Freescale Semiconductor
应用笔记

Document Number: AN4722
Rev 0, 07/2013

S12ZVH 系列上的实时计数器(RTC)

作者: Arturo Inzunza

内容

1 简介

本应用笔记将介绍 S12ZVH 系列微控制器中的嵌入式实时计数器(RTC)模块。该系列 MCU 的目标为汽车仪表盘市场，是 MagniV 产品集的一部分。MagniV 产品组合的设计理念是创造具有最高片上集成的产品，而 RTC 就是集成在 MCU 上的一个模块。

RTC 的主要用途是以浅显易懂的日历方式保持精确计时。这意味着时间以秒、分钟和小时保存，而不是毫秒或定时器计数。RTC 将提供显示当前时间的方法，而无需太多软件开销来驱动时钟。

RTC 模块产生一个 1-Hz 信号来驱动秒、分钟和小时计数器。该 1s 信号可由不同的时钟源和一个可编程分频器 (预分频器) 产生。此外，S12ZVH 上的 RTC 模块包含一个补偿机制，允许对其自身 1-Hz 时钟进行调节，比简单的分频器更精细。该机制还能用于补偿由外部原因 (如晶振老化、温度变化或电压波动) 导致的频率漂移。由于该补偿机制，模块产生的 1-Hz 信号比驱动该模块的晶振更为精确 (就偏差而言)。S12ZVH 系列可以同时支持两种晶振：主振荡器和专门用于 RTC 模块的 32.768 kHz 晶振。

除了补偿机制，RTC 还具有校准功能。由于每个晶振之间都有细微差别，因此需要一种机制来调节这些晶振上的固有差异。校准机制用于对所产生的信号进行内部或外部的测量和比较。校准和补偿机制共同确保 RTC 达到可能具有的最佳精度。

虽然 RTC 模块的主要目的是计时，但还可将其用于为其他应用产生周期性中断。除了秒、分钟和小时中断，该模块还能产生通用的 4 Hz 中断。用户可根据应用使用合适的中断，以用于其他目的。

2 RTCCLK 源

RTC 模块可通过 3 种不同的时钟源进行驱动。图 1 是来自 S12ZVH 参考手册中的 RTC 结构框图，显示了 3 种可用时钟源。CLKSRC 位用于选择所需的时钟源。

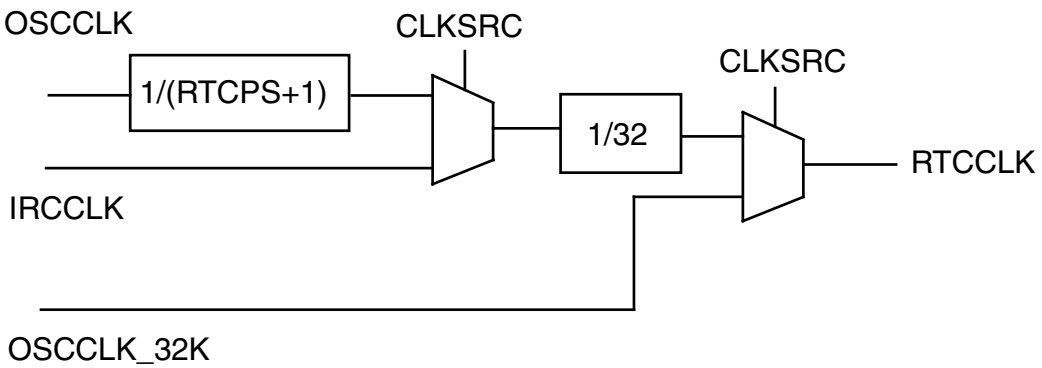

图 1. RTC 时钟源

每个时钟信号选项需要通过不同的分频器以得到 RTC 计数器(16 位长)可用的频率。建议该频率约为 32 kHz。考虑每种可用时钟源的属性很重要。此外，一些时钟在低功耗模式下会关闭，也必须考虑这个问题。

2.1 OSCCLK

OSCCLK 是主振荡器，为处理器和大多数 MCU 外设提供时钟。该振荡器的频率范围要求为 4 MHz 到 16 MHz，是用于 RTC 模块的最快时钟选项。由于速度快，因此需要特殊分频器以将频率降低至 1 MHz。该分频器由 RTCPS 位进行配置，当选择由 OSCCLK 驱动 RTC 时，该分频器生效。经过该分频器后，信号会通过一个固定的 1/32 分频器，由它将输入频率降低至最终的 31.25 kHz。这正是驱动 RTC 模块的时钟。

当设计人员不希望添加专用振荡器来计时的话，使用 OSCCLK 驱动 RTC 将变得非常有用。不过，OSCCLK 的劣势是在低功耗模式下会关闭。这意味着在 MCU 停止时 RTC 不能跟踪时间。如果不使用 STOP 模式，那么使用 OSCCLK 将不存在任何问题。

2.2 OSCCLK_32K

RTC 模块可以由专用的外部 32.768 kHz 晶振驱动。这些 32 kHz 振荡器专为时钟应用而设计，通常比更大型的振荡器的精度更高。飞思卡尔建议您在设计中添加此专用振荡器，这将获得 RTC 模块的最佳性能和可能的最高精度。

图 1 上的时钟选择图显示了 OSCCLK_32K 无需任何分频器便能直接驱动 RTC。与 OSCCLK 不同，OSCCLK_32K 时钟在 STOP 模式下不会关闭，随时可用。当应用需要最佳可能精度，以及可能会使用低功耗模式时，建议使用该时钟选项。

2.3 IRCCLK

IRCCLK 是内部 1 MHz RC 振荡器。该时钟不能为实际时钟应用提供所需的精度，但是当应用中的精确当前时间由另一模块跟踪，并持续与 S12ZVH 保持同步，则该时钟也可以使用。IRC 时钟的优势是在 STOP 模式下可以继续运行，并且无需外部器件。

在 RTC 时钟选择机制中，IRCCLK 仅需通过 1/32 固定分频器进行分频，便能产生 31.25 kHz 信号。不建议在需要良好计时精度的应用中使用 IRCCLK。该时钟源的频率偏差为± 1.3%（即 13,000 ppm）。

3 RTC 配置

RTC 产生 1 Hz 信号给到日历（秒、分钟和小时）计数器以形成实时信息。秒(RTCSECR)、分钟(RTCMINR)和小时(RTCHRR)寄存器作为模数计数器进行配置，在预定义值处返回。RTCSECR 和 RTCMINR 寄存器在 59 处返回，自动变为 0。RTCHRR 寄存器在 23 处返回，在下一个计数器递增时返回为 0。该预定义行为有效地降低了计算当前时间所需的软件开销。

3.1 RTCMOD

RTC 模块使用两个寄存器来产生 1 Hz 信号，用于模块上的其他计数器：RTCMOD 和 RTCCNT。RTCCNT 是一个简单的 16 位计数器，以 RTCCLK 速率进行计数。RTCCLK 速率取决于之前选择的时钟源。RTCMOD 寄存器如其名字作为模数寄存器使用，设置 RTCCNT 必须返回的值，以产生给到日历计数器的信号。图 2 显示了补偿机制未使能情况下产生 1 Hz 信号的流程。

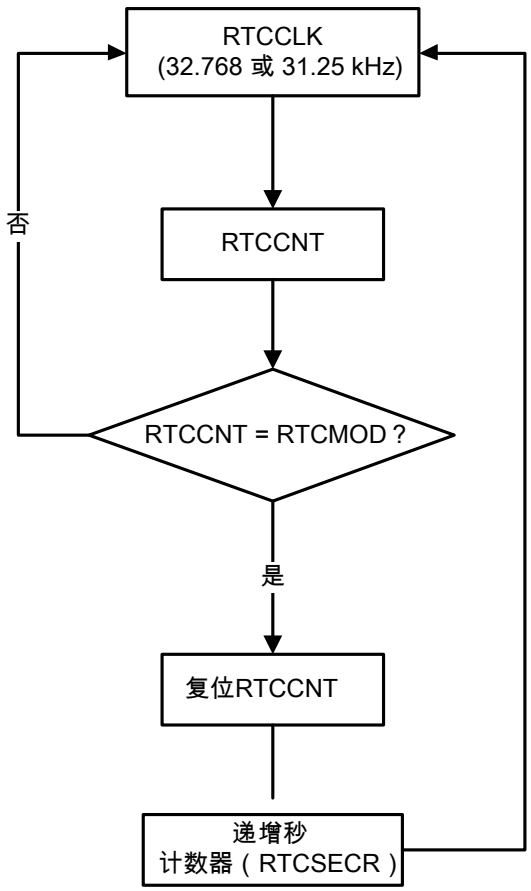

图 2. 补偿机制未使能情况下的 1 Hz 产生流程

该配置确保每次 RTCCNT 计数至 RTCMOD 时，在 1Hz 信号上会产生翻转。RTCMOD 寄存器必须根据选择的 RTCCLK 进行设置，例如:

- 当使用 OSCCLK_32K 时，RTCCLK 为 32.768 kHz。RTCMOD 寄存器必须设置为 32,768，从而使 RTC 秒计数器(RTCSECR)在每 32,768 次振荡器周期后递增 1。
- 当使用 OSCCLK 或 IRCCLK 时，RTCCLK 为 31.25 kHz。此时，RTCMOD 设置为 31,250。

用户可能需要调节 RTCMOD 寄存器使其与 RTCCLK 的实际频率相匹配，因为 RTCCLK 可能会存在一定的频率漂移。校准机制可以进一步微调 RTCMOD 的值，这将在本应用笔记的后文中进行描述。

4 补偿

RTCMOD 寄存器允许 1 Hz 信号的粗调，因为该寄存器只接受整数值，舍去小数值。例如，如果 RTCCLK 振荡在 32,768.46 Hz，RTCMOD 需设置为 32,768，用户将看到每秒 0.46 计数的累计误差。这意味着存在 14.03 ppm 的偏移或每月 36.38 秒的误差。该偏差仅考虑了 RTCCLK 和模块日历计数器之间的差异。

补偿机制可以考虑非整数部分的 RTCCLK 频率 (示例中提到的 0.46 Hz)，从而提升 RTC 模块的整体精度。补偿模块在一段预定义的时间周期内累计这些小数误差，然后补偿到累计的整数计数中去。图 3 为补偿算法的流程图。

补偿周期选择(CCS)是 RTC 模块执行补偿算法的周期。用户可在四种预定义的周期中进行选择: 5、15、30 和 60 秒。每次都需要执行补偿周期，RTC 模块等待额外的 Q 个计数来补偿累计误差。

接着之前的示例，其中 RTCCLK 振荡频率为 32768.46 Hz，我们假设用户选择每 5 秒补偿一次。在这 5 秒中，RTCCLK 累计了 2.3 个计数的误差（0.46 计数乘以 5 秒）。此时，用户将配置 Q 位为 2。每隔 5 秒，RTC 模块会在最后 1 秒时等待 2 个额外计数，以计入部分累计损失的计数小数。采用该配置可降低 RTC 误差至每 5 秒 0.3 个计数，即每秒 0.06 个计数。这也相当于在 RTCCLK 和模块时间之间存在 1.83 ppm 的偏移或每月 4.74 秒的误差。

如果 CCS 较大，则最小补偿单位越小。这意味着 60 秒的 CCS 比 5 秒的 CCS 提供更精细的补偿。继续之前的示例，如果用户选择 60 秒 CSS，则 RTC 累计误差 27.6 个计数（0.46 计数乘以 60 秒）。此时，Q 设为 28（最接近的整数值），误差将为每 60 秒 0.4 个计数。该误差相当于每秒 0.0066 个计数，为 0.203 ppm 的偏移或每月约 0.52 秒的误差。

注

本章讨论的数值仅计算了 RTCCLK 给到日历计数器过程中的精度损失。并未考虑 RTCCLK 精度或由于外部因素导致的该时钟源的变化。

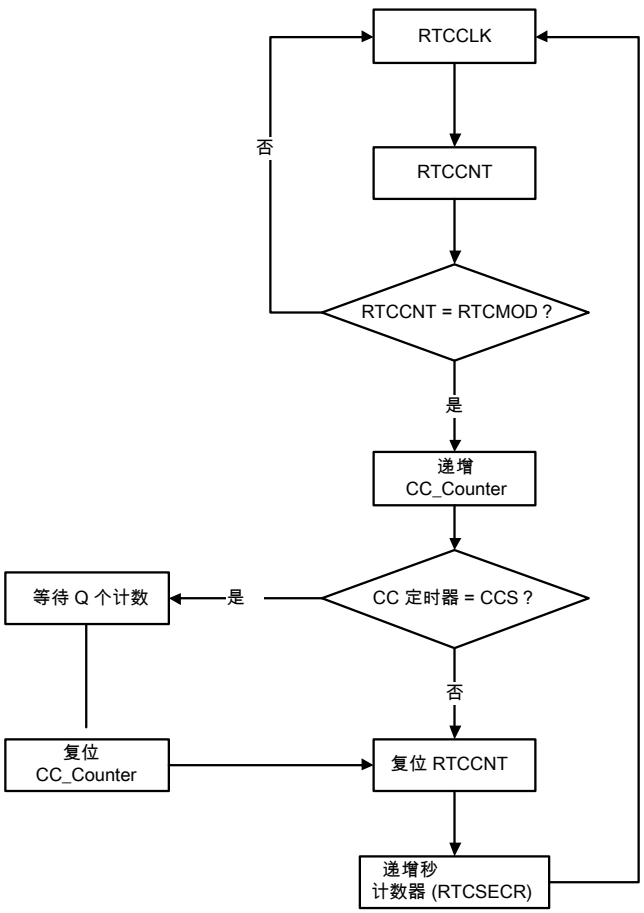

图 3. 补偿机制使能情况下的 1 Hz 产生流程

不同的 CCS 周期提供更精准的补偿以及更低的累计误差。表 1 汇总了每种 CCS 周期及其可能的最小调整值。

表 1. CCS 最小补偿步长

CCS	最小调整值	调整精度（1 s = 32,768 个计数）
5 s	1/5 个计数	6.1 µs
15 s	1/15 个计数	2.03 µs
30 s	1/30 个计数	1.02 µs
60 s	1/60 个计数	508.63 ns

可达到的最佳补偿为 60 秒 CCS，允许补偿 1/60 个计数。考虑到 RTCCLK 频率通常为 32.768 kHz，1/60 相当于补偿小数低至 508 ns 或 0.508 ppm。

5 校准

补偿机制在频率已知的情况下，可以最小化由 RTCCLK 产生的 1 Hz 信号的误差。但是，即使两块电路板使用了相同制造商和型号的晶振，每个晶振仍具有固有差异，从而增加计时误差。校准机制为每种情况调整补偿机制。提供两种选项用于校准针对 RTCCLK 频率的补偿机制：片外和片上。

5.1 片外校准

片外校准的目的是使用户可以从外部读取 RTCCLK 的值，并手动校准 RTC 模块。一旦完成，用户可以设置 CALCLK 输出 1 Hz 信号，从而可以确认所需的频率输出。除了 32 KHz 振荡器引脚，RTC 还具有一个用于校准机制的外部引脚 RTC_CAL。在 RTC_CAL 引脚上传输的信号称为 CALCLK，可以输出 RTCCLK 或 1 Hz 信号。通过使用外部测量设备，用户可以测量实际的 RTCCLK 频率，并调整 RTCMOD、CCS 和 Q 寄存器以获取 1 Hz 信号。

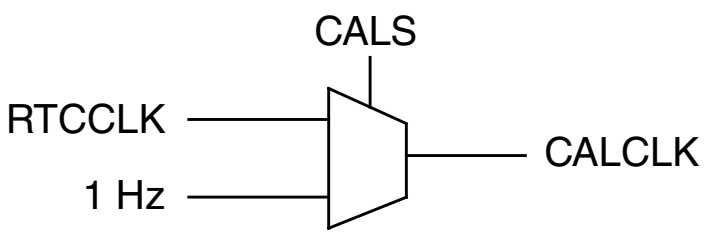

图 4. CALCLK 选择图

如图 4 所示，CALCLK 信号源可通过改变 CALS 位域进行选择。片外校准的主要优势是用户无需编写太多用于执行校准的代码。主要劣势是对于每个量产器件，需要进行太多的手动测量和参数调整工作。

注

默认情况下，CALCLK 信号没有引至 RTC_CAL 引脚，因为该引脚与其他外设共用。用户需要通过 PIMMISC 寄存器使能 CALCLK 输出到 RTC_CAL 引脚。

5.2 片上校准

片上校准通过软件提供自动的校准过程。自校准机制的基本原理是 MCU 能够检测出当前 RTC 1 Hz 信号（由 RTCCLK 和补偿机制产生）和 1 Hz 时钟参考（由外部提供并连接到 RTC_CAL 引脚）之间的差异。然后，MCU 可以利用这些差异对补偿寄存器进行自调节并将误差降为最低。片上校准的一大优势是工厂量产，用户只需提供一个精确的 1 Hz 信号，然后每个器件会自动尝试与之同步。从而使人工干预降至最低。

RTC_CAL 引脚与 Timer 1 的通道 1 (TIM1CH1)共用，从而允许使用该引脚通过输入捕捉机制来监视外部提供的精确 1 Hz 时钟。 CALCLK 信号可通过 MODRR2 寄存器从内部引线到 Timer 1 的通道 0 (TIM1CH0)。图 5 显示了用于校准的定时器内部连接。

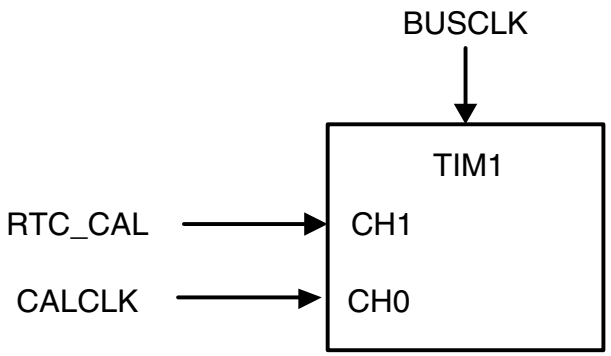

图 5. 定时器模块的片上配置

CALCLK 应配置为输出 1 Hz 信号（见图 4 ）。

定时器模块将用于确定两个脉冲的时长，以检测当前 RTC 1Hz 信号与理想信号之间的偏差。由于两个信号均由同一个时钟（这里是 BUSCLK）测量，因此测量误差大大降低。用户应选择合适的定时器频率（经定时器预分频器后的 BUSCLK）进行测量。较快的定时器时钟频率与较慢的频率相比将提供更好的测量分辨率，但是也会产生更多需要处理的计数数量，可能产生定时器溢出及使软件更复杂。当运行校准程序时，建议将 OSCCLK 直接设为 BUSCLK 源，而不是 PLL。PLL 的固有差异会增加测量误差。尽可能使用最精确的时钟源可降低测量中的差异。

5.2.1 片上校准示例

假设一个具有 4 MHz 外部主振荡器的设置，即 BUSCLK 为 2 MHz。设置定时器预分频器，使定时器频率略低于 60 kHz。这将确保一秒的测量小于 65536，从而符合 16 位变量。利用定时器模块的“精确定时器”特性可以达到 58.823 kHz (BUSCLK/34) 的频率。

RTC 模块配置为理想的 32.768 kHz 外部专用振荡器，因此 RTCMOD 设置为 32,768 且禁用补偿。作为示例，我们假设已获取以下测量数据：

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

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

等式 1. 片上频率计算

其中：

- A 和 B 分别是 CALCLK 和 RTC_CAL 的计数
- RTC_CAL 和 CALCLK 是实际的频率(~ 1 Hz)

RTC_CAL 和 CALCLK 之间的关系由分数 B/A 决定，其中 B 和 A 是每个时钟源的定时器测量值。根据以下公式 CALCLK 频率由 RTC 产生。该公式禁止了补偿

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

等式 2. 片上频率计算

.

其中：

- RTCCLK 是驱动 RTC 的时钟
- RTCMOD0 是测量过程中 RTCMOD 的值。假设为 32768。

从这些公式可获得:

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

等式 3. 在公式 2 中带入公式 1

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

等式 4. 解得 RTCCLK

注

请注意，RTC_CAL 频率假设为 1 Hz，因为其应为外部提供的精确信号。校准时，用户必须用实际的 RTC_CAL 频率替换该值。

RTCCLK 频率是 RTC 模块用来产生 1 Hz 信号的实际值。自校准软件可以根据该计算得到的 RTCCLK 频率来调整补偿算法。

6 温度补偿

与大多数电子元件一样，晶振会受外部因素影响，如温度、老化和电压变化等。校准流程通过设定合适的补偿值来帮助 RTC 设置基准。但是，温度和老化需要 RTC 相应的调节该补偿值，以匹配上晶振的降级。

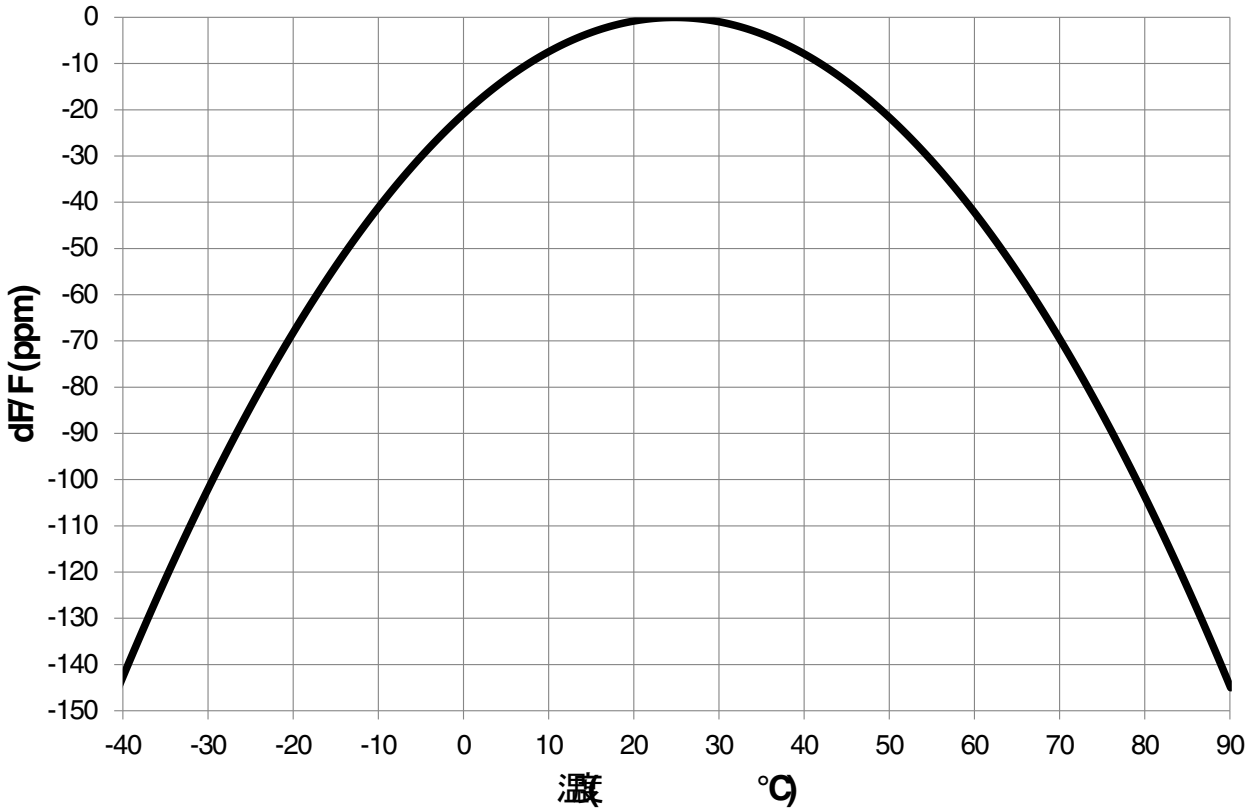

图 6. 振荡器精度损失与温度图示

晶振供应商定义了其晶振在时间和不同温度下的特性。这些值可用于实时补偿。图 6 是一个晶振供应商提供的图示。

为了补偿这些由于温度产生的频率变化，用户将需要在设计中添加一个温度传感器。该传感器必须尽可能地靠近 32 kHz 晶振放置，从而测量到的温度才能真正反映实际的晶振环境。温度补偿应遵循以下步骤：

1. 测量晶振温度。
2. 在晶振查找表中查找温度补偿值。
3. 加载调整后的补偿值到 RTC。

为了使该方法有效，用户必须生成一个查找表，将晶振温度与振荡器精度损失相关联。请记住，–10 ppm 等同于 0.001%频率下降。表 2 是一个调整表示例。

表 2. 温度补偿表示例

温度	损失(%)	减少的计数
–40	0.014340	470
–30	0.010258	336
–20	0.006855	225
–10	0.004133	135
0	0.002090	68
10	0.000728	24
20	0.000046	2

下一页继续介绍此表...

表 2. 温度补偿表示例 (继续)

温度	损失(%)	减少的计数
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

振荡器频率减少值（以计数为单位）可通过将 RTCCLK 标称频率乘以损失百分比计算得到。该值可利用温度传感器的反馈实时计算。问题是，该选项需要使用我们可能不希望的浮点计算。另一个方法是将 RTCCLK 标称频率(32,768 Hz)作为基数，然后预先计算出补偿算法需要对 RTCMOD 初始值所减的计数值。

温度是对晶振影响最大的外部因素。为实现更佳效果，应经常对 RTC 进行温度补偿。

7 结论

RTC 模块可以补偿 RTCCLK 频率上的变化，最小可达 508.6 ns 或 0.508 ppm。不过还有其他因素会影响实际的 1 Hz 信号精度。RTC 振荡器精度、校准信号测量、温度检测和温度查找表上的四舍五入均会增加输出信号上的总误差。遵循应用笔记上的步骤可以实现总体 10 ppm 的精度，即每天 0.864 秒的偏差。

8 参考

请参见最新的器件参考手册、参考设计和其他信息：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