Freescale Semiconductor
Application Note

Document Number:AN4722
Rev 0, 07/2013

Real Time Counter (RTC) on the S12ZVH Family

by: Arturo Inzunza

Contents

1 Introduction

This application note introduces embedded Real Time Counter (RTC) module that comes with the S12ZVH family of microcontrollers. This family of MCUs is aimed toward the automotive instrument cluster segment and is part of the MagniV set of products. The idea behind the MagniV portfolio is to create products that have the most possible integration on-chip and the RTC is one of the modules that is integrated with the MCU.

The main purpose of RTC is to keep accurate timing in easy-to-understand calendar units. This means the time is kept in seconds, minutes, and hours, instead of milliseconds or timer counts. The RTC will provide means of displaying the current time without too much software overhead driving the clock.

The RTC module generates a 1-Hz signal that drives the second, minute, and hour counters. This 1 s signal can be generated from different clock sources and a programmable frequency divider (prescaler). In addition, the RTC module on the S12ZVH includes a compensation mechanism that allows finer adjustments to its 1-Hz clock than a simple divider is capable of. This mechanism also allows compensating for frequency drifts caused by external causes such as crystal aging, temperature changes, or voltage fluctuations. This compensation mechanism provides the means to generate a 1-Hz signal that is more accurate (in terms of deviation) than the

crystal driving the module. The S12ZVH family is able to support two simultaneous crystals: the main oscillator and a 32.768 kHz crystal dedicated for the RTC module.

Besides this compensation mechanism, the RTC also has means for calibration. Since each crystal is slightly different from the next one, a mechanism is necessary to adjust these inherent variations on the crystals. The calibration mechanism is used to measure and compare the generated signal either internally or externally. The Calibration and Compensation mechanisms together ensure that the RTC has the best accuracy possible.

Although the main goal of the RTC module is to keep time, it can be used to generate periodical interrupts for other applications as well. Besides the second, minute, and hour interrupts, the module has a general purpose 4 Hz interrupt. The user can choose to use any of these interrupts for other purposes suitable to the application.

2 RTCCLK sources

The RTC module can be driven from three different clock sources. Figure 1 is an extract of the RTC block diagram available on the S12ZVH reference manual showing the three possible clock sources. The CLKSRC bits are used to select the desired clock source.

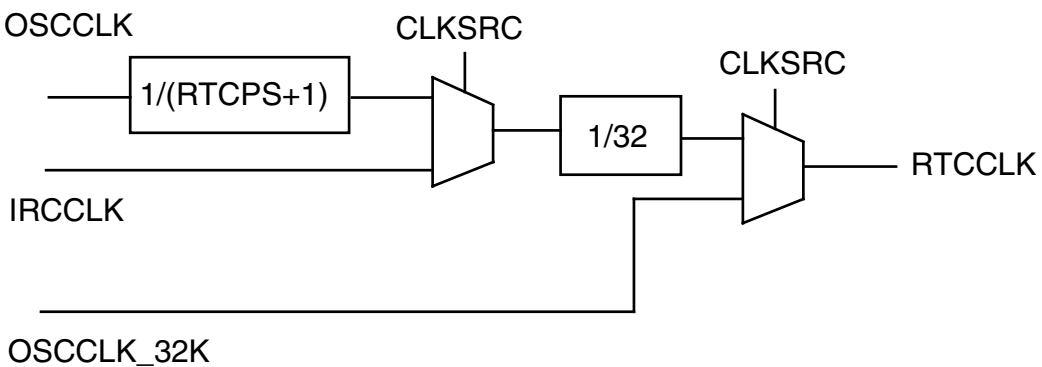

Figure 1. RTC clock sources

Each of the clock signal options need to pass through different dividers to reach a frequency usable by the RTC counter which is 16-bits long. It is recommended that this frequency is around 32 kHz. It is important to consider the properties of each of the available clock sources. In addition, some clocks are turned off during low-power consumption modes, this must be considered as well.

2.1 OSCCLK

The OSCCLK is the main oscillator and is the one that clocks the processor and most of the MCU peripherals. This oscillator needs to be in the range from 4 MHz to 16 MHz and is the fastest clock option for the RTC module. Because of its high speed, it requires a special divider to reduce the frequency to 1 MHz. This divider is configured in the RTCPS bits and can be applied when driving the RTC from OSCCLK. After this divider, the signal goes through a fixed 1/32 divider that brings the input frequency down to a final 31.25 kHz. This is the clock that actually drives the RTC module.

Using the OSCCLK to drive the RTC can be useful when the designer does not wish to add a dedicated oscillator for time keeping. On the other hand, the OSCCLK has the disadvantage of being shut down during low-power modes. This means that the RTC will not be able to track time when the MCU is stopped. If the STOP mode is not used, the OSCCLK can be used without problems.

2.2 OSCCLK_32K

The RTC module can be driven from a dedicated external 32.768 kHz crystal. These 32 kHz oscillators are designed for clocking applications and normally have better accuracy features than larger oscillators. Freescale recommends adding this dedicated oscillator into a design to obtain the best performance and possible accuracy from the RTC module.

The clock selection diagram on Figure 1 shows that the OSCCLK_32K does not require any divider as it can drive the RTC directly. Unlike the OSCCLK, the OSCCLK_32K clock is not turned off during STOP and can be used anytime. This clock option is recommended for applications requiring the best possible accuracy with the possibility of using the low-power consumption modes.

2.3 IRCCLK

The IRCCLK is the internal 1 MHz RC oscillator. This clock does not provide the required accuracy for an actual clocking application but can be used on applications where the accurate current time is tracked by another module and is constantly synchronized with the S12ZVH. The advantages of the IRC clock are that it continues to run in STOP mode and it does not require external components to operate.

On the RTC clock selection mechanism, the IRCCLK is divided only by the 1/32 fixed divider to generate a 31.25 kHz signal. It is not recommended to use the IRCCLK on applications that require good timing accuracy. This clock source has a frequency deviation of ± 1.3% (or 13,000 ppm).

3 RTC configuration

RTC works by generating a 1 Hz signal that feeds the calendar (second, minute, and hour) counters that form the real time information. The second (RTCSECR), minute (RTCMINR), and hour (RTCHRR) registers are configured to work as modulo counters wrapping up at predefined values. The RTCSECR and RTCMINR registers wrap at 59, going back to 0 automatically. The RTCHRR register wraps at 23, going back to 0 on the next counter increase. This predefined behavior effectively reduces the software required to calculate the current time.

3.1 RTCMOD

The RTC module uses two registers to generate 1 Hz signal, feeding the rest of counters on the module: the RTCMOD and RTCCNT. The RTCCNT is a simple 16-bit up counter that counts at the rate of the RTCCLK. The RTCCLK depends on the clock source selected previously. The RTCMOD register acts, as its name states, as a modulo register that sets the value at which the RTCCNT must wrap generating the signal that feeds the calendar counters. Figure 2 shows the workflow followed to generate the 1 Hz signal without enabling the compensation mechanism.

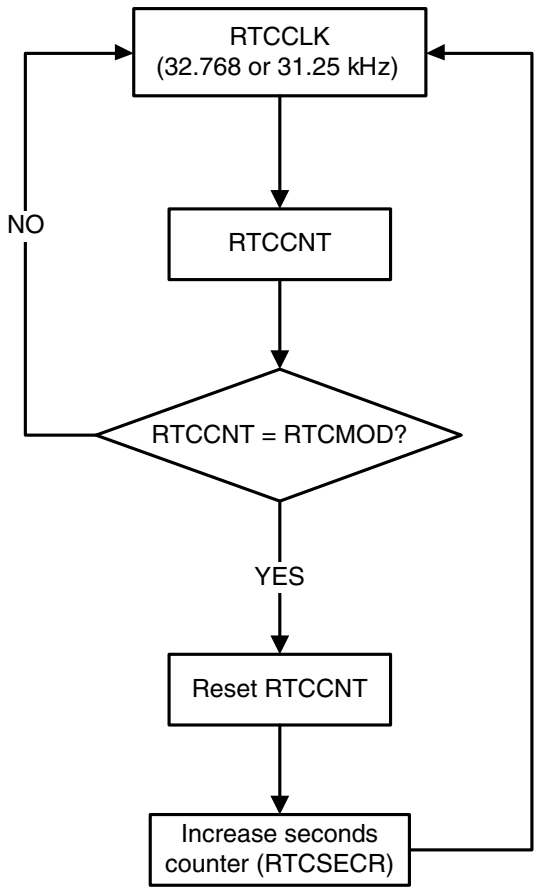

Figure 2. 1 Hz generation flow without enabling compensation

This configuration ensures that there is a toggle on 1 Hz signal on each RTCMOD count. The RTCMOD register must be set up accordingly to the RTCCLK selected, for example:

- When using the OSCCLK_32K, the RTCCLK is 32.768 kHz. The RTCMOD register must be set up to 32,768, which will cause the RTC seconds counter (RTCSECR) to increase by one each 32,768 oscillator cycles.
- When using the OSCCLK or the IRCCLK, the RTCCLK is 31.25 kHz. In this case, the RTCMOD is set to 31,250.

The user may need to adjust the RTCMOD register to match the actual frequency of the RTCCLK, which might have certain frequency drift. The calibration mechanism can help further tune the RTCMOD value and will be covered later in this application note.

4 Compensation

The RTCMOD register allows for a rough tune of the 1 Hz signal because it can only accept integer values and trunks the fractional values. For example, if the RTCCLK was oscillating at 32,768.46 Hz, the RTCMOD would need to be set to 32,768 and the user would see an accumulated error of 0.46 counts per second. This means a 14.03 ppm drift or 36.38 seconds per month. This deviation considers only the difference between the RTCCLK and the time on the module calendar counters.

The compensation mechanism can consider the non-integer part of the RTCCLK frequency (0.46 Hz in the mentioned example), thereby increasing the RTC module's overall accuracy. The compensation module accumulates these fractional errors for a predefined period of time and then compensates for the accumulated integer counts. Figure 3 represents the flow diagram of the compensation algorithm.

The Compensation Cycle Selection (CCS) is the periodicity at which the RTC module executes the compensation algorithm. The user can select between four predefined periods: 5, 15, 30, and 60 seconds. Each time the Compensation Cycle needs to be executed, the RTC module waits for an additional Q counts to compensate for the accumulated errors.

Following on the previous example, where RTCCLK oscillates at 32768.46 Hz, we will assume the user selected to compensate once each by 5 second. In those 5 second, the RTCCLK accumulated an error of 2.3 counts skipped (0.46 counts multiplied by 5 second). In this case, the user would configure the Q bits as 2. In every 5 second, the RTC module would wait for two additional counts on the last second, accounting for part of the accumulated missing count fractions. Using this configuration would reduce the RTC error to 0.3 counts in each 5 second, or 0.06 count per second. This is same as drift of 1.83 ppm or 4.74 second per month between the RTCCLK and the module's time.

If the CCS is larger, the minimum compensation unit gets smaller. This means that a 60 second CCS allows for finer compensation than the 5 second CCS. Continuing on the previous example, if the user selects a 60 second CSS, the RTC would see an accumulated error of 27.6 counts (0.46 counts multiplied by 60 second). In this case, Q would be set to 28 (the closest integer value), and the error would be of 0.4 counts in each 60 second. This error is the same as 0.0066 counts per second, which is 0.203 ppm of deviation or approximately 0.52 second per month.

NOTE

The values discussed in this section only account for the accuracy lost on the processing of the RTCCLK to feed the calendar counters. It does not consider RTCCLK precision or the variance this clock source might have due to external factors.

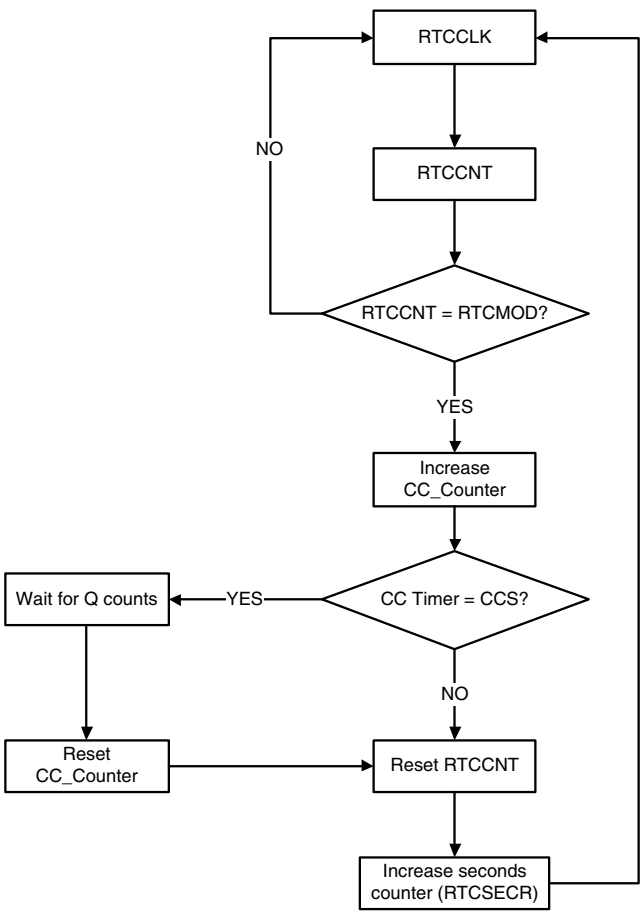

Figure 3. 1 Hz generation flow with compensation enabled

Different CCS periods allow for finer compensations and less accumulated errors. Table 1 summarizes each CCS period and its minimum adjustment possible.

Table 1. CCS minimum compensation steps

CCS	Minimum adjustment	Adjustment precision (1 s = 32,768 counts)
5 s	1/5 count	6.1 µs
15 s	1/15 count	2.03 µs
30 s	1/30 count	1.02 µs
60 s	1/60 count	508.63 ns

The best compensation possible is achieved with the 60 seconds CCS and allows compensating for 1/60 of count. Considering that the RTCCLK frequency is nominally 32.768 kHz, 1/60 equals to a compensation fraction as small as 508 ns or 0.508 ppm.

5 Calibration

The compensation mechanism allows minimizing the error in generating 1 Hz signal from the RTCCLK when its frequency is known. But even when using the same crystal manufacturer and model for two boards, each crystal has inherent differences that add to the timing errors. A calibration method allows adjusting the compensation mechanism for each case. The RTC provides two options to calibrate the compensation mechanism to the RTCCLK frequency, off-chip, and on-chip.

5.1 Off-Chip calibration

The off-chip calibration is intended for the user to externally read the RTCCLK value and manually calibrate the RTC module. Once this is done, the user can set CALCLK to output the 1 Hz signal so that the desired frequency output can be confirmed. Besides the 32 KHz oscillator pins, RTC has another external pin for the calibration mechanism called RTC_CAL. The signal transmitted on the RTC_CAL pin is called CALCLK, and can output either the RTCCLK or the 1 Hz signal. By using external measuring equipment the user can measure the actual RTCCLK frequency and adjust the RTCMOD, CCS, and Q registers to get a 1 Hz signal.

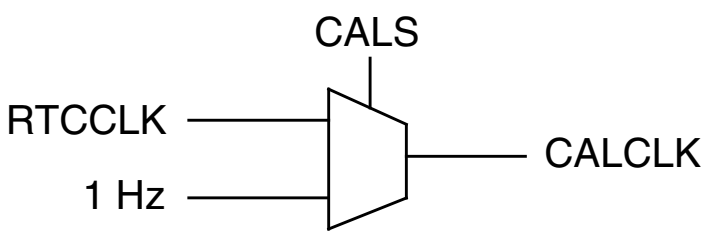

Figure 4. CALCLK selection diagram

As shown in Figure 4, the CALCLK signal source can be selected by changing the CALS bit-field. The main advantage of the off-chip calibration is that the user does not require much code to perform the calibration. The main disadvantage is that it involves much manual work in measuring and adjusting parameters for each production device.

NOTE

By default, the CALCLK signal is not routed to the RTC_CAL pin because this pin is shared with other peripherals. The user needs to enable the CALCLK output to the RTC_CAL pin via the PIMMISC register.

5.2 On-Chip calibration

The on-chip calibration provides opportunity to automate the calibration process via software. The basic principle of the self-calibration mechanism is that the MCU is able to detect the difference between the current RTC 1 Hz signal (generated by the RTCCLK and the compensation mechanism) and a 1 Hz clock reference (provided externally and connected to the RTC_CAL pin). The MCU can then use these differences to self-adjust the compensation registers and reduce this error to a minimum. The great advantage of the on-chip calibration is that at factory production, the user needs to provide only a precise 1 Hz signal and each device will attempt to synchronize to it automatically. The manual intervention of the user is reduced to a minimum.

The RTC_CAL pin is shared with the Timer 1, channel 1 (TIM1CH1), which allows using this pin to monitor the externally provided precise 1 Hz clock via the input capture mechanism. The CALCLK signal can be internally routed to the Timer 1, channel 0 (TIM1CH0) via the MODRR2 register. Figure 5 shows the timer internal connections used for the calibration.

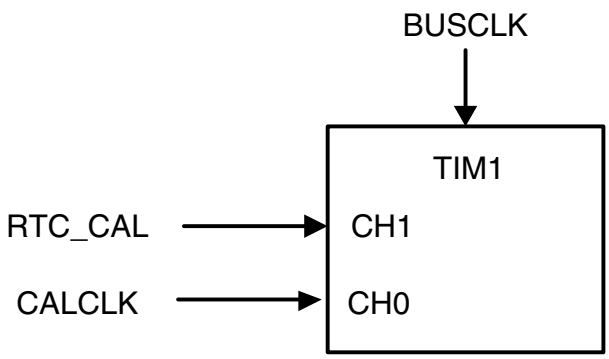

Figure 5. On-Chip configuration of the Timer module

CALCLK should be configured to output 1 Hz signal (see Figure 4).

The timer module will be used to determine the duration of each of the two pulses to detect the current drift of the RTC 1 Hz signal versus the ideal one. Since both signals are being measured by the same clock, BUSCLK in this case, the measurement errors are greatly reduced. It is the user's responsibility to select the appropriate timer frequency (BUSCLK after the timer prescalers) for the measurements. Faster timer clock frequencies will provide better measurement resolutions than slower frequencies, but will also generate larger count numbers to work with and could generate timer overflows, complicating the software. When running the calibration routine, it is recommended that the BUSCLK source is the OSCCLK directly, and not the PLL. The PLL has inherent variances that will add to the measurement errors. Using the most precise clock source available will reduce variance in the measurements.

5.2.1 On-Chip calibration example

Consider a setup that has an external main oscillator of 4 MHz, meaning the BUSCLK is 2 MHz. The timer prescaler is set so that the timer frequency is slightly less than 60 kHz. This will ensure measurement of a second to be less than 65536, which fits a 16-bit variable. Using the "precision timer" feature of the timer module allows reaching a frequency of 58.823 kHz (BUSCLK/34).

The RTC module is configured considering the ideal 32.768 kHz of the external dedicated oscillator, therefore RTCMOD is set to 32,768 and the compensation is disabled. As an example we assume the following measurements are obtained:

A. CALCLK (internal RTC 1 Hz): 60,385
B. RTC_CAL (external 1 Hz): 58,903

$$\frac{B-2}{A+2} RTC\_CAL < CALCLK < \frac{B+2}{A-2} RTC\_CAL \approx \frac{B}{A} RTC\_CAL$$

Equation 1. On chip frequency calculation

Where:

- A and B are the counts of the CALCLK and RTC_CAL respectively
- RTC_CAL and CALCLK are the exact frequencies in Hz (~ 1 Hz)

The relationship between RTC_CAL and CALCLK is defined by the fraction B/A, where B and A are the timer measurements for each of the clock sources. The CALCLK frequency is generated by the RTC according to the following equation. This equation considers that the compensation is disabled

$$CALCLK = \frac{RTCCLK}{RTCMOD_0}$$

Equation 2. On chip frequency calculation

.

Where:

- RTCCLK is the clock driving the RTC
- RTCMOD0 is the value of the RTCMOD register during the measurements. Assumed 32768.

From these equations it can be obtained that:

$$\frac{RTCCLK}{RTCMOD_0} = \frac{B}{A} RTC\_CAL$$

Equation 3. Replacing equation 1 in equation 2

$$RTCCLK = \frac{B}{A} RTC_{CAL} * RTCMOD_0 = \frac{58903}{60385} (1\ Hz) * 32768 = 31963.79\ Hz$$

Equation 4. Solving for RTCCLK

NOTE

Note that the RTC_CAL frequency was assumed as 1 Hz since it is supposed to be a precise externally provided signal. The user must replace this value with the actual RTC_CAL frequency used for calibration.

The RTCCLK frequency is the actual value that the RTC module uses to generate 1 Hz signal. The auto-calibration software can take this calculated RTCCLK frequency to adapt the compensation algorithm accordingly.

6 Temperature compensation

Crystals, as most electronic components, are affected by external agents as temperature, aging, voltage changes, etc. The calibration process helps baseline the RTC by setting a correct compensation value. But temperature and aging will require this compensation value to be adjusted accordingly to keep up with the crystal de-rating.

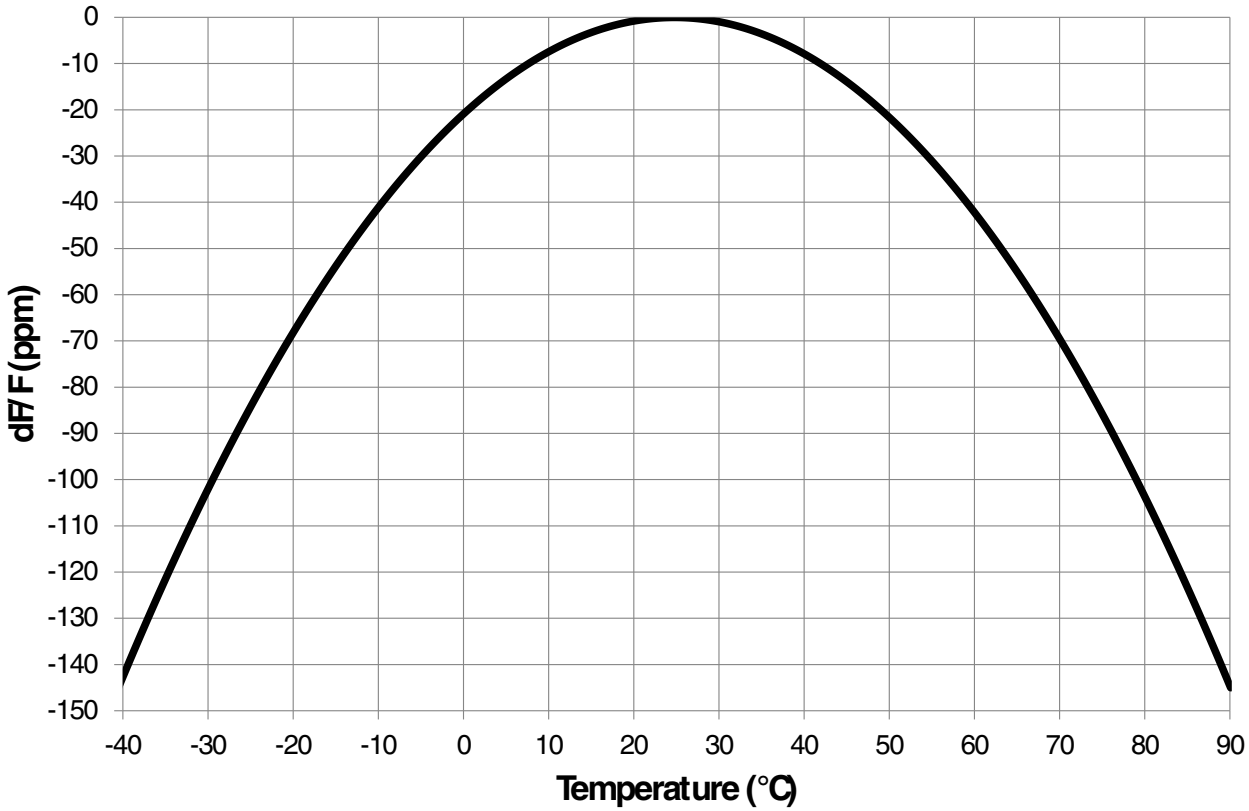

Figure 6. Oscillator accuracy loss versus temperature graph example

Crystal vendors characterize the behavior of their crystals over time and on different temperatures. These values can be used for on-the-run compensation. Figure 6 is an example of a graph provided by a crystal vendor.

To compensate for these frequency changes over temperature, the user will need to add a temperature sensor on the design. This sensor must be placed as close as possible to the 32 kHz crystal so that the temperature measurements really reflect the actual crystal environment. The temperature compensation should follow these steps:

1. Measure the crystal temperature.
2. Look for the temperature compensation value on the crystal lookup table.
3. Load the adjusted compensation values to the RTC.

For this approach to be useful, the user must generate a lookup table that relates the crystal temperature with the relative oscillator precision loss. Remember that –10 ppm is the same as a 0.001% frequency reduction. Table 2 is an example of this adjustment table.

Table 2. Temperature compensation table example

Temp	%Loss	Counts Less
–40	0.014340	470
–30	0.010258	336
–20	0.006855	225
–10	0.004133	135
0	0.002090	68
10	0.000728	24

Table continues on the next page...

Table 2. Temperature compensation table example (continued)

Temp	%Loss	Counts Less
20	0.000046	2
25	0.000000	0
30	0.000045	2
40	0.000721	24
50	0.002078	68
60	0.004116	135
70	0.006834	225
80	0.010231	336
90	0.014309	470

The oscillator frequency reduction (in counts) is calculated by multiplying the RTCCLK nominal frequency multiplied by the loss percentage. This value can be calculated on-the-run using the feedback of a temperature sensor. The problem is that this option requires floating point calculations that might not be desired. An alternative is to consider the RTCCLK nominal frequency (32,768 Hz) as a base, and calculate beforehand the counts that the compensation algorithm needs to reduce to the original RTCMOD.

The temperature is the external factor that affects crystals the most. To achieve better results, the RTC should be frequently compensated against temperature.

7 Conclusions

The RTC module is capable of compensating for changes on the RTCCLK frequency as small as 508.6 ns or 0.508 ppm. But several other factors add to the actual 1 Hz signal accuracy. The RTC oscillator accuracy, calibration signal metering, temperature sensing, and rounding on the temperature lookup table add to the total error on the output signal. Following the steps on this application note it is possible to achieve an overall 10 ppm precision, which is a 0.864 second deviation per day.

8 References

Please consult the updated device's reference manual, reference design, and other information at: http://www.freescale.com/S12ZVH

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number AN4722
Revision 0, 07/2013