AN3133
Application note

Using the STM8L15x real-time clock (RTC)

Introduction

A real-time clock (RTC) is a computer clock that keeps track of the current time. Although RTCs are often used in personal computers, servers and embedded systems, they are also present in almost any electronic device that requires accurate time keeping. Microcontrollers supporting RTC can be used for chronometer, alarm clock, watches, small electronic agendas, and many other devices.

This application note describes the features of the real-time clock (RTC) controller embedded in the STM8L15x microcontrollers, together with the steps required to configure the RTC calendar, the alarm, and the periodic wakeup unit.

Some useful configurations are described to allow the user to quickly and adequately configure the RTC for his application.

Three applicative examples are provided. They explain how to use the calendar, the alarm and the periodic wakeup unit.

Note: All examples and explanations are based on the STM8L15x firmware library and refer to the STM8L15x reference manual (RM0031).

Contents

List of tables

List of figures

1 Overview of the STM8L15x RTC

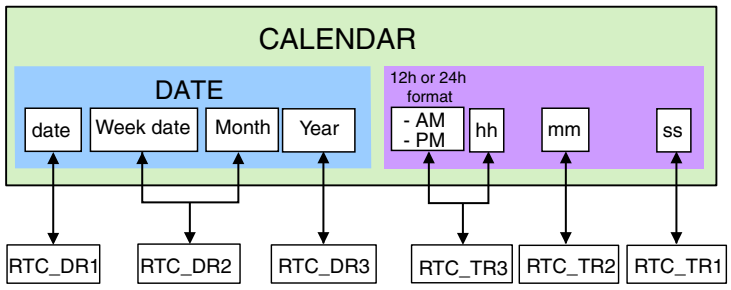
The RTC embedded in the STM8L15x provides a full-featured calendar, an alarm, and a periodic wakeup unit.

1.1 STM8L15x RTC calendar

A calendar keeps track of the time (hours, minutes, seconds) and date (day, week, month, year). The STM8L15x RTC calendar offers many features to easily configure and display the calendar data fields:

- Calendar with seconds, minutes, hours in 12-hour or 24-hour format, day of the week (day), day of the month (date), month, and year.
- Calendar in BCD format
- Automatic management of 28-, 29- (leap year), 30-, and 31-day months
- Daylight saving time adjustment programmable by software

Figure 1. STM8L15x RTC calendar fields

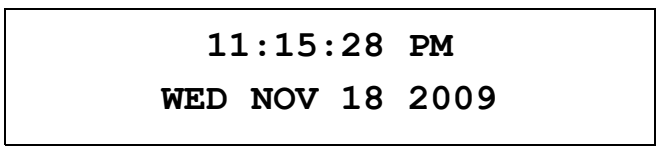


1. RCT_DRx and RTC_TRx are RTC registers.

In some microcontrollers, the calendar is a kind of software counter (usually 32-bit long) which represents the number of seconds. Software routines convert the counter value to hours, minutes, day of the month, day of the week, month and year. These data can be converted to BCD format (binary coded decimal) and displayed on a standard LCD which is particularly useful in countries where the hours are displayed in 12-hour format plus an AM/PM indicator (see *Figure 2*). Conversion routines uses significant program memory space and are CPU-time consuming which may be critical in some real-time applications.

When using the STM8L15x RTC calendar, software conversion routines are no more needed because their functionalities are performed by hardware.

Figure 2. Example of calendar display on an LCD



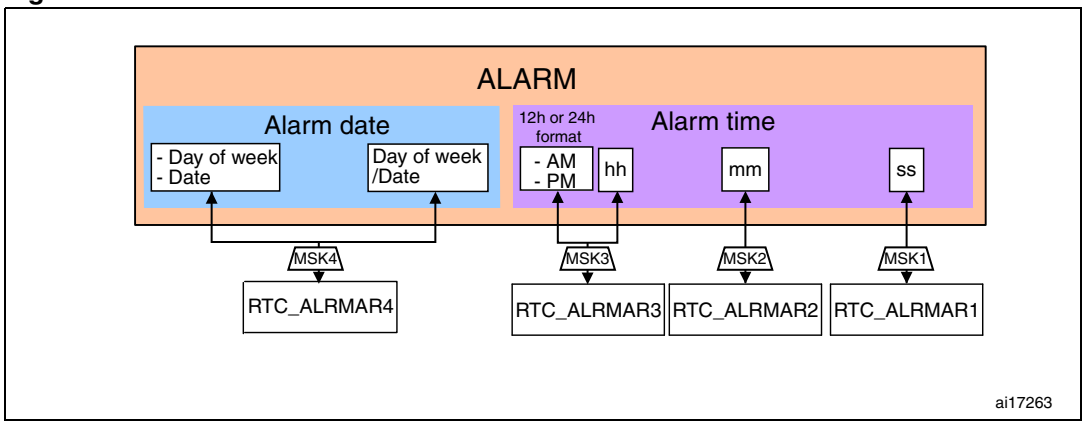
1.2 STM8L15x RTC alarm

An alarm can be generated at a given time or/and date programmed by the user.

The STM8L15x RTC provides a rich combination of alarms, and offers many features to easily configure, and display these alarms:

- Full programmable alarm: seconds, minutes, hours and date fields can be independently selected or masked to provide the user a rich combinations of alarms.
- Ability to exit the device from Active-halt mode when the alarm occurs.
- The alarm event can be routed to a specific output pad with configurable polarity.
- Dedicated alarm flag and Interrupt.

Figure 3. STM8L15x Alarm fields



1. RTC_ALRMARx are RTC registers.
2. MSKx are bits in the RTC_ALARMx registers. They allow enabling and disabling the RTC_ALARMx fields used for alarm and calendar comparison. For more details refer to *Table 6*.

In some microcontrollers, the alarm consist of a register with the same length as the RTC time counter. When the RTC time counter reaches the value programmed in the alarm register, a flag is set to indicate that an alarm event occurred.

The STM8L15x RTC alarm can be configured by hardware to generate different types of alarms. For more details refer to *Table 6*.

1.3 STM8L15x RTC periodic wakeup unit

Like many low consumption microcontrollers, the STM8L15x provides several low power modes to reduce the power consumption.

The STM8L15x features a periodic timebase and wakeup unit that can wake up the system when the STM8L15x operates in low power mode. This unit is a programmable downcounting auto-reload timer. When this counter reaches zero, a flag and an interrupt (if enabled) are generated.

The wakeup unit features are the following:

- Programmable downcounting auto-reload timer
- Specific flag and interrupt capable to wake up the device from low power modes
- Wakeup alternate function output which can be routed to RTC_ALARM output (unique pad for both Alarm and Wakeup events) with configurable polarity.
- A full set of prescalers to select the desired waiting period.

1.4 RTC and low-power consumption

The STM8L15xx RTC is designed to minimize the power consumption. The prescalers used for the calendar are divided in 2: synchronous and asynchronous.

Increasing the value of the asynchronous prescaler allows to reduce the power consumption.

The RTC continues working in reset mode and its registers are not reset except by Power-on reset. RTC registers values are not lost after a reset and the calendar keeps the correct time and date.

After a system reset or a power-on reset the STM8L15x operates in Run mode. In addition, the device supports five low power modes to achieve the best compromise between low power consumption, short startup time and available wakeup sources.

The RTC peripheral can be active in the following low power modes:

- Wait
- Low power run
- Low power wait
- Active-halt

The RTC cannot wake up the device from Low power run and Low power wait mode since there is no associated event.

The RTC remains active in Low power run, Low power wait and Active-halt mode only if the clock source is LSI or LSE. If the RTC clock is HSI or HSE, and the HALT instruction is executed, the RTC is stopped (since the HSI and HSE clocks are stopped in Halt mode) and cannot wake up the device.

Refer to the low power modes section of the STM8L15x reference manual for more details about low power modes.

Table 1. Low power modes where RTC is actor

Mode	Entry	Oscillator	CPU	Peripherals status	wake up
Wait mode	WFI/WFE(1)	ON	ON	ON	Internal or external event, reset
Low power run mode	Software sequence	LSI or LSE clock	ON	ON except RTC and 1 other peripheral(2)	Software sequence, reset
Low power wait mode	Software sequence + WFE	LSI or LSE clock	OFF	ON except RTC and 1 other peripheral(2)	Internal or external event, reset
Active-halt mode	HALT(3)	Off except LSI or LSE clock	OFF	OFF except RTC and possibly LCD	External interrupts, RTC interrupt, reset

1. There is no event associated to the RTC. As a consequence, the interrupt is served both in WFE and WFI modes.
2. Only the peripherals which running with the LSI or LSE clock can be enabled.
3. Before executing the HALT instruction, the application must clear all pending peripheral interrupts by clearing the corresponding interrupt bit in the peripheral configuration register. Otherwise, the HALT instruction is not executed and program execution continues.

1.5 Signals generated by RTC

The RTC peripheral has 2 outputs:

- RTC_CALIB: it can be used to generate an external clock.
- RTC_ALARM: unique output resulting from the multiplexing of the RTC alarm and wakeup events.

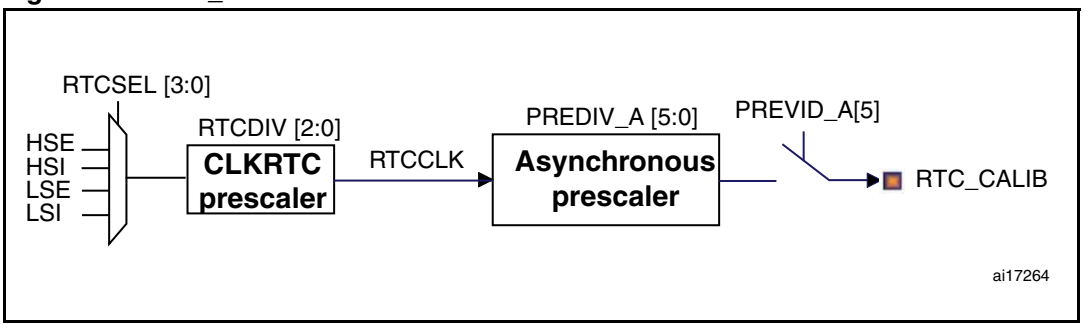
1.5.1 RTC_CALIB output

The RTC_CALIB output is used to generate a variable-frequency square signal. Depending on the user application, this signal can play the role of a reference clock to calibrate an external device, or be connected to a buzzer to generate a sound.

The signal frequency is configured through the 6 LSB bits (PREDIV_A [5:0]) of the Asynchronous prescaler register, RTC_APRER. PREDIV_A[5] must be set to 1 to enable the RTC_CALIB output signal generation. If PREDIV_A[5] is reset, no signal is output on RTC_CALIB.

Note: The RTC_CALIB output is available on PD3 for 28-pin devices and on PD6 for 32- and 48-pin devices.

Figure 4. RTC_CALIB clock sources



1. RTCDIV[2:0] and RTCSEL[3:0] are bits of the CLK_CRTCR register.

1.5.2 RTC_ALARM output

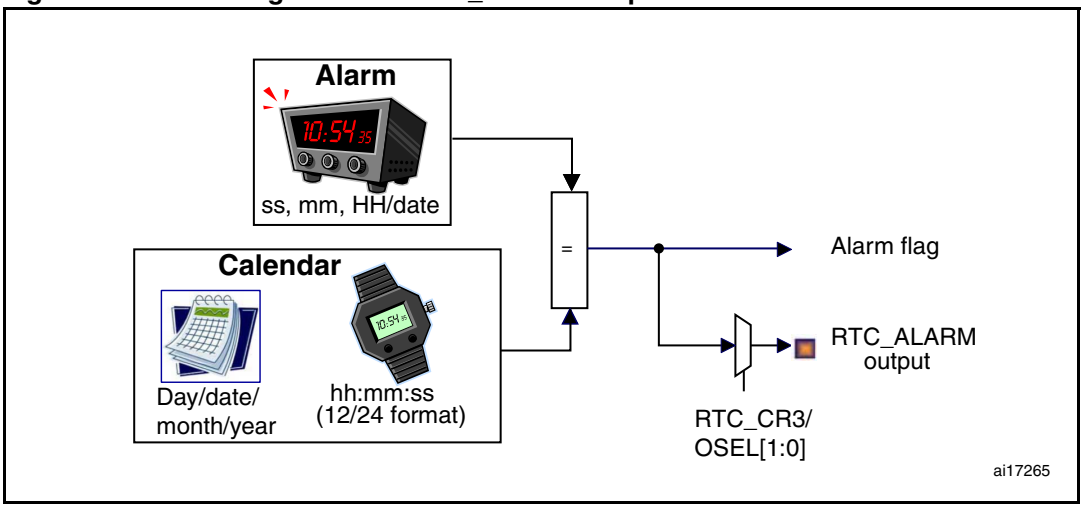
The RTC_ALARM output can either be connected to the RTC alarm unit to trigger an external action, or routed to the RTC wakeup unit to wake up an external device.

Note: The RTC_ALARM pin is on PB3 for 28-pin devices, on PD7 for 32- and 48-pin devices.

RTC_ALARM output connected to the RTC alarm unit

When the calendar reaches the value pre-programmed in the RTC_ALRMARx registers, the alarm flag (ALRAF bit in RTC_ISR2 register) is set to 1. If the alarm flag is routed to the RTC_ALARM output (OSEL[1:0] bits set to 01 in RTC_CR3), this pin is set to V_{DD} or to GND, depending on the polarity selected. The output toggles when the alarm flag is cleared.

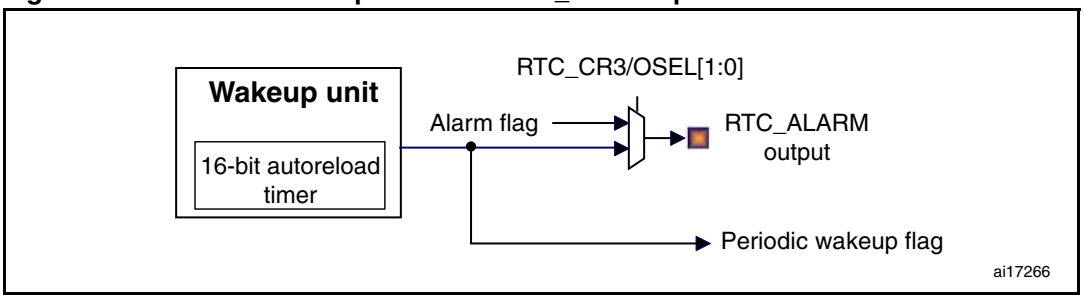
Figure 5. Alarm flag routed to RTC_ALARM output



RTC_ALARM output connected to the wakeup unit

When the wakeup downcounting timer reaches 0, the wakeup flag is set to 1. If this flag is selected as source for the RTC_ALARM output (OSEL[1:0] bits set to 11 in RTC_CR3 register), the output will be set depending to the polarity selected and will remain set as long as the flag is not cleared.

Figure 6. Periodic wake up routed to RTC_ALARM pinout



1.6 RTC security aspects

1.6.1 RTC Register write protection

To protect RTC registers against possible parasitic write accesses after reset, the RTC registers are automatically locked. they must be unlocked to update the current calendar time and date.

Writing to the RTC registers is enabled by programming a key into the Write protection register, RTC_WPR.

The following steps are required to unlock the RTC register write protection:

1. Write 0xCA into the RTC_WPR register.
2. Write 0x53 into the RTC_WPR register.

Writing a wrong key automatically reactivates the RTC register write access protection.

1.6.2 Enter/Exit initialization mode

The RTC can operate in two modes:

- Initialization mode where the counters are stopped.
- Free-running mode where the counters are running.

The calendar cannot be updated while the counters are running. The RTC must consequently be switched to initialization mode before updating the time and date.

When operating in this mode, the counters are stopped. They start counting from the new value when the RTC enters free-running mode.

The INIT bit of the RTC_ISR1 register allows to switch from one mode to another, while the INITF bit can be used to check the RTC current mode.

The RTC must be in initialization mode to program the time and date registers (RTC_TRx and RTC_DRx) and the prescaler registers (RTC_SPRERx and RTC_APRER). This is done by setting the INIT bit and waiting until RTC_ISR1_INITF flag is set.

To return to the free-running mode and restart counting, the RTC must exit initialization mode. This is done by resetting the INIT bit.

Only a power-on reset can reset the calendar. A system reset does not affected it but resets the shadow registers which are read by the application. They will be updated again when the RSF bit is set. After a system reset, the application can check the INITS status flag in RTC_ISR1 to verify if the calendar is already initialized. This flag is reset when the calendar

year field is set to 0x00 (power-on reset value), meaning that the calendar must be initialized.

1.6.3 Synchronization

When the application reads the calendar, it actually accesses shadow registers which contain a copy of the real calendar time and date clocked by the RTCCLK clock. To make sure that the shadow registers are updated with the current calendar value, the application must check that the RSF bit is set in the RTC_ISR1 register. This bit is set by hardware each time the calendar time and date shadow registers are updated, that is when the RTCCLK clock is synchronized with the system clock SYSCLK. The application software must clear the RSF bit after reading the calendar registers.

When the system is woken up from Active-halt mode (SYSCLK was off), the application must first clear the RSF bit, and then wait until it is set again before reading the calendar registers. This ensures that the value read by the application is the current calendar value, and not the value before entering Active-halt mode.

2 Programming the STM8L15x RTC

2.1 Initializing the calendar

The steps described in *Table 2* are required to configure correctly the calendar time and date.

Table 2. Steps to initialize the calendar

Step	What to do	How to do it	Comments
1	Enter initialization phase mode	Set INIT bit to 1 in RTC_ISR1 register	The calendar counter is stopped to allow update.
2	Wait for the confirmation of the initialization mode (clock synchronization).	Poll INITF bit of in RTC_ISR1 until it is set	It takes around 2 RTCCLK clock cycles.
3	Program the 3 prescaler registers if needed.	RTC_APRER and RTC_SPRERx	
4	Load time and date values in the shadow registers	Set RTC_TRx and RTC_DRx registers	
5	Configure the time format (12h or 24h)	Set FMT bit in RTC_CR1 register	
6	Exit initialization mode.	Clear the INIT bit in RTC_ISR1 register	The current calendar counter is then automatically loaded and the counting restarts after 4 RTCCLK clock cycles.

2.2 Programming the alarm

The steps described in *Table 3* are required to configure the alarm.

Table 3. Steps to configure the alarm

Step	What to do	How to do it	Comments
1	Disable the alarm	Clear ALRAE in RTC_CR2.	
2	Check that the RTC_ALRMARx registers can be accessed.	Poll ALRAWF until it is set in RTC_ISR1.	It takes around 2 RTCCLK clock cycles (clock synchronization).
3	Configure the alarm.	Configure RTC_ALRMARx registers.	The alarm hour format be the same as the RTC Calendar in RTC_ALARM3(1).
4	Enable the alarm again	Set ALRAE in RTC_CR2.	

1. As an example, if the alarm is configured to occurs at 3:00:00 PM, the alarm will not occur even if the calendar time is 15:00:00, because the RTC calendar is 24-hour format and the alarm is 12-hour format.

2.3 Programming the Auto-wakeup unit

The steps described in *Table 4* are required to configure the Auto-wakeup unit.

Table 4. Steps to configure the Auto wake-up unit

Step	What to do	How to do it	Comments
1	Disable the wakeup timer.	Clear WUTE in RTC_CR2.	
2	Make sure the access to Wakeup auto-reload counter and to the WUCKSEL[2:0] bits is allowed.	Poll WUTWF until it is set in RTC_ISR1.	It takes around 2 RTCCLK clock cycles.
3	Program the value into the wakeup timer.	Set RTC_WUTRL and RTC_WUTRH.	see *Section 3.4: Maximum and minimum RTC wakeup period*
4	Select the desired clock source.	Program WUCKSEL[2:0] bits in RTC_CR1.	
5	Enable the wakeup timer again.	Set WUTE in RTC_CR2 register.	The wakeup timer restarts down-counting.

3 Useful RTC configuration examples

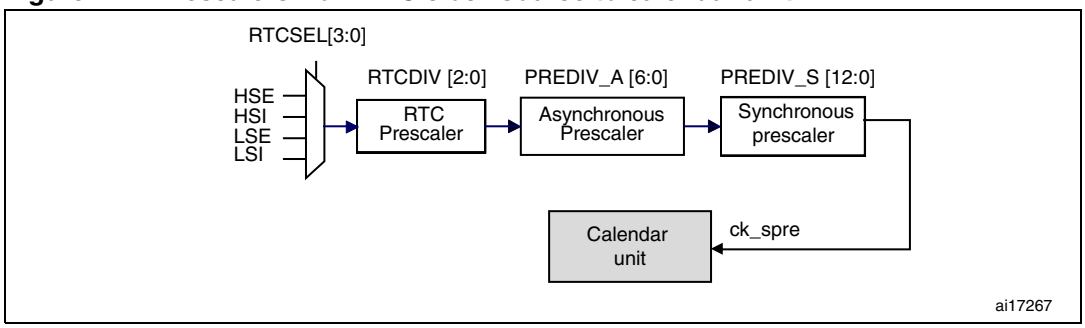
This section explains how to configure the STM8L15x RTC, and provides examples of configurations.

All the values provided in this section correspond to an HSE clock frequency of 1 MHz. However the HSE frequency can be up to 16 MHz.

3.1 Delivering a 1-Hz signal to the calendar using different clock sources

The RTC features several prescalers that allow delivering a 1-Hz clock to the calendar unit whatever the clock source.

Figure 7. Prescalers from RTC clock source to calendar unit



1. RTCDIV[2:0] and RTCSEL[3:0] are bits of the CLK_CRTCR register.

The formula to calculate ck_spre is:

$$ck\_spre = \frac{CLKSrc}{2^{RTCDIV[2:0]} \times (PREVID\_A + 1) \times (PREVID\_S + 1)}$$

where:

CLKSrc can be any clock source: HSE, HSI, LSE or LSI

RTCDIV[2:0] can be 0,1,2,..., or 6

PREDIV_A can be 1,2,3,..., or 127

PREDIV_S can be 0,1,2,..., 8191

Table 5 shows several possibilities to obtain ck_spre = 1 Hz.

Table 5. Calendar clock equal to 1Hz with different clock sources

Clock source	Prescalers			ck_spre
	RTCDIV[2:0]	PREDIV_A[6:0]	PREDIV_S[12:0]	
HSE = 1 MHz	6 (div64)	124 (div125)	124 (div125)	1 Hz
HSI = 16 MHz	6 (div64)	124 (div125)	1999 (div2000)	1 Hz
LSE = 32.768 kHz	0 (div1)	127 (div128)	255 (div256)	1 Hz
LSI = 38 kHz	0 (div1)	124 (div125)	303 (div304)	1 Hz

3.2 Configuring the alarm behavior using the MSKx bits

The alarm behavior can be configured through the MSKx bits (x=1, 2, 3, 4) bits of the RTC_ALRMARx registers.

Table 6 shows all the possible alarm settings. As an example, to configure the alarm time to 23:15:07 on Monday, MSKx bits must be set to 0000b.

Table 6. Alarm combination

MSK4	MSK3	MSK2	MSK1	Alarm behavior
0	0	0	0	**All fields are used in alarm comparison:** Alarm occurs at 23:15:07, each Monday.
0	0	0	1	**Seconds don't care in alarm comparison** The alarm occurs every second of 23:15, each Monday.
0	0	1	0	**Minutes don't care in alarm comparison** The alarm occurs at the 7th second of every minute of 23:XX, each Monday.
0	0	1	1	**Minutes and seconds don't care in alarm comparison**
0	1	0	0	**Hours don't care in alarm comparison**
0	1	0	1	**Hours and seconds don't care in alarm comparison**
0	1	1	0	**Hours and minutes don't care in alarm comparison**
0	1	1	1	**Hours, minutes and seconds don't care in alarm comparison:** The alarm is set every second, each Monday, during the whole day.
1	0	0	0	**Week day (or date, if selected) don't care in alarm comparison:** Alarm occurs all days at 23:15:07.
1	0	0	1	**Week day and seconds don't care in alarm comparison**
1	0	1	0	**Week day and minutes don't care in alarm comparison**

Table 6. Alarm combination (continued)

MSK4	MSK3	MSK2	MSK1	Alarm behavior
1	0	1	1	**Week day, minutes and seconds don't care in alarm comparison**
1	1	0	0	**Week day and Hours don't care in alarm comparison**
1	1	0	1	**Week day, Hours and seconds don't care in alarm comparison**
1	1	1	0	**Week day, Hours and minutes don't care in alarm comparison**
1	1	1	1	**Alarm occurs continuously what ever is the calendar time**

3.3 Maximum and minimum RTC_CALIB output frequency

The RTC can output the RTCCLK clock divided by a 6-bit asynchronous prescaler. The divider factor is configured through PREDIV_A[5:0] to the RTC_APRER register.

RTC_CALIB maximum and minimum frequencies are 484.85 kHz and 8 Hz, respectively.

Table 7. RTC_CALIB output frequency versus clock source

Clock source	RTC_CALIB frequency	
	Minimum (RTCDIV[2:0] = 111b and PREDIV_A[5:0] = 111 111b)	**Maximum (RTCDIV[2:0] = 000b and PREDIV_A[5:0] = 100 000b(1))**
HSE = 1 MHz	244.141 Hz	30,303.03 Hz
HSI = 16 MHz	3.906 kHz	**484,848.5 Hz**
LSE = 32 768 Hz	**8 Hz**	992.97 Hz
LSI = 38 kHz	9.277 Hz	1,151.515 Hz

1. PREDIV_A[5] must be set to 1 to enable the RTC_CALIB output signal generation. If PREDIV_A[5] bit is reset, no signal is output on RTC_CALIB.

3.4 Maximum and minimum RTC wakeup period

The wakeup unit clock is configured through the WUCKSEL[2:0] bits of RTC_CR1 register. Three different configurations are possible:

- Configuration 1
  WUCKSEL[2:0] = 0xxb for short wakeup periods (see *Section 3.4.1*)
- Configuration 2
  WUCKSEL[2:0] = 10xb for medium wakeup periods (see *Section 3.4.2*)
- Configuration 3
  WUCKSEL[2:0] = 11xb for long wakeup periods (see *Section 3.4.3*)

3.4.1 Periodic timebase/wakeup clock configuration 1

Figure 8 shows the prescaler connection to the timebase/wakeup unit and *Table 8* gives the timebase/wakeup clock resolutions corresponding to configuration 1.

Figure 8. Prescalers connected to the timebase/wakeup unit for configuration 1

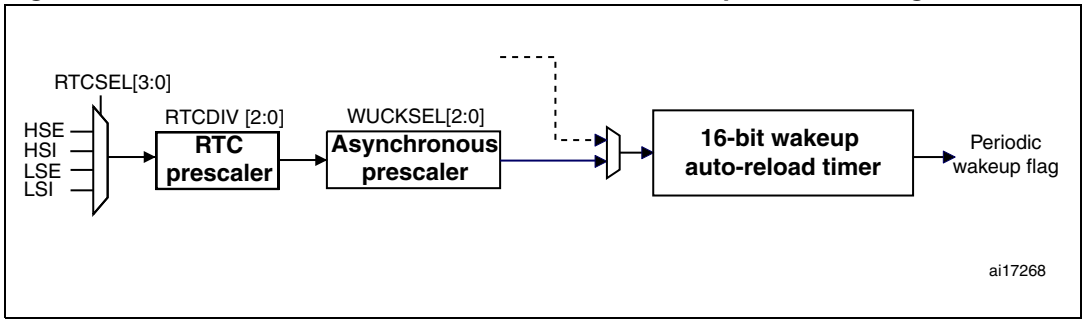


Table 8. Timebase/wakeup unit period resolution with clock configuration 1

Clock source	Wakeup period resolution			
	RTCDIV[2:0] = 111b (div64)		RTCDIV[2:0] = 000b (div1)	
	WUCKSEL[2:0] = 000b (div16)	WUCKSEL[2:0] = 011b (div2)	WUCKSEL[2:0] = 000b (div16)	WUCKSEL[2:0] = 011b (div2)
HSE = 1 MHz	1.024 ms	0.128 ms	16 µs	2 µs
HSI = 16 MHz	0.064 ms	0.008 ms	1 µs	**0.125 µs**
LSE = 32 768 Hz	**31.25 ms**	3.90625 ms	488.2812 µs	61.0351 µs
LSI = 38 kHz	26.9473 ms	3.368421 ms	421.0526 µs	52.6315 µs

The minimum timebase/wakeup resolution is 0.125 µs, and the maximum resolution 31.25 ms. As a result:

- The minimum timebase/wakeup period is (0x0001 + 1) x 0.125 µs = 0.250 µs.

  The timebase/wakeup timer counter WUT[15:0] cannot be set to 0x0000 with WUCKSEL[2:0]=011b ($f_{RTCCLK}/2$) because this configuration is forbidden. Refer to the STM8L15x reference manual for more details.
- The maximum timebase/wakeup period is (0xFFFF+ 1) x 31.25 ms = 2048 s.

3.4.2 Periodic timebase/wake up clock configuration 2

Figure 9 shows the prescaler connection to the timebase/wakeup unit and *Table 9* gives the timebase/wakeup clock resolutions corresponding to configuration 2.

Figure 9. Prescalers connected to the wake up unit for configurations 2 and 3

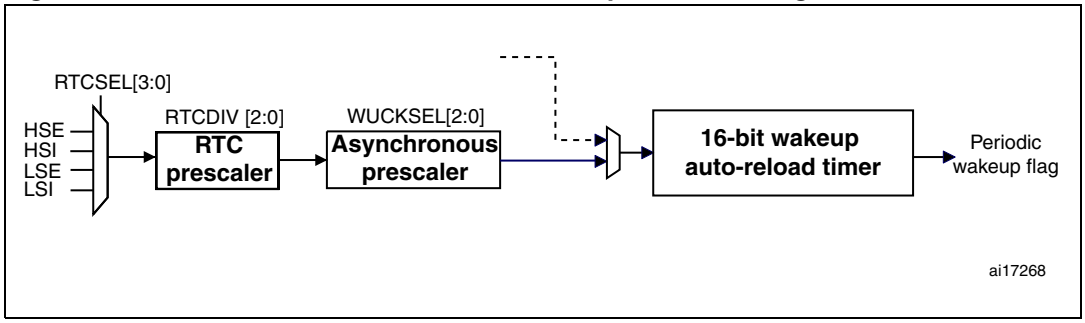


Table 9. Timebase/wakeup unit period resolution with clock configuration 2

Clock source	Wakeup period resolution			
	RTCDIV[2:0] = div64 RTCDIV_A[6:0] = div128 RTCDIV_S[12:0] = div8192		RTCDIV[2:0] = div1 RTCDIV_A[6:0] = div2(1) RTCDIV_S[12:0] = div1	
HSE = 1 MHz	1.024 ms	0.128 ms	16 µs	2 µs
HSI = 16 MHz	0.064 ms	0.008 ms	1 µs	**0.125 µs**
LSE = 32 768 Hz	**31.25 ms**	3.90625 ms	488.2812 µs	61.0351 µs
LSI = 38 kHz	26.9473 ms	3.368421 ms	421.0526 µs	52.6315 µs

1. PREDIV_A minimum value is 1.

The minimum resolution for configuration 2 is 0.125 µs, and the maximum resolution 2048 s.

As a result:

- The minimum timebase/wakeup period is (0x0000 + 1) x 0.125 µs = 0.125 µs.
- The maximum timebase/wakeup period is (0xFFFF+ 1) x 2048 s = 134217728 s (more than 4 years).

3.4.3 Periodic timebase/wakeup clock configuration 3

For this configuration, the resolution is the same as for configuration 2. However the timebase/wakeup counter downcounts starting from 0x1FFFF to 0x00000, instead of 0xFFFF to 0x0000 for configuration 2.

As a result:

- The minimum timebase/wakeup period is (0x10000 + 1) x 0.125 µs = 8.192125 ms.
- The maximum timebase/wakeup period is (0x1FFFF+ 1) x 2048 s = 268435456 s (more than 8 years and a half)

3.4.4 Summary of timebase/wakeup period extrema

The minimum and maximum period values, according on the configuration, are listed in the following table.

Table 10. Minimum and maximum timebase/wakeup period

Configuration	Minimum period	Maximum period
1	0.250 µs	2048 s
2	**0.125 µs**	More than 4 years
3	8.192125 ms	**More than 8 years and a half**

4 RTC firmware API

4.1 Function groups

The STM8L15x RTC driver can be divided into 8 function groups related to the functionalities embedded in the RTC peripheral.

- RTC initialization and configuration functions
- RTC time and date functions
- RTC alarm functions
- RTC wakeup functions
- RTC daylight saving functions
- RTC output pin configuration function
- RTC calibration output pin function
- RTC flags and interrupt functions

Table 11. RTC function groups

Group	Function name	Function Description
1	RTC_DeInit	Deinitialize the RTC registers to their default reset values.
	RTC_Init	Initialize the RTC registers according to the specified parameters in RTC_InitStruct <Hour format, Asynchronous predivisor, Asynchronous predivisor>.
	RTC_StructInit	Fill each RTC_InitStruct member with its default value.
	RTC_EnterInitMode	Enter the RTC Initialization mode.
	RTC_ExitInitMode	Exit the RTC Initialization mode.
	RTC_WriteProtectionCmd	Enable or disable the RTC registers write protection.
	RTC_WaitForSynchro	Wait until the RTC Time and Date registers (RTC_TRx and RTC_DRx) are synchronized.
	RTC_RatioCmd	Configure the RTC ratio.
2	RTC_SetTime	Set the RTC current time < RTC hours, RTC minutes, RTC seconds, RTC 12-hour clock period (AM/PM)>.
	RTC_SetDate	Set the RTC current date. < Calendar weekday, Calendar Month, Calendar date, Calendar year>.
	RTC_GetTime	Get the RTC current time.
	RTC_GetDate	Get the RTC current date.
3	RTC_SetAlarm	Set the RTC alarm configuration. < Alarm time fields, Alarm masks, Alarm date/Weekday selection, Alarm Date/Weekday value>.
	RTC_GetAlarm	Get the RTC alarm configuration.
	RTC_AlarmCmd	Enable or disable the RTC alarm.

Table 11. RTC function groups (continued)

Group	Function name	Function Description
4	RTC_WakeUpClockConfig	Configure the RTC wakeup clock source.
	RTC_SetWakeUpCounter	Set the RTC Wakeup counters.
	RTC_GetWakeUpCounter	Return the RTC Wakeup timer counter value.
	RTC_WakeUpCmd	Enable or Disable the RTC Wakeup timer.
5	RTC_DayLightSavingConfig	Add or subtract one hour from the current time depending on the daylight saving parameter.
	RTC_GetStoreOperation	Return the daylight saving stored operation.
6	RTC_OutputConfig	Configure the RTC output for the output pin.
7	RTC_CalibOutputCmd	Enable or disable connection of the RTCCLK/ PREDIV_A[6:0] clock to be output through the relative pin.
8	RTC_ITConfig	Enable or disable the specified RTC interrupts.
	RTC_GetFlagStatus	Check whether the specified RTC flag is set or not.
	RTC_ClearFlag	Clear the RTC pending flags.
	RTC_GetITStatus	Check whether the specified RTC interrupt has occurred or not.
	RTC_ClearITPendingBit	Clear the RTC interrupt pending bits.

4.2 Application examples

The STM8L15x RTC firmware is provided with a set of examples to quickly become familiar with the firmware library.

The present section provides three examples:

- The first one shows how to configure and display calendar and alarm settings.
- The second example shows how to configure wakeup unit in low power mode.
- The third example shows how to generate wakeup events at a frequency higher than 1 Hz.

4.2.1 Example 1: calendar and alarm

This example provides a short description of how to use the RTC peripheral calendar features: seconds, minutes, hours (12 or 24 format), day, date, month, and year.

This example is delivered within the STML8L15x firmware library available from http://wwwst.com/stm8l (Documents and files for STM8L family). It is located at STM8l15x_StdPeriph_Lib.zip\Project\STM8L15x_StdPeriph_Examples\RTC\RTC_Calenda\.

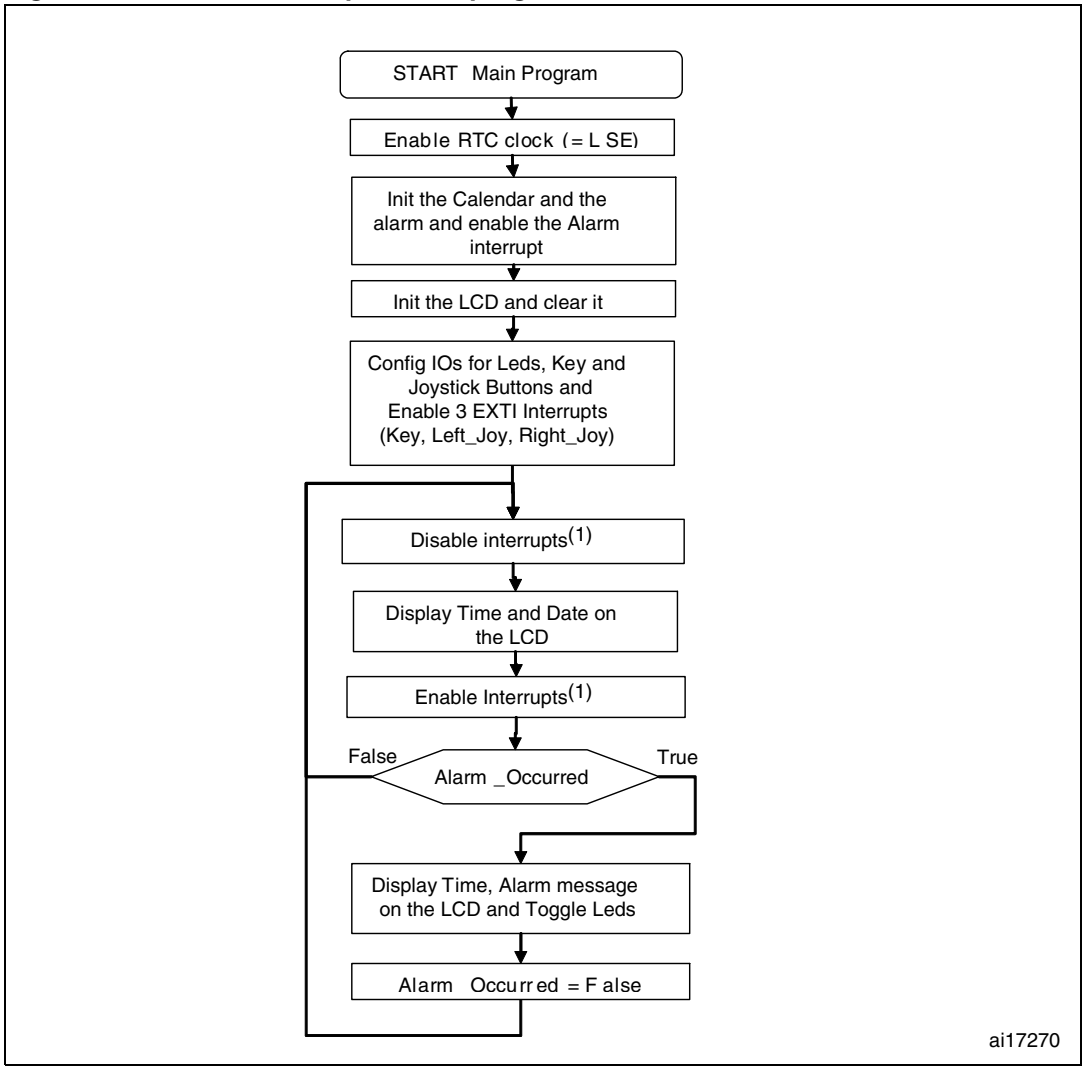
This example runs on the STM8L1526-EVAL. It allows configuring the RTC calendar and alarm and displaying their values in real-time using the LCD and joystick.

After power-up, the default date and time are displayed on the LCD. The user can then modify the date, time and alarm using the joystick buttons.

When an alarm occurs, a message is displayed on the LCD, and the LEDs toggle for a second.

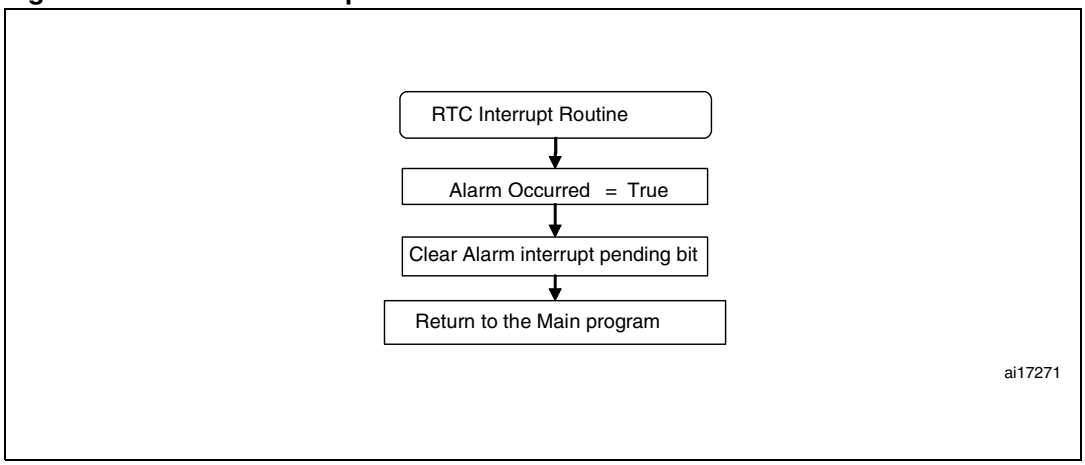
The flowcharts of this example are presented in *Figure 10* and *Figure 11*.

Figure 10. Calendar example: main program flowchart



1. These steps are added to have a secure display on the LCD (which uses SPI connections).

Figure 11. Calendar example: RTC alarm ISR flowchart



4.2.2 Example 2: wakeup from low power mode

This example explains how to use the STM8L15x LCD embedded controller to drive the LCD glass mounted on STM8L1526-EVAL board and how to exit the MCU from Active-halt mode using the wakeup unit.

This example is delivered within the STML8L15x firmware library available from http://wwwst.com/stm8l (Documents and files for STM8L family). It is located at STM8l15x_StdPeriph_Lib.zip\Project\STM8L15x_StdPeriph_Examples\LCD\LCD_SegmentsDrive\.

This example performs the following actions:

1. The first step consists in displaying in scrolling mode an ‘STM8L’ string on LCD display.
2. The ‘LP MODE’ string is then displayed, and the MCU enters Active-halt mode.
3. After 20 seconds the MCU is woken up from Active-halt mode by the RTC wakeup event and continue processing.
4. These steps are executed in a infinite loop.

The flowcharts of this example are presented in *Figure 12* and *Figure 13*.

Figure 12. Wakeup from low power mode example: main program flowchart

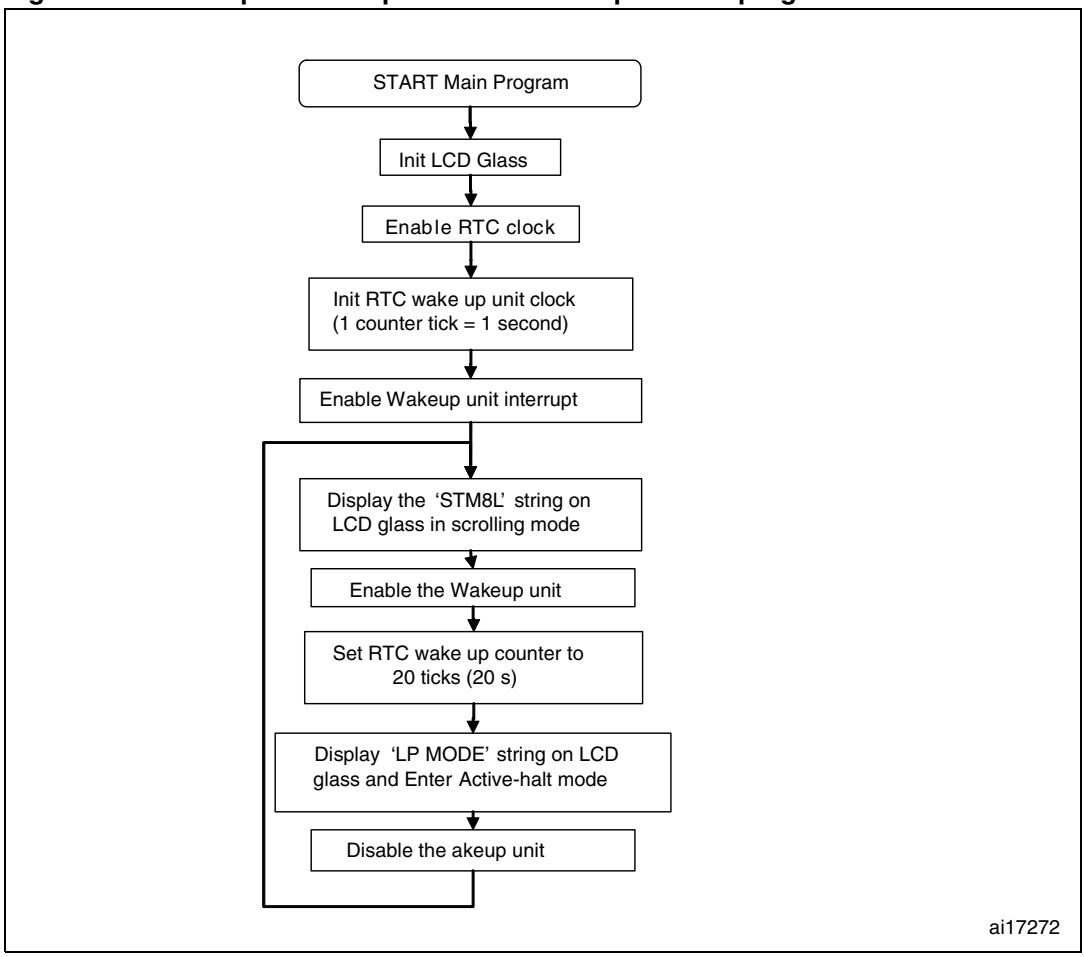
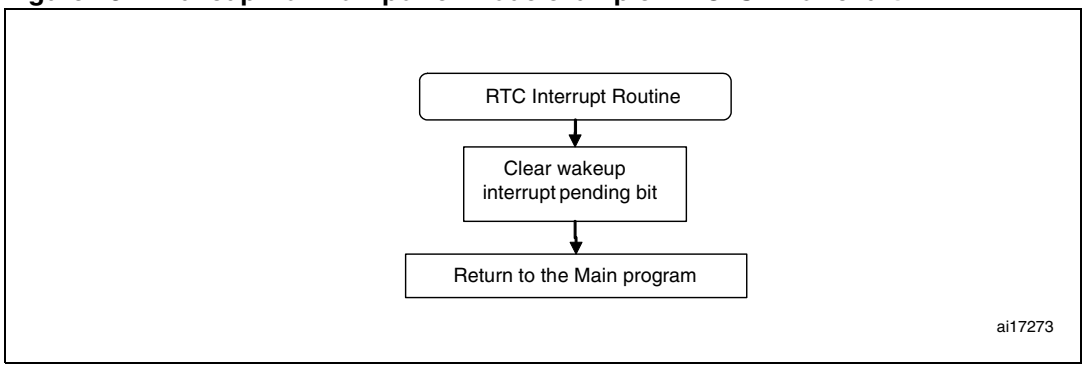


Figure 13. Wakeup from low power mode example: RTC ISR flowchart



4.2.3 Example 3: periodic event generation using the wakeup unit

This example explains how to configure the RTC periodic wakeup unit event to toggle LEDs each 500 ms.

The LSE clock is the RTCCLK source (default RTC clock) and the wakeup timer clock selection is configured to RTCCLK/16 (WUCKSEL[2:0]= 000b) to obtains a wakeup period

resolution of 488.28125 µs. The wakeup unit 16-bit timer downcounter is loaded with 1023 to generate a periodic event every 500 ms (see equation below).

Periodic event = (Timer_DownCounter +1) x Timer_resolution = 1024 step = 500 ms

The code for this example is part of the present application note firmware package. It is available under STM8L15x_AN3133_FW_V1.0.0\Project\RTC_PeriodicWakeup500ms.

5 Revision history

Table 12. Document revision history

Date	Revision	Changes
02-Feb-2010	1	Initial release
08-Mar-2010	2	Updated PREDIV_A[5:0] for maximum RTC_CALIB frequency in *Table 7: RTC_CALIB output frequency versus clock source*.