AN3133
Application note

Using the STM8L05xxx/STM8L101xx/STM8L15xxx/ STM8L162xx/STM8AL31xx/STM8AL3Lxx real-time clock

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 that support RTCs can be used for chronometers, alarm clocks, watches, small electronic agendas, and many other devices.

This application note describes the features of the RTC controller embedded in medium density, medium+, and high density STM8L05xxx/STM8L101xx/STM8L15xxx/STM8L162xx/ STM8AL31xx/STM8AL3Lxx devices, together with the steps required to configure the RTC for use with the calendar, alarm, periodic wakeup unit, tamper detection and chronometer. *Table 1* shows the STM8 family members covered by this application note.

Five application examples are provided with useful configuration information to allow the user to quickly and correctly configure the RTC for calendar, alarm, periodic wakeup unit, tamper detection and chronometer applications.

Note: All examples and explanations in this document are based on the STM8L05x/STM8L15x/ STM8L16x/STM8AL31x/STM8AL3Lx standard peripheral library. Please refer to the STM8L05xx, STM8L15xx, STM8L162x, STM8AL31xx and STM8AL3Lxx microcontroller family reference manual (RM0031) for more details.

Table 1. Applicable products

Product family	Part numbers
Microcontrollers	– STM8L05xxx – STM8L101xx – STM8L151C2/F2/G2/K2, STM8L151C3/F3/G3/K3 – STM8L151x4, STM8L151x6, STM8L151x8 – STM8L152x4, STM8L152x6, STM8L152x8 – STM8L162R8, STM8L162M8 – STM8AL313x, STM8AL314x, STM8AL316x – STM8AL3L4x, STM8AL3L6x

Contents

List of tables

List of figures

1 Real-time clock overview

The real-time clock (RTC) embedded in the STM8L05xxx/STM8L101xx/STM8L15xxx/ STM8L162xx/STM8AL31xx/STM8AL3Lxx microcontrollers (referred to in this document as STM8L/STM8AL) can be used to provide a full-featured calendar, alarm and periodic wakeup unit.

Additional features are available on medium+ and high density devices, such as calendar synchronization, digital calibration and advanced tamper detection.

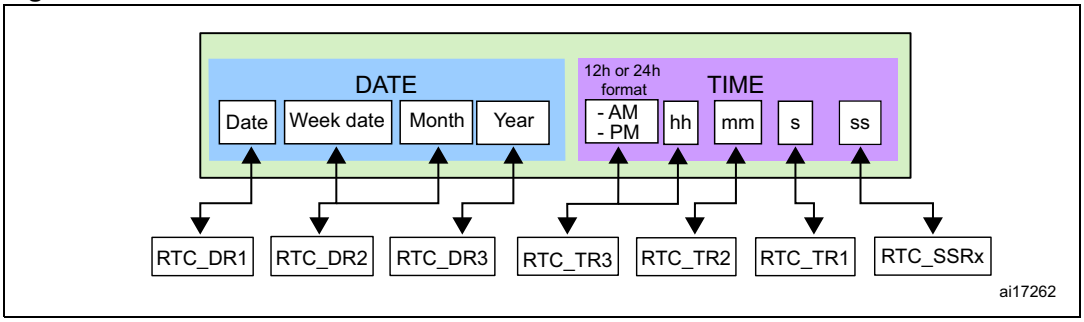
Refer to *Section 4: RTC features summary* for the complete list of features available on medium, medium+ and high density devices.

1.1 RTC calendar

A calendar keeps track of the time (hours, minutes, seconds and sub-seconds) and date (day, week, month, year). The STM8L/STM8AL 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 (binary-coded decimal) format
- Sub-second field in binary format, on medium+ and high density devices
- Automatic management of 28-, 29- (leap year), 30-, and 31-day months
- Daylight saving time adjustment programmable by software

Figure 1. STM8L/STM8AL RTC calendar fields



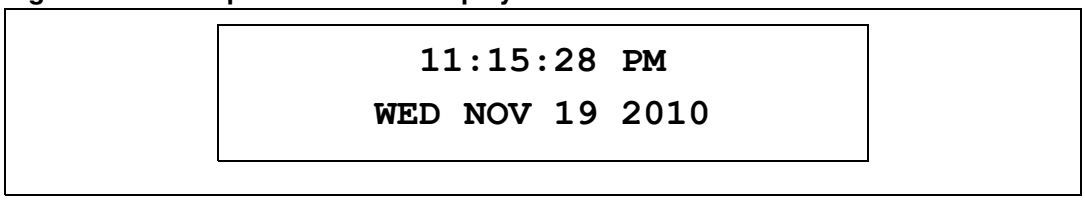
1. RCT_DRx, RTC_TRx and RTC_SSRx are RTC registers.
2. RTC_SSRx registers are read only.

A software calendar can be a kind of software counter (usually 32 bits 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 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 use significant program memory space and are CPU-time consuming which may be critical in certain real-time applications.

When using the STM8L/STM8AL RTC calendar, software conversion routines are no longer needed because their functions are performed by hardware.

The STM8L/STM8AL RTC calendar is provided in BCD format: this saves from having to perform binary to BCD software conversion routines, which use significant program memory space and CPU-time that may be critical in certain real-time applications.

Figure 2. Example of calendar display on an LCD



1.1.1 Sub-seconds

Sub-second values are read from RTC registers RTC_SSRH and RTC_SSRL.

The sub-seconds field is adjustable and can be up to 0xFFFF, or 65535 in decimal, depending on the value set on RTC_SSRH and RTC_SSRL.

SSSS[15:0] (included in RTC_SSRH/RTC_SSRL) is the value in the synchronous prescaler's counter. Given that this counter continually counts down to zero and then reloads the value from RTC_SPRE[14:0], following is the formula for calculating the fraction of a second:

$$\text{Second fraction} = (\text{PREDIV\_S} - \text{SS}) / (\text{PREDIV\_S} + 1)$$

For example: If RTC_SPRE[14:0] = 0x7FFF, then calendar sub-seconds SS starts downcounting from 0x7FFF to 0. which means that the sub-second resolution is equal to 1/(PREDIV_S + 1) = 30.517578125 µs.

Note: 1 The sub-seconds field can be up to 0xFFFF when using the "shift control" feature, by adding 0x7FFF sub-second fractions.

2 SS can be larger than RTC_SPRE only after a shift operation. In this case, the "second fraction" is negative which (intuitively) indicates that the correct time/date is at least a second less than indicated by RTC_TRx/RTC_DRx.

1.1.2 Fine RTC calendar adjustments

For accurate RTC adjustments, a "shift control" feature enables the user to add/subtract a number of sub-seconds to/from the current calendar.

The shift is used to synchronize the RTC to a master clock: SS[15:0] (included in RTC_SSRH/RTC_SSRL) can be read with RTCCLK/PREDIV_A resolution, and a correction can be applied with RTCCLK/(PREDIV_A+1).

The number of sub-seconds that can be added is "1s– *n*" and the number of sub-seconds that can be subtracted is "–*n*" (where *n* can be up to 32767 (0x7FFF) sub-seconds).

RTC calendar adjustment examples

If RTC_SPRE[14:0] = 1023, RTC_APRE = 31 and RTC current calendar time is 3h, 25mn, 32s and SS = 511, the calendar time is read as 3h, 25mn, 32s and 500ms (03h25'32"500) since (1023–511)*32/32768 = 500 ms.

Example 1: If the user performs a negative shift in time of 100ms (to reach 03h25’32”400), he must subtract “102” subseconds (102 = 100 ms * 32768 / 32). This means that the sub-second[15:0] field will be equal to 613 (511–(–102)).

This operation is performed by configuring:

– RTC_SHIFTRH_ADD1S = 0 and
– RTC_SHIFTRx_SUBFS[14:0] = 102

Example 2: If the user performs a positive shift in time of 100 ms (to reach the equivalent 03'25''32°600), he must add 1s–(1023-102 +1) sub-seconds, which means that the sub-second[15:0] field will be equal to 1433 (511–(–(1023–102+1))) and the seconds field will be equal to 33 (32+1). In this case, the user must take care about the fact that he will not read 3h25’32 in the time register right after the shift operation. He will read 3h25’33, with a sub-second value = 1433

This operation is performed by configuring:

– RTC_SHIFTRH_ADD1S = 1 and
– RTC_SHIFTRx_SUBFS[14:0] = 1023 – 102 + 1 = 922

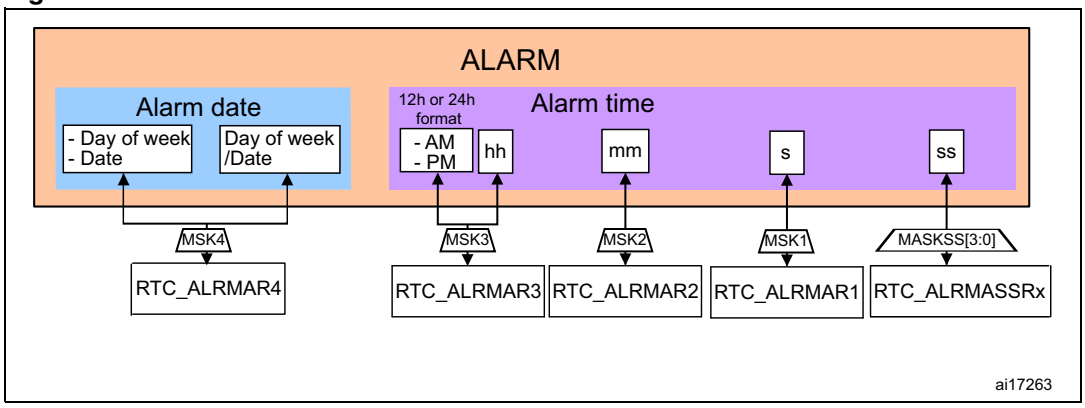
1.2 RTC alarm

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

The STM8L/STM8AL RTC provides a rich combination of alarms, and offers many features to easily configure, and display these alarms:

- Full programmable alarm: sub-seconds, seconds, minutes, hours and date fields can be independently selected or masked to provide the user a rich combination 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. RTC alarm fields



1. RTC_ALRMARx and RTC_ALRMASSRx are RTC registers.
2. MSKx and MASKSS[3:0] are bits in the RTC_ALARMx and RTC_ALRMASSMSKR registers which enable/disable the RTC_ALARMx fields used for alarm and calendar comparison. For more details refer to *Table 8*.

The alarm consists 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 STM8L/STM8AL RTC alarm can be configured by hardware to generate different types of alarms. For more details refer to *Table 8*.

1.3 RTC periodic wakeup unit

Like many low consumption microcontrollers, STM8L/STM8AL microcontrollers provide several low power modes to reduce power consumption.

STM8L/STM8AL microcontrollers feature a periodic timebase and wakeup unit that can wake up the system when the device 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 has the following features:

- Programmable downcounting auto-reload timer
- Specific flag and interrupt capable of waking 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 smooth digital calibration

The RTC clock frequency can be digitally calibrated by a series of small adjustments by adding or subtracting RTC clock cycles.

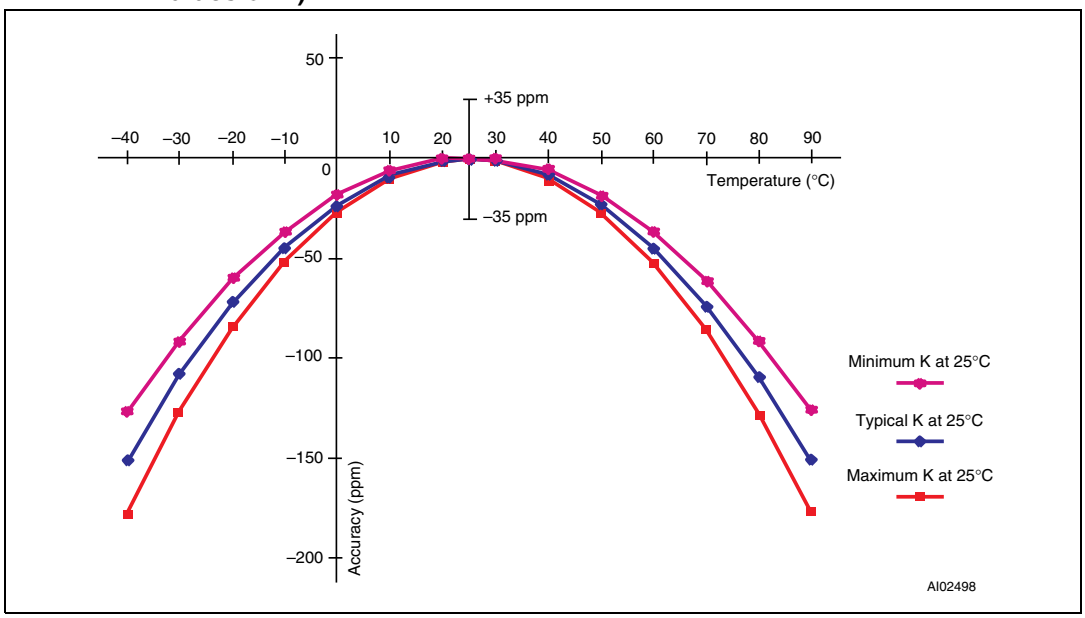
The RTC calibration block is designed to compensate the accuracy of typical crystal oscillators.

Crystal accuracy is highly dependant on:

- Temperature
- Crystal aging

Crystal accuracy is typically ±35 ppm at 25° C (see *Figure 4*) which corresponds to ±1.5 min. per month.

Figure 4. Typical crystal accuracy plotted against temperature (and different values of K)



In *Figure 4*, accuracy = K x $(T - T_O)^2$

where, T_O = 25° C ±5° C and K = –0.036 ppm/°C^2 ± 0.006 ppm/°C^2

RTC clock smooth digital calibration consists in masking N (configurable) 32 kHz clock pulses that are fairly well distributed in a configurable window (8s, 16s or 32s).

The number of masked or added pulses is configured by bits CALP and CALM[8:0] in the RTC_CALRH and RTC_CALRL registers. By default, the window is 32s. It can be reduced to 8s or 16s by setting bits CALW8 or CALW16 in the RTC_CALRH register. Reducing the calibration window allows to test the calibration result in a lesser time, which can be useful for factory tests. As a drawback, the digital calibration resolution is decreased when the window size is smaller.

The calibration range is from –487.1 ppm to +488.5 ppm, which corresponds to a correction of approximately ±0.05%.

A 1 Hz output is provided to measure the quartz crystal frequency and calibration results.

The calibration value can be changed on the fly so that it can be changed when a temperature change is detected.

The measurement window must be multiple of the calibration window.

Table 2. Calibration window description

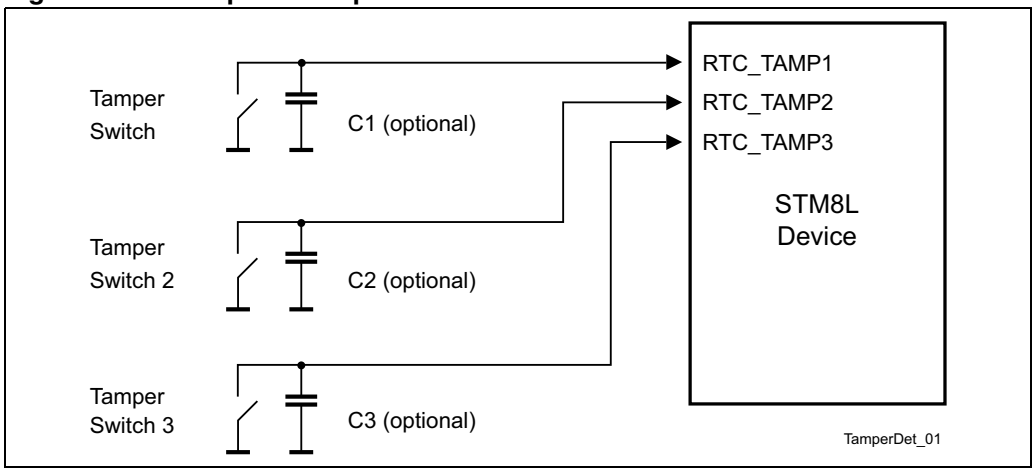
Calibration window	Accuracy	Calibration step
8 s	±1.91 ppm	3.81 ppm
16 s	±0.95 ppm	1.91 ppm
32 s	±0.48 ppm	0.95 ppm

1.5 RTC tamper detection

The RTC includes 3 tamper detection inputs. The active level can be configured independently for each tamper input. Each tamper input has an individual flag (bit RTC_ISR2_TAMPxF). A tamper detection event generates an interrupt when the RTC_TAMPCR1.TAMPIE bit is set.

This interrupt can wake up the device from Active-halt mode.

Figure 5. Example of tamper detection circuit

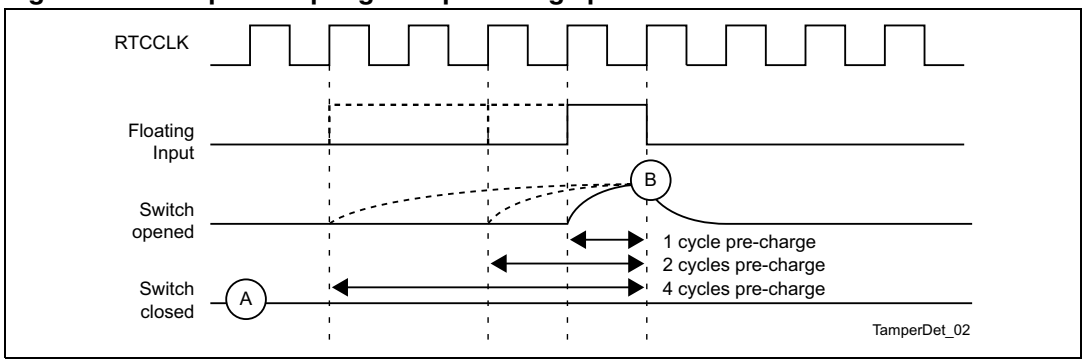


1. C1 C2 and C3 are optional (filtering can be performed by software).

The tamper inputs are sampled at a programmable rate from 1 Hz to 128 Hz (with RTCCLK at 32.768 kHz). This reduces power consumption as the pull-up is applied only during the precharge time, once every sampling period. Consequently, a trade-off must be made between the sampling frequency, which impacts the tamper detection latency, and the consumption due to the pull-up resistor.

Biasing can be performed using the MCU I/Os pull-up resistors (RTC_TCR2.TAMPPUDIS = 0). When the precharge is enabled, the length of the pulse during which the internal pull-up is applied is programmable from 1 to 8 RTCCLK cycles, in order to support different capacitance values. The RTC_TAMPx pin level is sampled at the end of this pre-charging pulse (see *Figure 6*). When the internal pull-up is not applied, the I/Os Schmitt triggers are disabled in order to avoid extra consumption if the tamper switch is open.

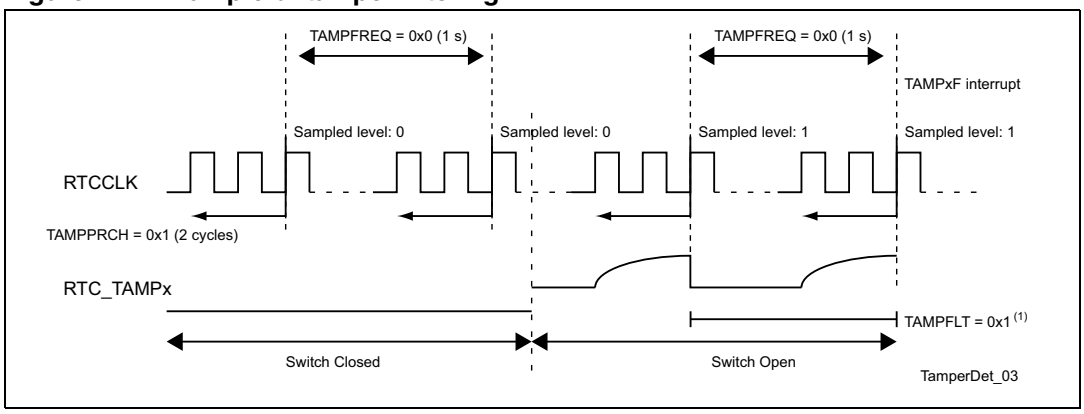
Figure 6. Tamper sampling with pre-charge pulse



Note: In Figure 6, Point B indicates where input voltage sampling is performed.

Digital filtering is performed by configuring the number of identical and consecutive active levels which must be detected in order to generate a tamper event, and an interrupt which will wake up the device from Active-halt mode. The number of consecutive active levels before issuing an event can be 1, 2, 4 or 8.

Figure 7. Example of tamper filtering



1. Tamper is set after 2 consecutive samples at the active level.

Figure 7 shows a tamper detection with the following configuration:

- TAMPxLEVEL = 0x1: High level
- TAMPFREQ = 0x0: Tamper sampling frequency = 1 Hz
- TAMPPRCH = 0x1: Tamper precharge duration = 2 cycles
- TAMPFLT = 0x1: Tamper filter count = 2 consecutive samples

1.6 RTC and low-power consumption

The STM8L/STM8AL 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 reduces the power consumption.

The RTC continues working in reset mode and its registers are not reset except by a 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 device 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 STM8L05xx, STM8L15xx, STM8L162x, STM8AL31xx and STM8AL3Lxx microcontroller family reference manual (RM0031) for more details about low power modes.

Table 3. 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	Software sequence, reset
Low power wait mode	Software sequence + WFE	LSI or LSE clock	OFF	ON	Internal or external event, reset
Active-halt mode	HALT(2)	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. 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.

1.7 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.

1.7.1 RTC_CALIB output

The RTC_CALIB output is used to generate a variable-frequency 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.

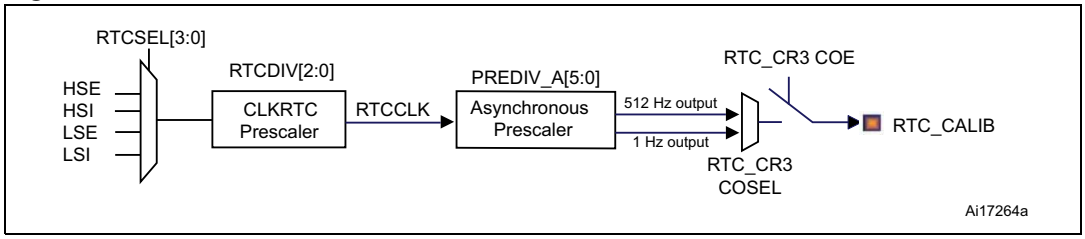
When **COSEL=0** (512Hz output), RTC_CALIB is the output of the 5th stage of the 6-bit asynchronous prescaler. So if PREDIV_A[5]=0, no signal is output on RTC_CALIB.

When RTCCLK frequency is 32.768kHz and PREDIV_A[6:0] = 0x7F, RTC_CALIB frequency is 512Hz.

When **COSEL=1(**1Hz output). RTC_CALIB is the output of the 8th stage of the 15-bit synchronous prescaler. So if PREDIV_A[6:0] = 0x7F and PREDIV_S[15:0] = 0xFF, RTC_CALIB frequency is 1Hz.

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

Figure 8. RTC_CALIB clock sources



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

1.7.2 RTC_ALARM output

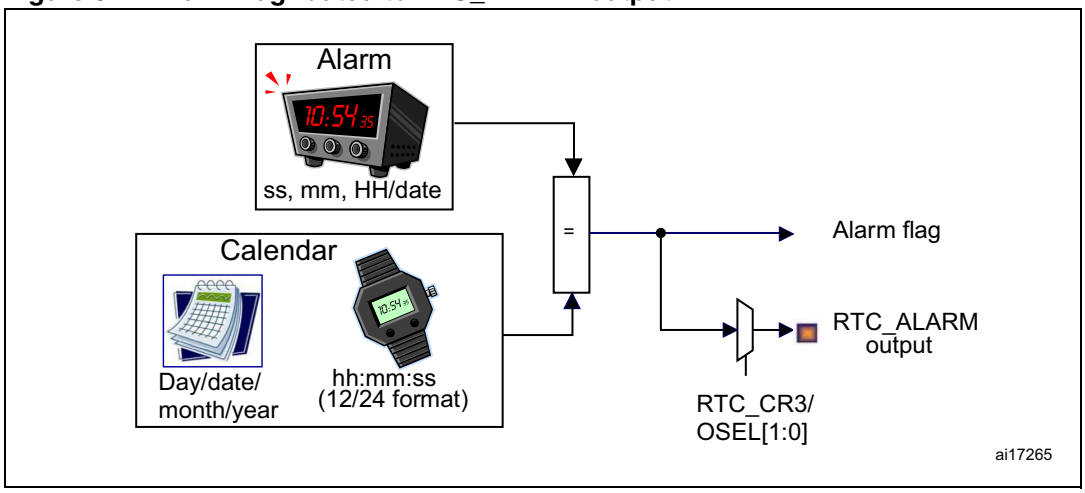
The RTC_ALARM output can either be connected to the RTC alarm unit to trigger an external action, or routed to the RTC wakeup unit to wake up an external device.

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.

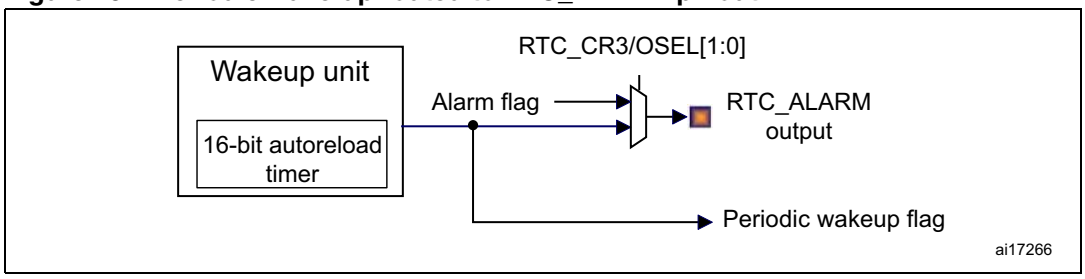
Figure 9. Alarm flag routed to RTC_ALARM output



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 10. Periodic wake-up routed to RTC_ALARM pinout



1.8 RTC security aspects

1.8.1 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 in the Write protection register (RTC_WPR).

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

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

Writing an incorrect key automatically reactivates the RTC register write access protection.

1.8.2 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 the RTC_ISR1_INITF flag is set.

To return to 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.

1.8.3 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.

On medium+ and high density devices, it is possible to directly read the calendar instead of reading shadow registers. This is configured by setting the BYPSHAD bit in the RTC_CR1 register. In this case, it is not necessary to wait for the synchronization time, but the calendar registers consistency must be checked by SW by executing a SW vote.

The user must read the required calendar fields values. Then the read operation must be performed again. The results of the two read sequence are then compared. If the results match, the read result is correct. If they do not match, the fields must be read once more, and the 3rd read result is valid.

2 Programming the RTC

2.1 Initializing the calendar

Table 4 describes the steps required to correctly configure the calendar time and date.

Table 4. Steps to initialize the calendar

Step	What to do	How to do it	Comments
1	Enter Initialization mode.	Set INIT bit to '1' in RTC_ISR1 register.	The calendar counter is stopped to allow update.
2	Wait for the confirmation of Initialization mode (clock synchronization).	Poll INITF bit of in RTC_ISR1 until it is set.	It takes approximately 2 RTCCLK clock cycles for medium density devices.
3	Program the 3 prescaler registers if needed.	Registers 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

Table 5 describes the steps required to configure the alarm.

Table 5. Steps to configure the alarm

Step	What to do	How to do it	Comments
1	Disable the alarm.	Clear ALRAE bit in RTC_CR2 register.	
2	Check that the RTC_ALRMARx registers can be accessed.	Poll ALRAWF bit until it is set in RTC_ISR1.	It takes approximately 2 RTCCLK clock cycles (clock synchronization). On medium+ and high density, there is no synchronization time to wait for.
3	Configure the alarm.	Configure RTC_ALRMARx registers.	The alarm hour format must be the same as the RTC Calendar in RTC_ALARM3(1).
4	Re-enable the alarm.	Set ALRAE bit in RTC_CR2 register.	

1. As an example, if the alarm is configured to occur 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

Table 6 describes the steps required to configure the Auto-wakeup unit.

Table 6. Steps to configure the Auto wake-up unit

Step	What to do	How to do it	Comments
1	Disable the wakeup timer.	Clear WUTE bit in RTC_CR2 register.	
2	Ensure access to Wakeup auto-reload counter and bits WUCKSEL[2:0] is allowed.	Poll WUTWF until it is set in RTC_ISR1.	It takes approximately 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 register.	
5	Re-enable the wakeup timer.	Set WUTE bit in RTC_CR2 register.	The wakeup timer restarts down-counting.

3 Useful RTC configuration examples

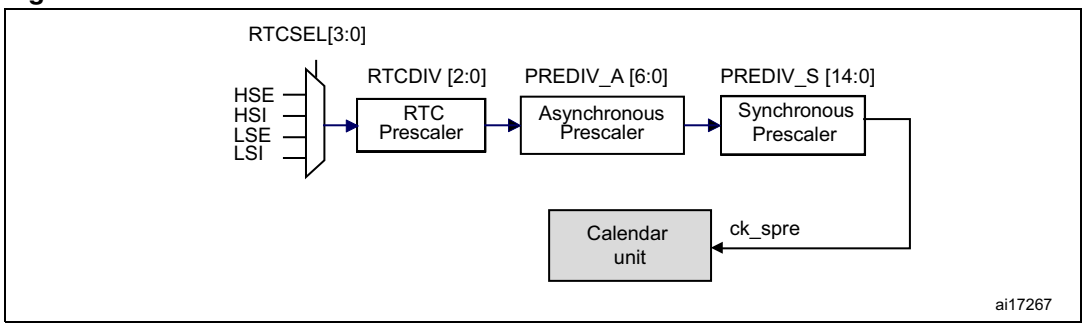
This section explains how to configure the RTC and provides examples of configurations.

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, regardless of the clock source.

Figure 11. 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:

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

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 (for medium-density products)

PREDIV_S can be 0,1,2,..., 32767 (for medium+ and high-density products)

Table 7 shows several possibilities to obtain ck_spre = 1 Hz.

Table 7. Calendar clock equal to 1Hz with different clock sources

Clock source	Prescalers			ck_spre
	RTCDIV[2:0]	PREDIV_A[6:0]	PREDIV_S(1)	
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(2)	0 (div1)	124 (div125)	303 (div304)	1 Hz

1. Bits [12:0] for medium-density products and Bits [14:0] for medium+ and high-density products.
2. LSI accuracy is not suitable for calendar application.

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) of the RTC_ALRMARx registers and the MASKSS[3:0] bits of the RTC_ALRMASSMSKR register.

Table 8 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 8. Alarm combination

MASKSS [3:0]	MSK4	MSK3	MSK2	MSK1	Alarm behavior
0x0	0	0	0	0	**All fields are used in alarm comparison:** Alarm occurs at 23:15:07, each Monday.
0x0	0	0	0	1	**Seconds don't care in alarm comparison** The alarm occurs every second of 23:15, each Monday.
0x0	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.
0x0	0	0	1	1	**Minutes and seconds don't care in alarm comparison**
0x0	0	1	0	0	**Hours don't care in alarm comparison**
0x0	0	1	0	1	**Hours and seconds don't care in alarm comparison**
0x0	0	1	1	0	**Hours and minutes don't care in alarm comparison**
0x0	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.
0x0	1	0	0	0	**Week day (or date, if selected) don't care in alarm comparison** Alarm occurs all days at 23:15:07.
0x0	1	0	0	1	**Week day and seconds don't care in alarm comparison**
0x0	1	0	1	0	**Week day and minutes don't care in alarm comparison**
0x0	1	0	1	1	**Week day, minutes and seconds don't care in alarm comparison**
0x0	1	1	0	0	**Week day and Hours don't care in alarm comparison**

Table 8. Alarm combination (continued)

MASKSS [3:0]	MSK4	MSK3	MSK2	MSK1	Alarm behavior
0x0	1	1	0	1	**Week day, Hours and seconds don't care in alarm comparison**
0x0	1	1	1	0	**Week day, Hours and minutes don't care in alarm comparison**
0x0	1	1	1	1	**Alarm occurs every second**

Table 9. Mask configurations for setting an alarm every 125 ms (for RTCCLK = 32.768kHz)

Prescaler Configuration	MSKSS[3:0]	MSK4	MSK3	MSK2	MSK1	Comment
PREDIV_A = 0x7F PREDIV_S = 0xFF	0x5	1	1	1	1	Alarm occurs every 2^5 * 128/32768 s
PREDIV_A = 0x0 PREDIV_S = 0x7FFF	0xC	1	1	1	1	Alarm occurs every 2^{12} * 1/32768 s

3.3 Maximum and minimum RTC_CALIB output frequency

On medium density devices, or when COSEL = 0, the RTC can output the RTCCLK clock divided by a 6-bit asynchronous prescaler. The divider factor is configured through bits PREDIV_A[5:0] of the RTC_APRER register.

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

Table 10. 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	30,303.030 Hz
HSI = 16 MHz	3.906 kHz	**484,848.500 Hz**
LSE = 32 768 Hz	**8.000 Hz**	992.970 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.

When COSEL = 1, the RTC output frequency is the ck_spre frequency.

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 12 shows the prescaler connection to the timebase/wakeup unit and *Table 11* gives the timebase/wakeup clock resolutions corresponding to configuration 1.

Figure 12. Prescalers connected to the timebase/wakeup unit for configuration 1

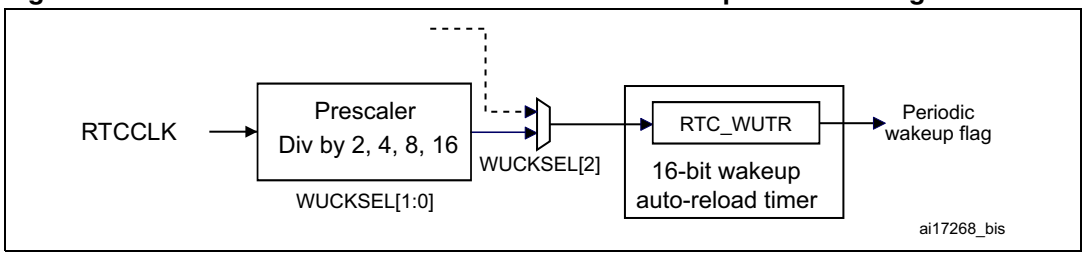


Table 11. Timebase/wakeup unit period resolution with clock configuration 1

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 prohibited. Refer to the STM8L05xx, STM8L15xx, STM8L162x, STM8AL31xx and STM8AL3Lxx microcontroller family reference manual (RM0031) 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 13 shows the prescaler connection to the timebase/wakeup unit and *Table 12* gives the timebase/wakeup clock resolutions corresponding to configuration 2 for medium density products and *Table 13* for medium+ and high-density products.

Figure 13. Prescalers connected to the wake up unit for configurations 2 and 3

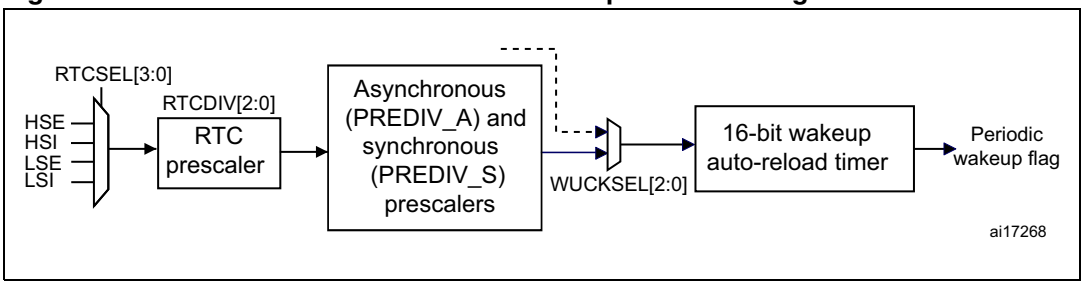


Table 12. Timebase/wakeup unit period resolution with clock configuration 2

Clock source	Wakeup period resolution	
	RTCDIV[2:0] = div64 PREDIV_A[6:0] = div128 PREDIV_S[12:0] = div8192	**RTCDIV[2:0] = div1 PREDIV_A[6:0] = div2(1) PREDIV_S[12:0] = div1**
HSE = 1 MHz	67.10 s	2 µs
HSI = 16 MHz	4.19 s	**125 ns**
LSE = 32 768 Hz	**2048 s**	61 µs
LSI = 38 kHz	1766.02 s	52.6 µs

1. PREDIV_A minimum value is '1' on medium density devices.

Table 13. Timebase/wakeup unit period resolution with clock configuration 2

Clock source	Wakeup period resolution	
	RTCDIV[2:0] = div64 PREDIV_A[6:0] = div128 PREDIV_S[14:0] = 32768	**RTCDIV[2:0] = div1 PREDIV_A[6:0] = div2(1) PREDIV_S[14:0] = div1**
HSE = 1 MHz	268.43 s	1 µs
HSI = 16 MHz	16.77 s	**62.5 ns**
LSE = 32 768 Hz	**8192 s**	30.5 µs
LSI = 38 kHz	7064.09 s	26.3 µs

1. PREDIV_A minimum value is '1' on medium density devices.

The minimum resolution for configuration 2 is 0.125 µs, and the maximum resolution 8192 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 8192 s = 536870912 s (more than 16 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.

- For medium-density products
  - The minimum timebase/wakeup period is:
    (0x10000 + 1) x 125 ns = 8.19 ms
  - The maximum timebase/wakeup period is:
    (0x1FFFF+ 1) x 2048 s = more than 8 years
- For medium+ and high-density products
  - The minimum timebase/wakeup period is:
    (0x10000 + 1) x 62.5 ns = 4.09 ms
  - The maximum timebase/wakeup period is:
    (0x1FFFF+ 1) x 8192 s = more than 33 years

3.4.4 Summary of timebase/wakeup period extrema

The minimum and maximum period values, according on the configuration, are listed in *Table 14* for medium density products and *Table 15* for medium+ and high-density products.

Table 14. Min. and max. timebase/wakeup period (medium density products)

Configuration	Minimum period	Maximum period
1	250 ns	2048 s
2	**125 ns**	More than 4 years
3	8.192125 ms	**More than 8 years**

Table 15. Min. and max. timebase/wakeup period (med+ and high-density products)

Configuration	Minimum period	Maximum period
1	0.250 µs	2048 s
2	**62.5 ns**	More than 16 years
3	4.096 ms	**More than 33 years**

4 RTC features summary

Table 16. Summary of RTC features by product family

RTC features		Medium+ and High density	Medium density
Prescalers	Asynchronous	7 bits	Same
	Synchronous	15 bit	13 bit
Bypass shadow		Available	Not available
Calendar	Time	– 12h/24h time format – Hours – Minutes – Seconds	Same
	Sub-seconds	– Sub-seconds	Not available
	Date	– Weekday – Date – Month – Year	Same
	Daylight operation	Add or subtract 1 hour to compensate for daylight savings time.	Same
Wake-up unit		3 possible configurations for: – Short period wakeup – Medium period wakeup – Long period wakeup	Same but less than high-density products due to additional Synchronous Prescaler bits.
Alarm	Time	– 12h/24h time format – Hours – Minutes – Seconds	Same
	Sub-seconds	– Sub-seconds	Not available
	Date	Date or Weekday	Same
	Masks	Masks for time, date and sub-seconds	Masks for time and date only.
RTC outputs	RTC_ALARM pin	Depending on configuration, it can deliver Alarm or WakeUp status.	Same
	RTC_CALIB pin	Depending on configuration, it can deliver 1-Hz or 512-Hz signal.	512-Hz signal only
Tamper detection		– Configurable filter – Configurable sampling frequency – Configurable pins input precharge duration – Configurable sensitivity	Not available
Shift control		Add and subtract sub-seconds to adjust the time.	Not available
Smooth calibration		Recalibrate the RTC clock for crystal accuracy compensation.	Not available

5 RTC firmware API

5.1 Function groups

The STM8L/STM8AL RTC driver can be divided into 11 function groups related to the functions embedded in the RTC peripheral.

1. RTC initialization and configuration
2. RTC time and date
3. RTC alarm
4. RTC wakeup
5. RTC daylight saving
6. RTC output pin configuration
7. RTC calibration output pin
8. RTC flags and interrupts
9. RTC tamper detection configuration
10. RTC synchronization shift control
11. RTC smooth calibration

Table 17. RTC function groups

Group ID	Function name	Description	High density	Medium density
1	RTC_DeInit	Deinitializes the RTC registers to their default reset values.	Yes	Yes
	RTC_Init	Initializes the RTC registers according to the specified parameters in RTC_InitStruct <Hour format, Asynchronous predivisor, Asynchronous predivisor>.	Yes	Yes
	RTC_StructInit	Fills each RTC_InitStruct member with its default value.	Yes	Yes
	RTC_EnterInitMode	Enters the RTC Initialization mode.	Yes	Yes
	RTC_ExitInitMode	Exits the RTC Initialization mode.	Yes	Yes
	RTC_WriteProtectionCmd	Enables or disables the RTC registers write protection.	Yes	Yes
	RTC_WaitForSynchro	Waits until the RTC Time and Date registers (RTC_TRx and RTC_DRx) are synchronized.	Yes	Yes
	RTC_RatioCmd	Configures the RTC ratio.	Yes	Yes
	RTC_TimeStructInit	Fills each RTC_TimeStruct member with its default value (Time = 00h:00min:00sec).	Yes	Yes
	RTC_DateStructInit	Fills each RTC_DateStruct member with its default value (Monday 01 January xx00).	Yes	Yes
	RTC_AlarmStructInit	Fills each RTC_AlarmStruct member with its default value (Time = 00h:00mn:00sec / Date = 1st day of the month/Mask = all fields are masked).	Yes	Yes
	RTC_BypassShadowCmd	Enables or disables the Bypass Shadow feature.	Yes	

Table 17. RTC function groups (continued)

Group ID	Function name	Description	High density	Medium density
2	RTC_SetTime	Sets the RTC current time < RTC hours, RTC minutes, RTC seconds, RTC 12-hour clock period (AM/PM)>.	Yes	Yes
	RTC_SetDate	Sets the current RTC date. < Calendar weekday, Calendar Month, Calendar date, Calendar year>.	Yes	Yes
	RTC_GetTime	Gets the current RTC time.	Yes	Yes
	RTC_GetDate	Gets the current RTC date.	Yes	Yes
	RTC_GetSubSecond	Gets the current RTC Calendar Subseconds value.	Yes	
3	RTC_SetAlarm	Sets the RTC alarm configuration. < Alarm time fields, Alarm masks, Alarm date/Weekday selection, Alarm Date/Weekday value>.	Yes	Yes
	RTC_GetAlarm	Gets the RTC alarm configuration.	Yes	Yes
	RTC_AlarmCmd	Enables or disables the RTC alarm.	Yes	Yes
	RTC_AlarmSubSecondConfig	Configures the RTC Alarm Subseconds value and mask.	Yes	
4	RTC_WakeUpClockConfig	Configures the RTC wakeup clock source.	Yes	Yes
	RTC_SetWakeUpCounter	Sets the RTC Wakeup counter value.	Yes	Yes
	RTC_GetWakeUpCounter	Returns the RTC Wakeup timer counter value.	Yes	Yes
	RTC_WakeUpCmd	Enables or disables the RTC Wakeup timer.	Yes	Yes
5	RTC_DayLightSavingConfig	Adds or subtracts one hour from the current time depending on the daylight saving parameter.	Yes	Yes
	RTC_GetStoreOperation	Returns the daylight saving stored operation.	Yes	Yes
6	RTC_OutputConfig	Configures the RTC output for the output pinout (RTC_ALARM pin)	Yes	Yes
7	RTC_CalibOutputCmd	Enables or disables the connection of the RTCCLK/PREDIV_A[6:0] clock to be output through the relative pinout (RTC_CALIB pin).	Yes	Yes
	RTC_CalibOutputConfig	Configures the Calib Pinout (RTC_CALIB) Selection (1 Hz or 512 Hz).	Yes	only 512Hz is available
8	RTC_ITConfig	Enables or disables the specified RTC interrupts.	Yes	Yes
	RTC_GetFlagStatus	Checks whether the specified RTC flag is set or not.	Yes	Yes
	RTC_ClearFlag	Clears the RTC pending flags.	Yes	Yes
	RTC_GetITStatus	Checks whether the specified RTC interrupt has occurred or not.	Yes	Yes
	RTC_ClearITPendingBit	Clears the RTC interrupt pending bits.	Yes	Yes

Table 17. RTC function groups (continued)

Group ID	Function name	Description	High density	Medium density
9	RTC_TamperFilterConfig	Configures the Tampers Filter.	Yes	
	RTC_TamperSamplingFreqConfig	Configures the Tampers Sampling Frequency.	Yes	
	RTC_TamperPinsPrechargeDuration	Configures the Tampers Pins input Precharge Duration.	Yes	
	RTC_TamperLevelConfig	Configures the Tamper Sensitive Level.	Yes	
	RTC_TamperCmd	Enables or disables the Tamper detection.	Yes	
10	RTC_SynchroShiftConfig	Configures the Synchronization Shift Control Settings.	Yes	
11	RTC_SmoothCalibConfig	Configures the Smooth Calibration Settings.	Yes	

6 Application examples

STM8L/STM8AL RTC firmware is provided with a set of examples so that the user can quickly become familiar with the firmware library.

This section provides five 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.
- The fourth example shows how to use the tamper detection features.
- The fifth example shows how to use calendar and tamper detect features to implement an accurate chronometer.

6.1 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 STM8L05x/STM8L15x/STM8L16x/STM8AL31x/ STM8AL3Lx standard peripheral library available from http://wwwst.com/stm8l (documents and files for the STM8L family). It is located at STM8l15x_StdPeriph_Lib.zip\Project\ STM8L15x_StdPeriph_Examples\RTC\RTC_WakeupLPMode\

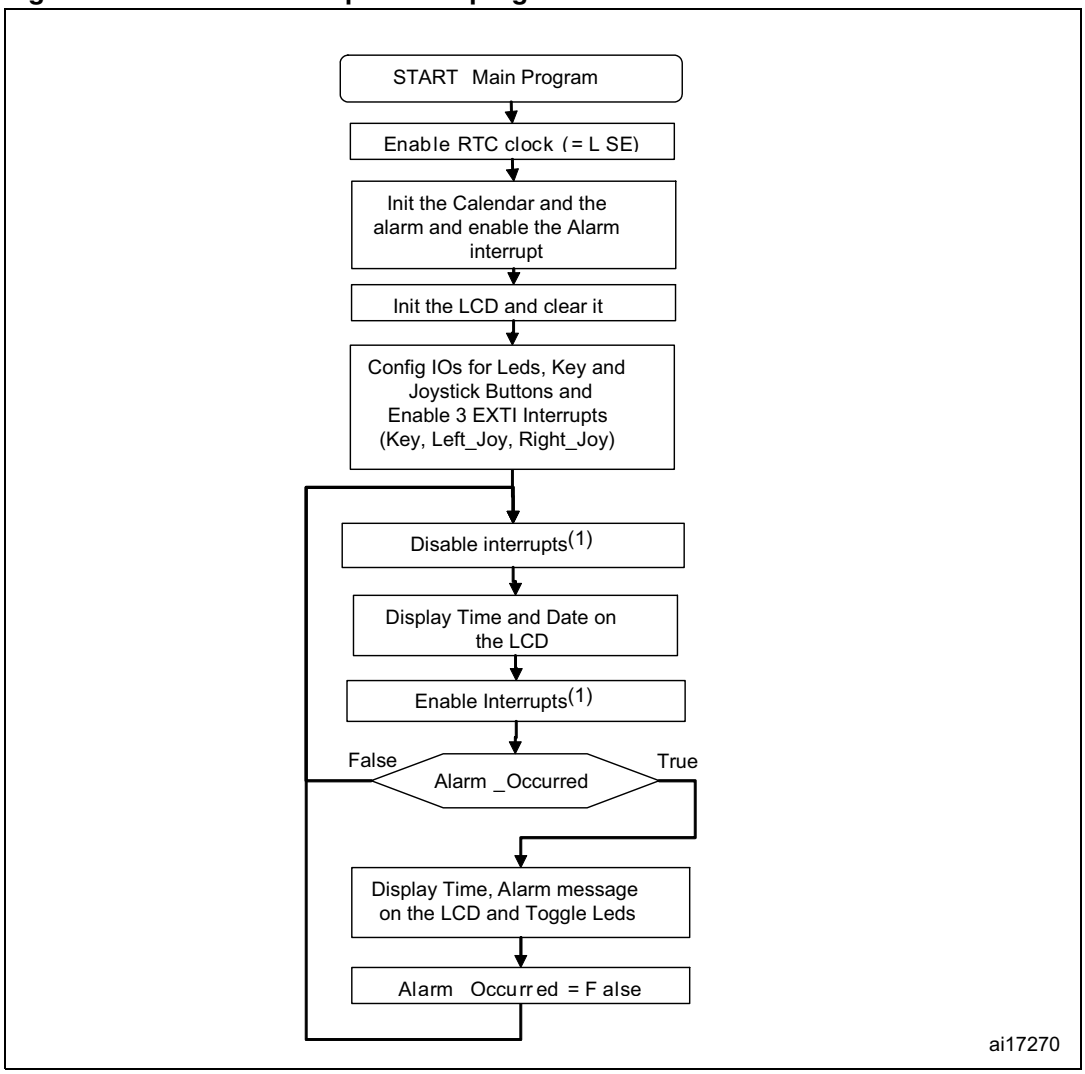
This example runs on both the STM8L1526-EVAL and STM8L1528-EVAL boards. It enables configuring the RTC calendar and alarm and displaying their values in real-time using the LCD and joystick.

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.

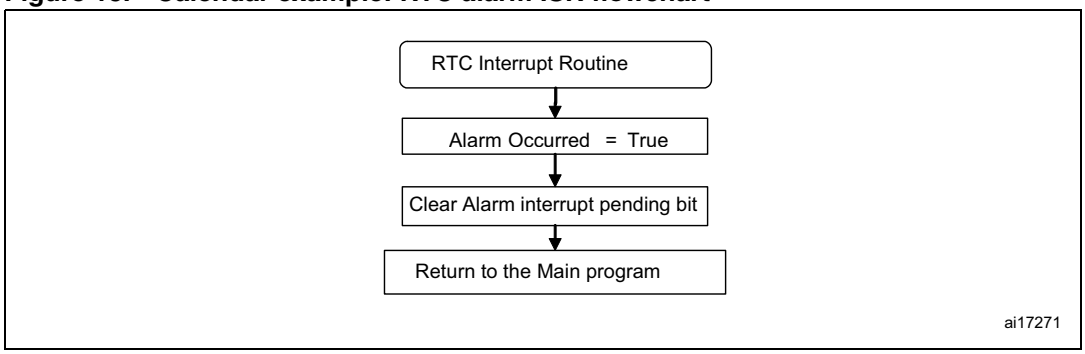
The flowcharts of this example are presented in *Figure 14* and *Figure 15*.

Figure 14. Calendar example: main program flowchart



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

Figure 15. Calendar example: RTC alarm ISR flowchart



6.2 Example 2: Wakeup from low power mode

This example explains how to use the STM8L15xx LCD embedded controller to drive the LCD glass mounted on STM8L1526-EVAL and STM8L1528-EVAL boards and how to exit the MCU from Active-halt mode using the wakeup unit.

This example is delivered within the STM8L05x/STM8L15x/STM8L16x/STM8AL31x/ STM8AL3Lx standard peripheral library available from http://wwwst.com/stm8l (documents and files for the 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 an ‘STM8L’ string on the LCD display in scrolling mode.
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 continues processing.
4. These steps are executed in an infinite loop.

The flowcharts of this example are presented in *Figure 16* and *Figure 17*.

Figure 16. Wakeup from low power mode example: main program flowchart

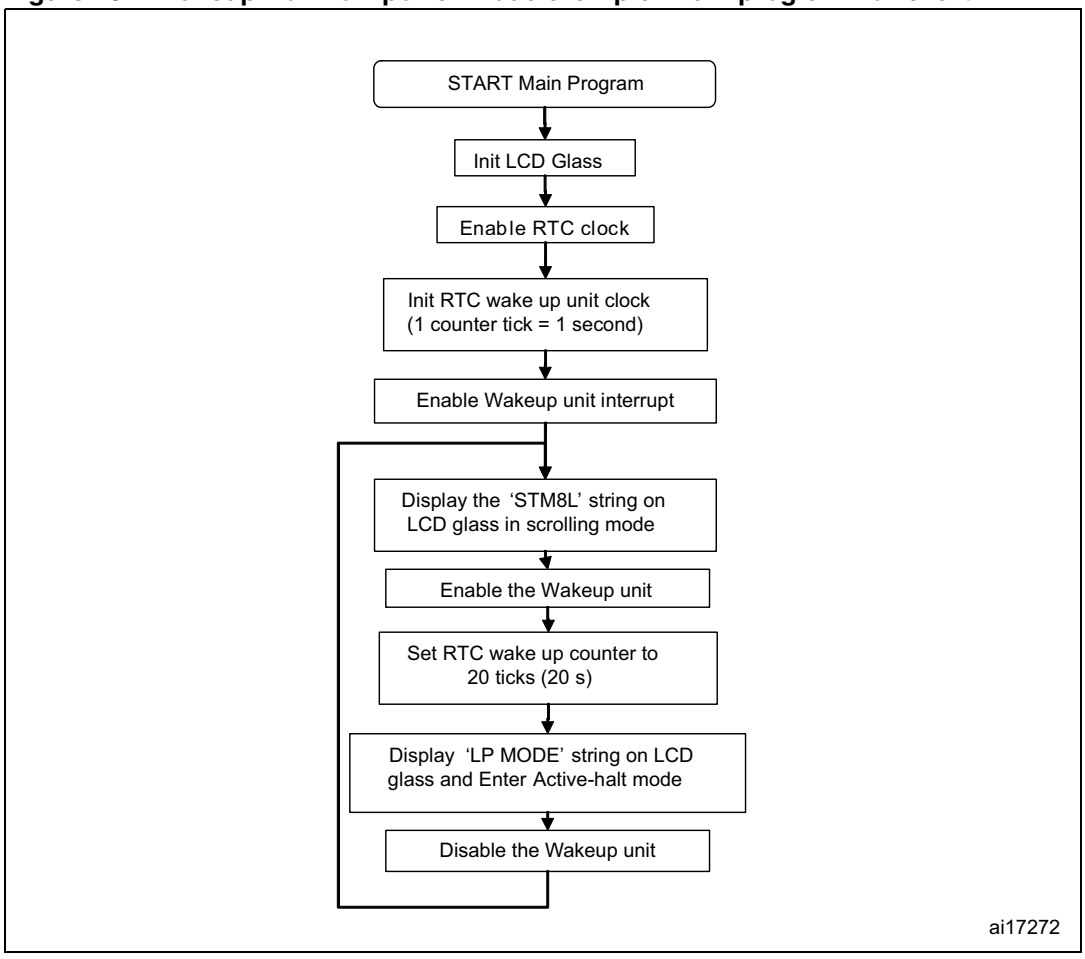
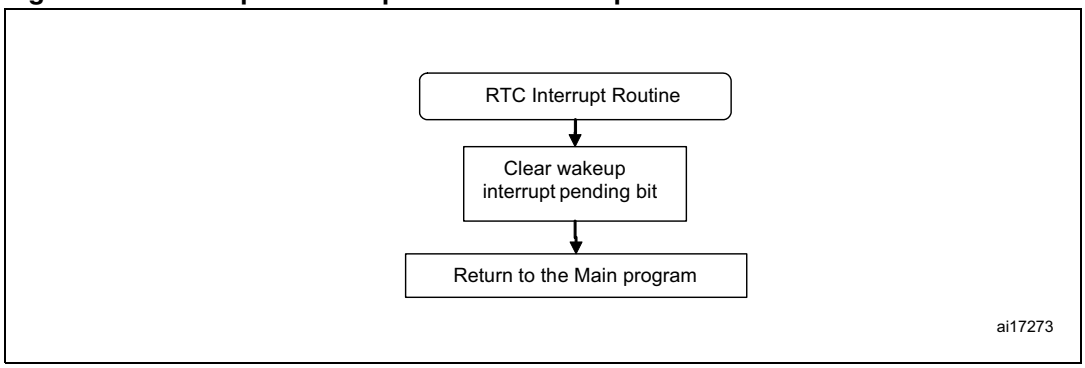


Figure 17. Wakeup from low power mode example: RTC ISR flowchart



6.3 Example 3: Periodic event generation using the wakeup unit

This example explains how to configure the RTC periodic wakeup unit event to toggle LEDs every 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

This example is delivered within the STM8L05x/STM8L15x/STM8L16x/STM8AL31x/ STM8AL3Lx standard peripheral library available from STM8l15x_StdPeriph_Lib.zip\Project\STM8L15x_StdPeriph_Examples\RTC\RTC_Periodic Wakeup500ms\

6.4 Example 4: Tamper detection

This example provides a short description of how to use the RTC peripheral's tamper detection features: sample duration, filter, and number of samples to wait for before the Tamper event.

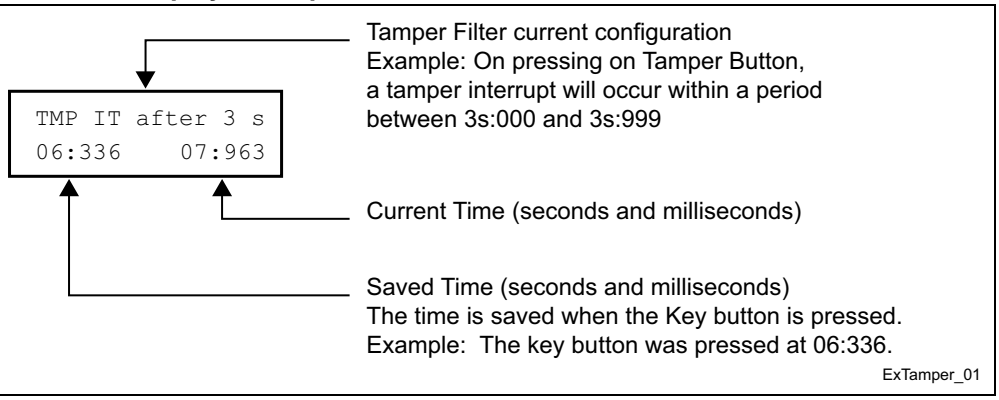
The user can manipulate the application using the Tamper, Key and Joystick navigation buttons.

After startup, 2 counters (00:000 00:000) are displayed on the LCD. Both of them consist of 2 fields [seconds on 2 digits]:[milliseconds on 3 digits].

The first counter is used to store the time when the key button is pressed.

The second counter is used to display the current time.

Figure 18. LCD display description



User can manipulate the application using the Tamper, Key and Joystick navigation buttons, the buttons actions are described in the following table.

Table 18. Buttons and corresponding actions

Button	Action
Joystick UP	To increase the delay between the Tamper 1 button press and the Tamper 1 Interrupt. Four Pre-configurations can be selected: 0s / 1s / 3s / 7s. Note: 0s means that a tamper interrupt occurs between 0s:000 and 0s:999.
Joystick DOWN	Same as the Joystick UP, but to decrease instead on increasing the delay between the Tamper 1 button press and the Tamper 1 Interrupt.
KEY	Store the current Time and display it on “saved time” LCD side
TAMPER	If it is kept pressed during the delay (selected using Joystick UP or DOWN buttons), the Tamper Interrupt occurs, LED3 is On, the MCU enters in the halt mode and LCD display is frozen. When the CPU exits Halt mode, LED3 is off. Note: To exit from Halt mode, press the Key button (Joystick UP or DOWN can also be used).

To observe the exact time spent between the Tamper 1 button press and the Tamper 1 Interrupt, the user must simultaneously press the Key button and Tamper button until LED3 switches on; meaning that Tamper1 Interrupt occurred. Then the User can check the delay value between the Tamper 1 button press and the Tamper 1 Interrupt.

This example is delivered within the STM8L05x/STM8L15x/STM8L16x/STM8AL31x/ STM8AL3Lx standard peripheral library available from STM8l15x_StdPeriph_Lib.zip\Project\STM8L15x_StdPeriph_Examples\RTC\RTC_Tamper1 Detection\

The flowcharts below explain the example procedures.

Figure 19. Main program flowchart

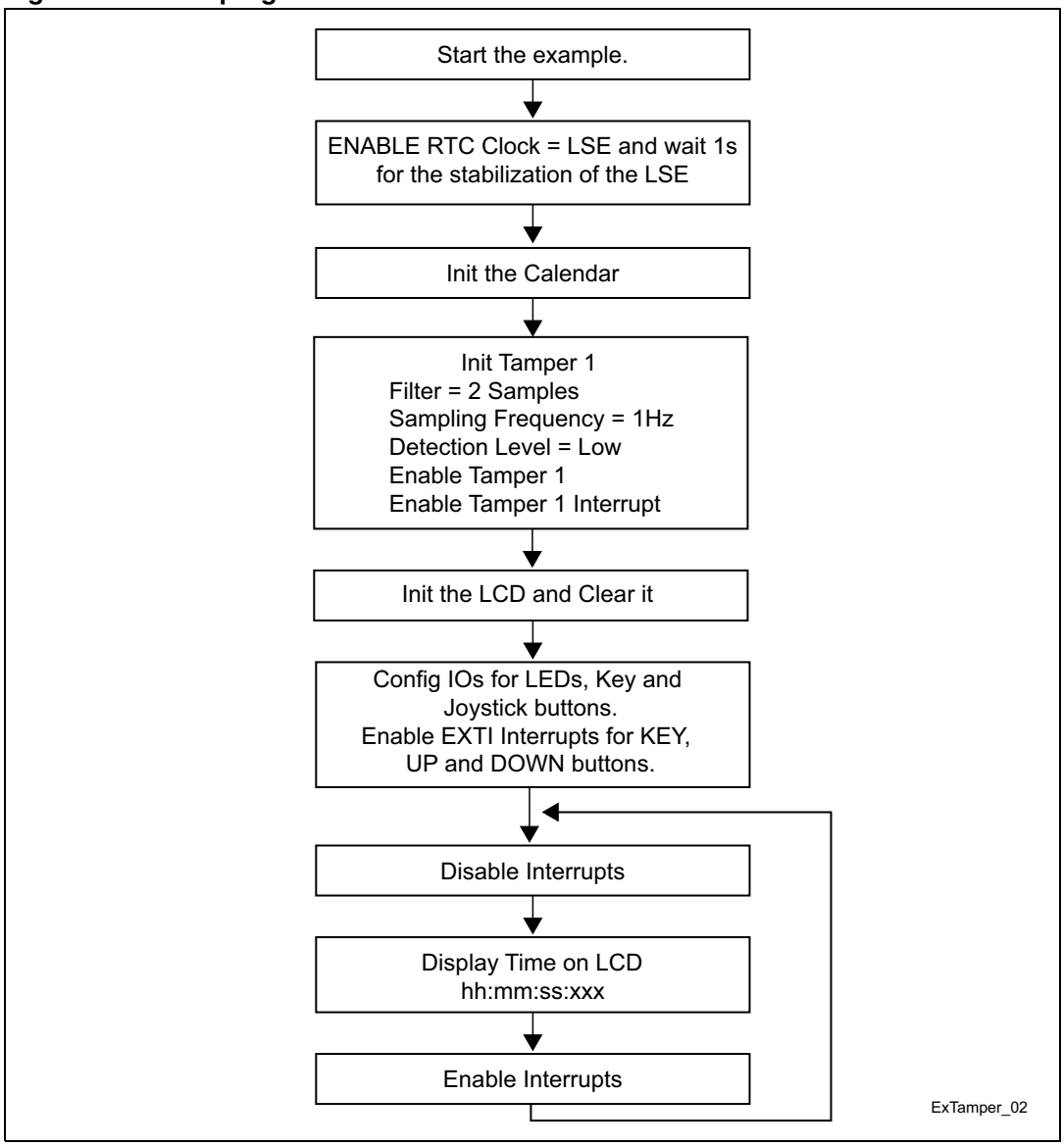


Figure 20. RTC Tamper ISR flowchart

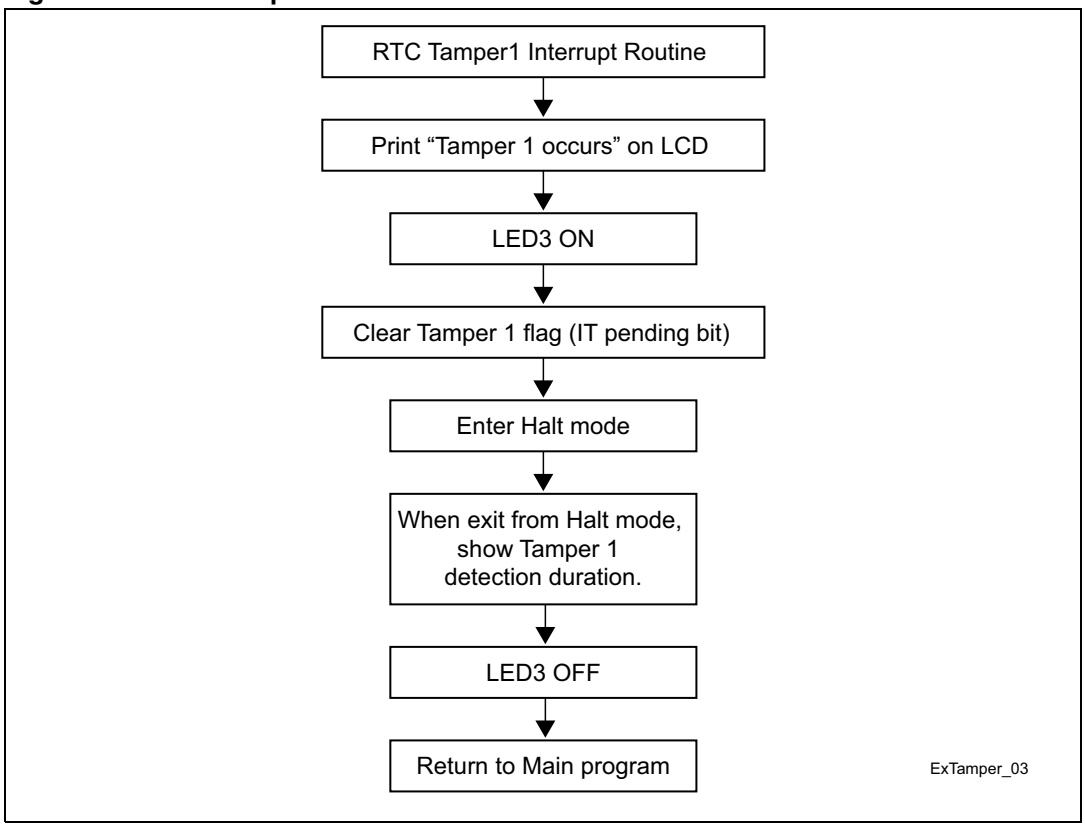


Figure 21. EXTI ISR flowchart for UP button

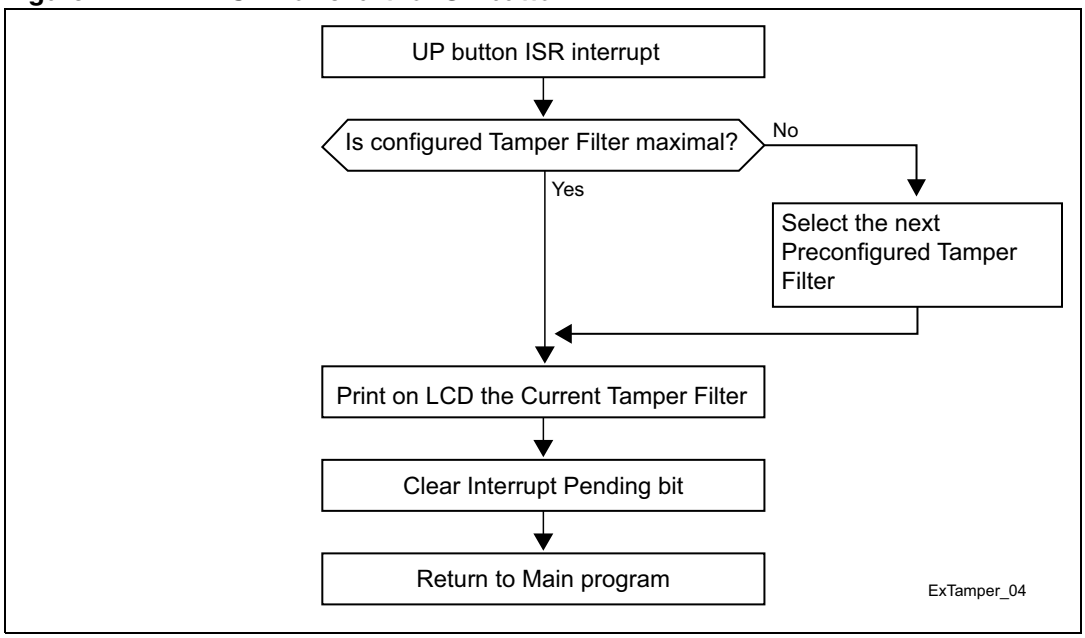


Figure 22. EXTI ISR flowchart for DOWN button

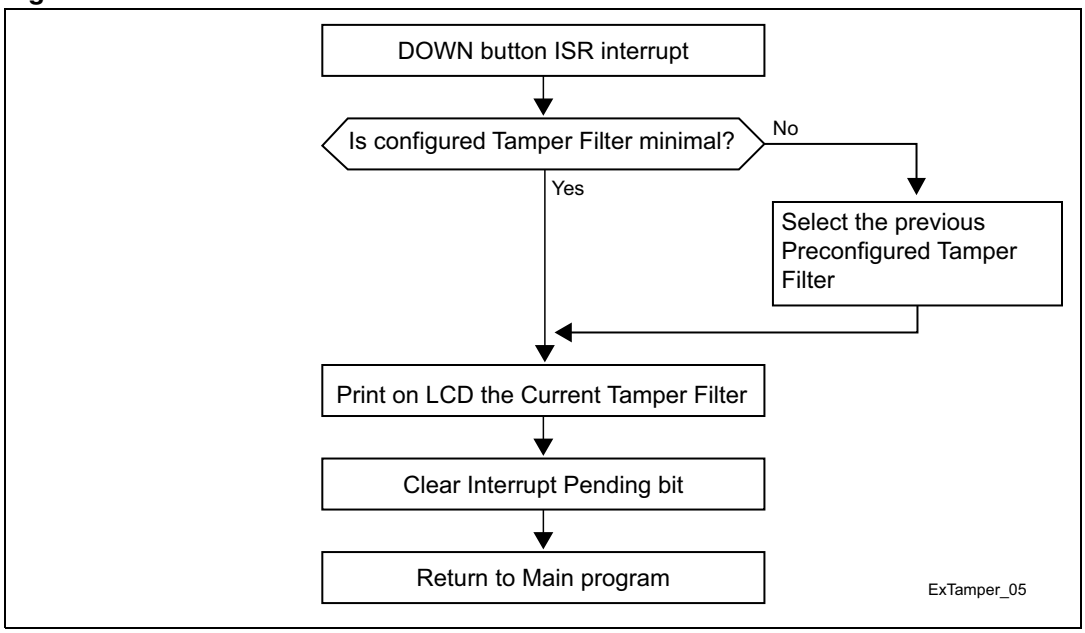
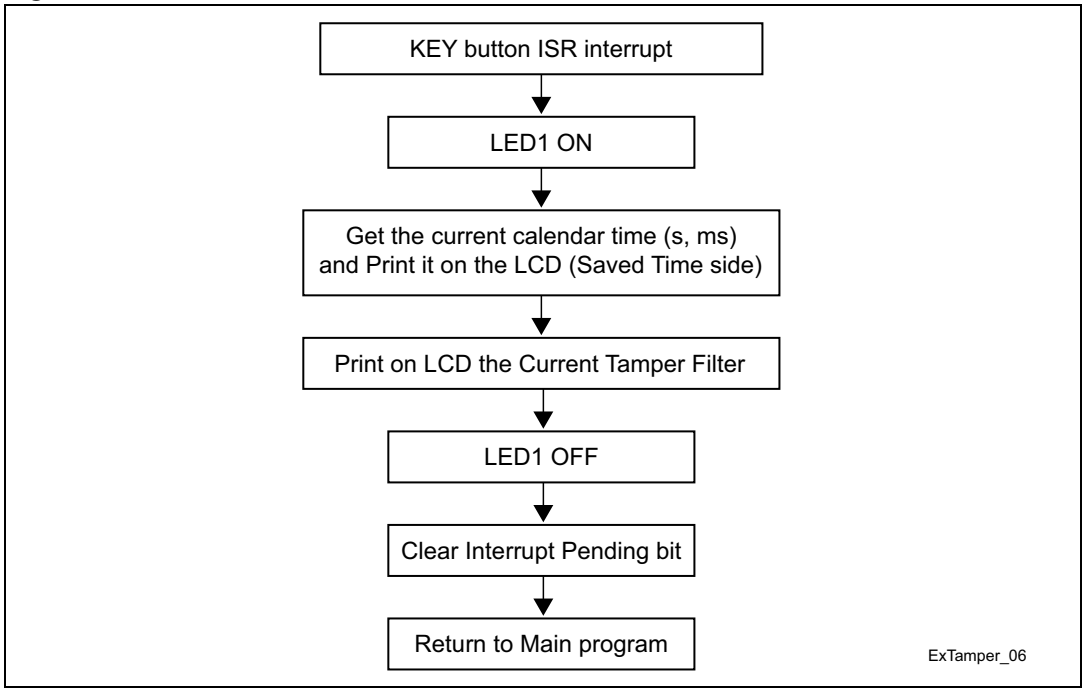


Figure 23. EXTI ISR flowchart for KEY button



6.5 Example 5: Chronometer

This example provides a short description of how to use the RTC peripheral's calendar (sub-seconds, seconds, minutes, hours) and Tamper features to simulate an accurate chronometer with 9 counter records, and Start/Pause/Resume actions.

This example is delivered within the STM8L05x/STM8L15x/STM8L16x/STM8AL31x/ STM8AL3Lx standard peripheral library available from STM8l15x_StdPeriph_Lib\Project\STM8L15x_StdPeriph_Examples\RTC\RTC_ChronoSub Second\

In this example an interactive human interface is developed using STM8L1528-EVAL's LCD and joystick to allow user to use the chronometer with the real-time display.

After startup, a default 00:00:00:000 chronometer counter is displayed on the LCD, which correspond to [Hours]:[minutes]:[seconds]:[milliseconds].

User can manipulate the chronometer features using the Tamper, Key and Joystick navigation buttons.

The buttons actions are described in the following table.

Table 19. Buttons and corresponding actions

Button	Action
Joystick SEL	To Start/Pause/Resume the chronometer counter
Joystick DOWN	To enter to the “Recorded Times menu”, where you can navigate using the Joystick RIGHT/LEFT buttons.
Joystick UP	to exit from the “Recorded Times menu”
TAMPER	Keep it pressed during 2sec to enter to the “reset menu” where you can select using RIGHT/LEFT buttons to clear the counter and/or the recorded times. Note: During Pause, Pressing on Tamper Button does not have any effect.
Joystick LEFT Joystick RIGHT	Used in the “reset menu” to clear the counter and/or the recorded times.
KEY	To store the current chronometer counter and rank. Note: 9 records can be performed.

The flowcharts below explain the example procedures.

Figure 24. Main program flowchart

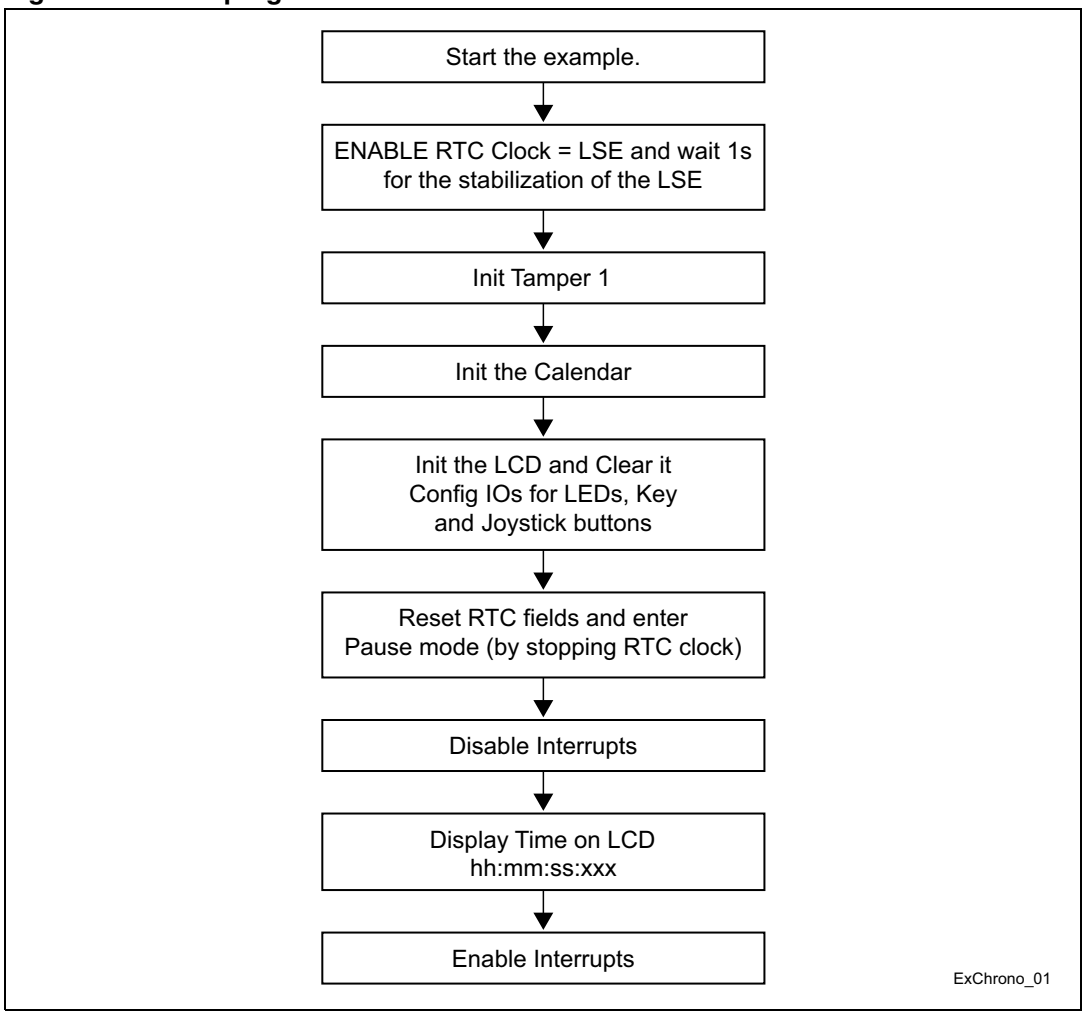


Figure 25. RTC Tamper ISR flowchart

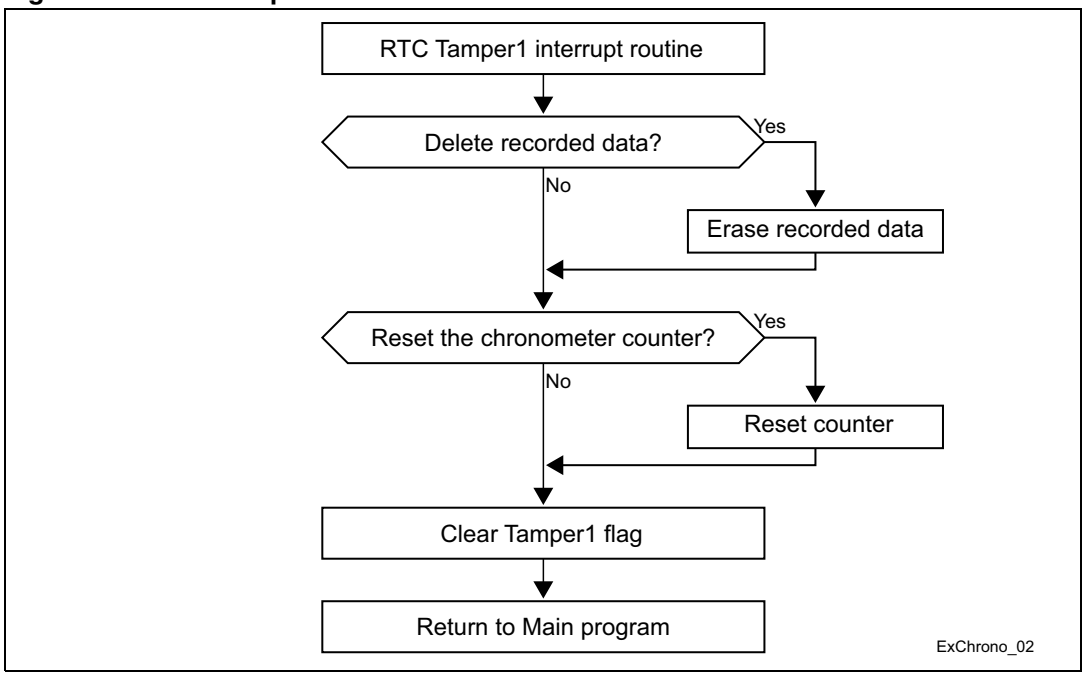


Figure 26. EXTI ISR flowchart for DOWN button

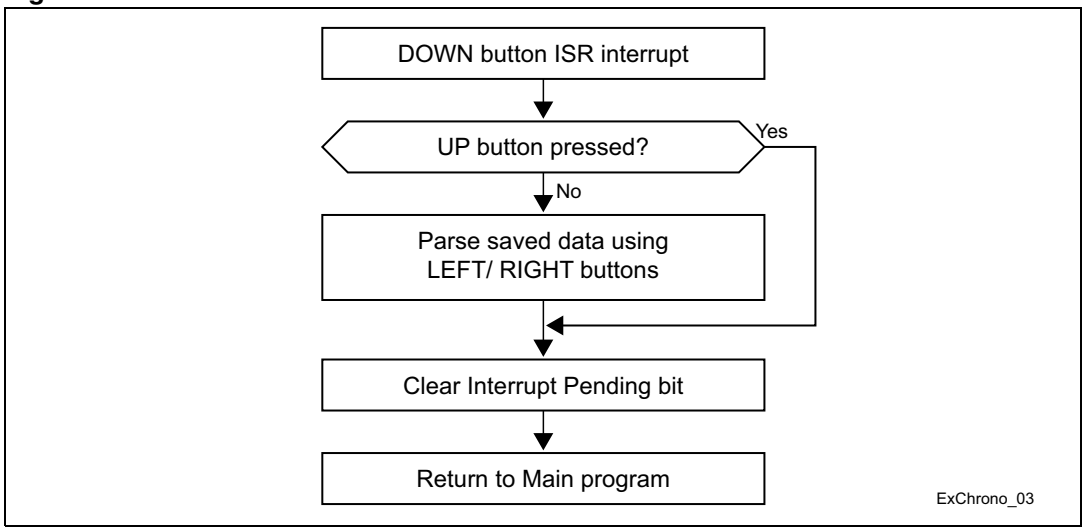
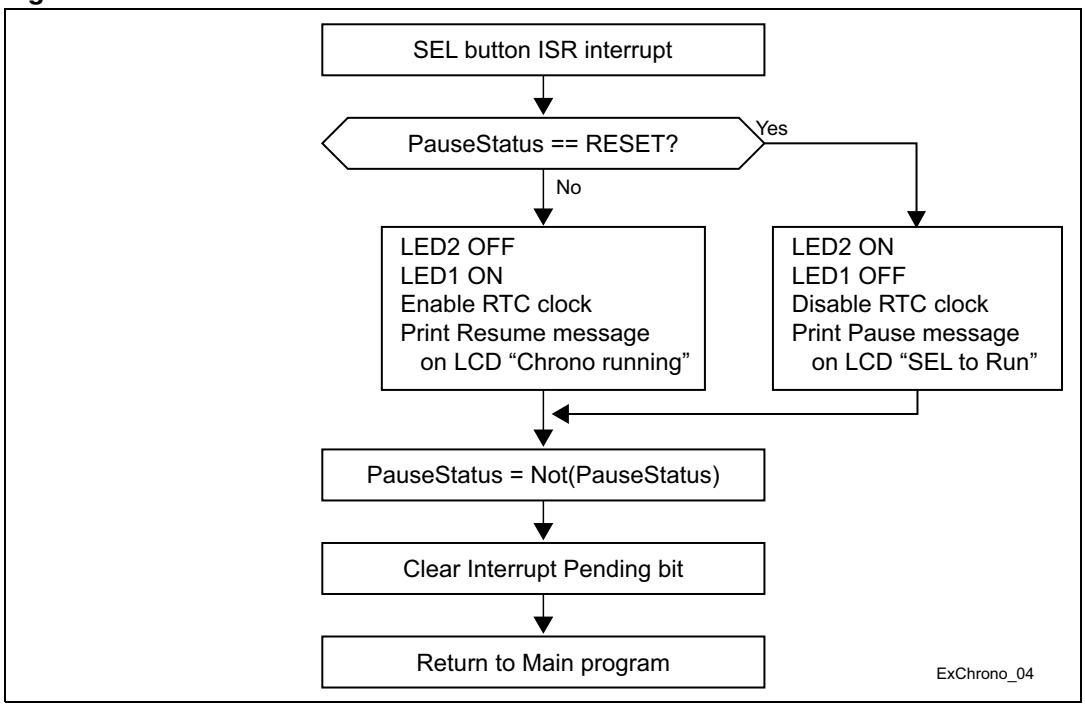
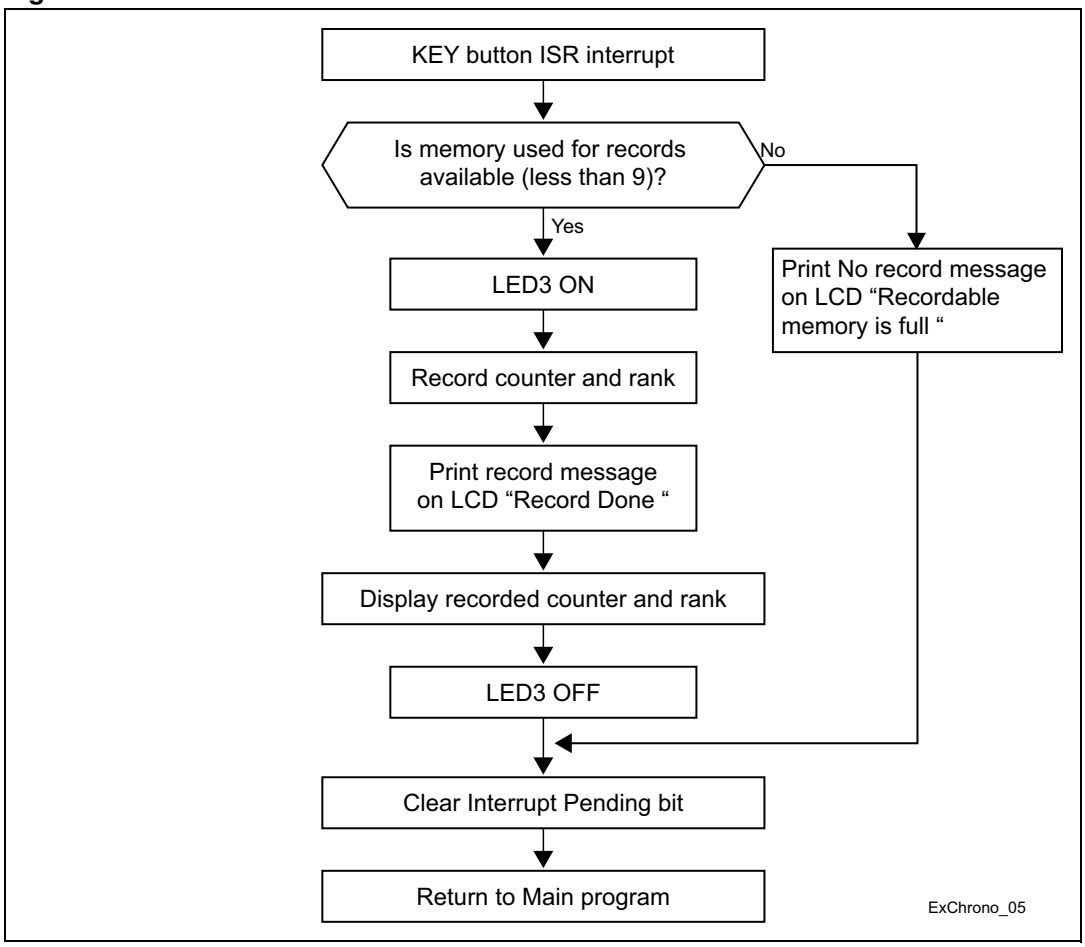


Figure 27. EXTI ISR flowchart for SEL button



Note: PauseStatus is a status variable used to save the last operation: Pause or Running. It is initialized to “RESET” with means running.

Figure 28. EXTI ISR flowchart for KEY button



7 Revision history

Table 20. 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 10: RTC_CALIB output frequency versus clock source*.
08-Jun-2011	3	Updated document for STM8L16x microcontrollers. Updated *RTC calendar adjustment examples on page 7* and *Section 1.5: RTC tamper detection on page 11*. Added *Example 4: Tamper detection on page 32* and *Example 5: Chronometer on page 37*.
21-Sep-2012	4	Added STM8L05xx products. Added *Table 1: Applicable products*.
21-Nov-2012	5	Added STM8AL31xx and STM8AL3Lxx products.

Please Read Carefully:

Information in this document is provided solely in connection with ST products. STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.

All ST products are sold pursuant to ST’s terms and conditions of sale.

Purchasers are solely responsible for the choice, selection and use of the ST products and services described herein, and ST assumes no liability whatsoever relating to the choice, selection or use of the ST products and services described herein.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by ST for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

UNLESS OTHERWISE SET FORTH IN ST’S TERMS AND CONDITIONS OF SALE ST DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE USE AND/OR SALE OF ST PRODUCTS INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

UNLESS EXPRESSLY APPROVED IN WRITING BY TWO AUTHORIZED ST REPRESENTATIVES, ST PRODUCTS ARE NOT RECOMMENDED, AUTHORIZED OR WARRANTED FOR USE IN MILITARY, AIR CRAFT, SPACE, LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS, NOR IN PRODUCTS OR SYSTEMS WHERE FAILURE OR MALFUNCTION MAY RESULT IN PERSONAL INJURY, DEATH, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE. ST PRODUCTS WHICH ARE NOT SPECIFIED AS "AUTOMOTIVE GRADE" MAY ONLY BE USED IN AUTOMOTIVE APPLICATIONS AT USER’S OWN RISK.

Resale of ST products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by ST for the ST product or service described herein and shall not create or extend in any manner whatsoever, any liability of ST.

ST and the ST logo are trademarks or registered trademarks of ST in various countries.

Information in this document supersedes and replaces all information previously supplied.

The ST logo is a registered trademark of STMicroelectronics. All other names are the property of their respective owners.

© 2012 STMicroelectronics - All rights reserved

STMicroelectronics group of companies

Australia - Belgium - Brazil - Canada - China - Czech Republic - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan - Malaysia - Malta - Morocco - Philippines - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States of America

www.st.com