_XMC1_EVENT_t)
00275                     GLOBAL_SCU_XMC1_EVENT_RTC_ALARM,
00276
00277                                     handler->config->ai_liste
00278                                     ner);

```

```

00264     }
00265     #endif
00266 #endif
00267
00268     return (XMC_RTC_STATUS_t)((uint32_t)pi_status & (uint32_t)ai_status);
00269 }
00270
00271 /*
00272  * This function is used to set RTC time.
00273  */
00274 RTC_STATUS_t RTC_SetTime(XMC_RTC_TIME_t *current_time)
00275 {
00276     RTC_STATUS_t status = RTC_STATUS_SUCCESS;
00277     XMC_RTC_TIME_t time_val;
00278
00279     XMC_ASSERT("RTC_SetTime: NULL pointer", current_time != NULL);
00280
00281     /* copy to local structure to keep data safe */
00282     time_val.year = current_time->year;
00283     time_val.month = current_time->month;
00284     time_val.days = current_time->days;
00285     time_val.hours = current_time->hours;
00286     time_val.minutes = current_time->minutes;
00287     time_val.seconds = current_time->seconds;
00288
00289     if ((time_val.days != 0U) && (time_val.month != 0U))
00290     {
00291         time_val.days = time_val.days - 1U;
00292         time_val.month = time_val.month - 1U;
00293
00294         XMC_RTC_SetTime(&time_val);
00295     }

```

```

00296     else
00297     {
00298         status = RTC_STATUS_FAILURE;
00299     }
00300
00301     return (status);
00302 }
00303
00304 /*
00305  * This function is used to get RTC time.
00306  */
00307 void RTC_GetTime(XMC_RTC_TIME_t *current_time)
00308 {
00309     XMC_ASSERT("RTC_GetTime: NULL pointer", current_time != NULL);
00310
00311     XMC_RTC_GetTime(current_time);
00312
00313     current_time->days = current_time->days +
00314     1U;
00315     current_time->month = current_time->month
00316     + 1U;
00317 }
00318 /*
00319  * This function is used to set Alarm time.
00320  */
00321 RTC_STATUS_t RTC_SetAlarmTime(XMC_RTC_ALARM_t *alarm)
00322 {
00323     RTC_STATUS_t status = RTC_STATUS_SUCCESS;
00324     XMC_RTC_ALARM_t alarm_val;
00325
00326     XMC_ASSERT("RTC_SetAlarmTime: NULL pointer", alarm != NULL);
00327

```

```

00327  /* copy to local structure to keep data safe */
00328  alarm_val.year = alarm->year;
00329  alarm_val.month = alarm->month;
00330  alarm_val.days = alarm->days;
00331  alarm_val.hours = alarm->hours;
00332  alarm_val.minutes = alarm->minutes;
00333  alarm_val.seconds = alarm->seconds;
00334
00335  if ((alarm_val.days != 0U) && (alarm_val.month != 0U))
00336  {
00337      alarm_val.days = alarm_val.days - 1U;
00338      alarm_val.month = alarm_val.month - 1U;
00339
00340      XMC_RTC_SetAlarm(&alarm_val);
00341  }
00342  else
00343  {
00344      status = RTC_STATUS_FAILURE;
00345  }
00346
00347  return (status);
00348 }
00349
00350 /*
00351  * This function is used to get Alarm time from XMC.
00352  * And returns in standard time format.
00353  */
00354 void RTC_GetAlarmTime(XMC_RTC_ALARM_t *alarm
)
00355 {
00356  XMC_ASSERT("RTC_GetAlarmTime: NULL pointer", alarm != NULL);
00357
00358  XMC_RTC_GetAlarm(alarm);

```

```

00359
00360     alarm->days = alarm->days + 1U;
00361     alarm->month = alarm->month + 1U;
00362 }
00363
00364 /*
00365  *   This function is used to get event statu
00366  */
00367 uint32_t RTC_GetFlagStatus(void)
00368 {
00369     uint32_t event_status;
00370
00371     event_status = XMC_RTC_GetEventStatus();
00372
00373     return (event_status);
00374 }
00375
00376 /*
00377  *   This function is to get the time in seco
00378  *   nds calculated from Epoch time
00379  *   (01/01/1970).
00380  */
00380 RTC_STATUS_t RTC_Time(time_t* time_value)
00381 {
00382     uint32_t elapsedyear;
00383     uint32_t elapsedmonth;
00384     uint32_t elapseddays;
00385     uint32_t elapsedseconds;
00386
00387     RTC_STATUS_t status;
00388     XMC_RTC_TIME_t curr_time;
00389
00390     XMC_ASSERT("RTC_Time: NULL pointer", time_
00391 value != NULL);
00392     /*Check if RTC module is enabled and no NU

```

```

LL pointer*/
00393     if (true == XMC_RTC_IsRunning())
00394     {
00395         /* Read values from TIM0 and TIM1 registers */
00396         XMC_RTC_GetTime(&curr_time);
00397
00398         /*Count number of Days for Elapsed Years since Epoch*/
00399         elapseddays = ((uint32_t)curr_time.year
- RTC_EPOCH_YEAR) * RTC_DAYS_IN_AN_YEAR;
00400
00401         /* Add the number of days to be adjusted for leap years,
start from previous year and check backward */
00402         for (elapsedyear = ((uint32_t)curr_time.
year - 1U); elapsedyear>= (uint32_t)1970; elapsedyear--)
00403         {
00404             if (RTC_lleapyear((uint16_t)elapsedyear))
00405             {
00406                 elapseddays++;
00407             }
00408         }
00409         /*If current year is leap year add 1 only if March or later*/
00410         if (RTC_lleapyear(curr_time.year))
00411         {
00412             if(curr_time.month > 2U)
00413             {
00414                 elapseddays++;
00415             }
00416         }
00417
00418
00419         /*Add number of Days from Elapsed months

```

```

    from current year*/
00420     for (elapsedmonth = (curr_time.month); e
lapsedmonth != 0U; elapsedmonth--)
00421     {
00422         elapseddays += RTC_DAYS_IN_MONTH[elaps
edmonth];
00423     }
00424
00425     /*Add Elapsed days from current month*/
00426     elapseddays += curr_time.days;
00427
00428     /*Accumulate the total seconds for Elaps
edDays*/
00429     elapsedseconds = (elapseddays * RTC_SECO
NDS_IN_A_DAY);
00430
00431     /*Add seconds for current hour, minute a
nd seconds*/
00432     elapsedseconds += ((uint32_t)curr_time.h
ours * RTC_SECONDS_IN_AN_HOUR);
00433     elapsedseconds += ((uint32_t)curr_time.m
inutes * RTC_SECONDS_IN_A_MINUTE);
00434     elapsedseconds += (uint32_t)curr_time.se
conds;
00435
00436     *time_value = (time_t)elapsedseconds;
00437     status = RTC_STATUS_SUCCESS;
00438 }
00439 else
00440 {
00441     status = RTC_STATUS_FAILURE;
00442 }
00443 return (status);
00444 }
00445
00446 /* This function returns 1 if it is leap yea
r otherwise 0.*/

```

```
00447 uint8_t RTC_lleapyear(uint16_t year)
00448 {
00449     uint8_t valid = 0U;
00450
00451     if (((year) % 400U) == 0U) || (((year) %
00452         100U) != 0U) &&
00453         (((year) %4U) == 0U)))
00454     {
00455         valid = 1U;
00456     }
00457     return (valid);
00458 }
```

