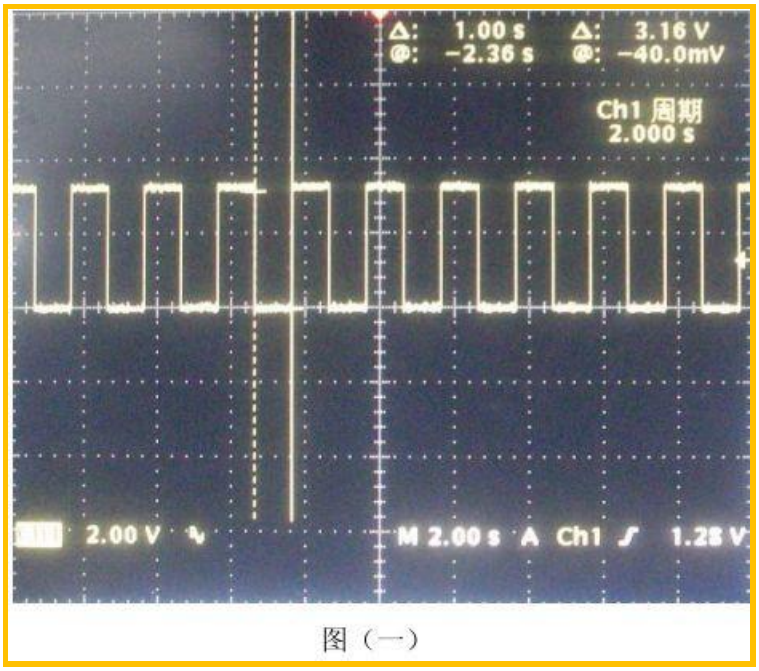
# RTC 计秒不均匀

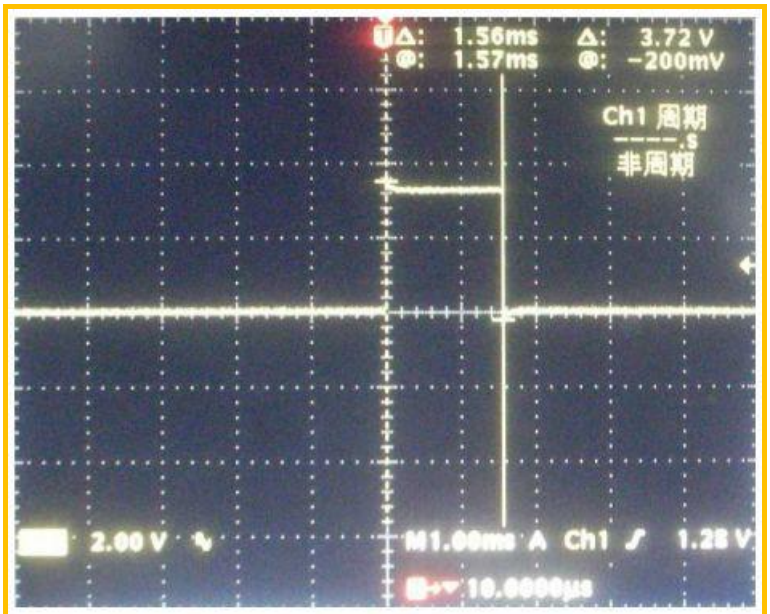
## 问题：

该问题由某客户提出，发生在 STM32F103RBT6 器件上。据其工程师讲述：其产品为车载 GPS 导航监 控设备，其中使用了 STM32 作为主控器件，负责管理整个设备。在该产品的设计中，使用了 STM32 的 RTC，并将其计时显示在产品的屏幕上。计时显示的更新是由 RTC 的秒中断来完成的，即由 RTC 的秒中断服务程序从 RTC 中读出新的时间并更新到相关的变量中，再触发屏幕刷新程序更新屏幕上的 显示。在测试时发现屏幕上显示时间的秒部分走时不均匀，时快时慢，甚至会丢掉某个中间值而发生 跳变。 对该显示时间做长时间计时的测量，发现其长时间计时是准确的，即秒长度的平均值是准确 的。将程序中的其它中断关掉，只保留 RTC 的秒中断，问题依旧。通过在 RTC 秒中断服务程序中加 入对 GPIO 翻转的代码来测量 RTC 秒中断响应的时间间隔，发现其是均匀的，如图（一）所示，说明 并非 RTC 的秒中断响应不及时而导致显示时间的波动。



图（一）

用同样的方法测量从 RTC 秒中断得到响应到完成屏幕上显示时间的更新所消耗的时间，结果为 1.56mS，如图（二）所示。这一延时不足以对屏幕上显示时间造成的可察觉的波动。到此，不知下一 步该如何定位问题原因，请求技术支持。




图（二）

## 调研：

重复观察现象，如其所述。为了便于观察，修改代码，在每次从 RTC 读取新的时间之后，将保存时间 的变量通过 UART 打印到终端软件上。其结果如图（三）所示，与设备屏幕上的显示时间是一致的：


图（三）

进一步修改代码，在每次从 RTC 读取时间数据之前加入 1mS 的等待，如表（一）所示，其中变量 TimeDisplay 由 RTC 的秒中断服务程序置“1”，触发该任务更新屏幕显示时间：

```
void Time_Show(void)
{
  /* Infinite loop */
  while (1)
  {
    /* If 1s has paased */
    if (TimeDisplay == 1)
    {
      /* Display current time */
      Delay(1);     //在从 RTC 读取时间之前延时 1mS
      Time_Display(RTC_GetCounter());
      TimeDisplay = 0;
      GPIO_WriteBit(GPIO_LED, GPIO_Pin_6, Bit_RESET);
    }
  }
}
```

表（一）

编译后重新测试，结果表明之前所述现象不再发生，如图（四）所示：


图（四）

## 结论：

对 RTC 的时间数据的读取与 RTC 内部对时间数据的更新在时间上存在竞争关系，以至两者在时间上 的顺序不确定。于是，读到的时间数据，时而是新数据，时而是旧数据，从而导致读回的时间数据的 取值会发生跳变、或者不变的现象，而不是稳定的递增。

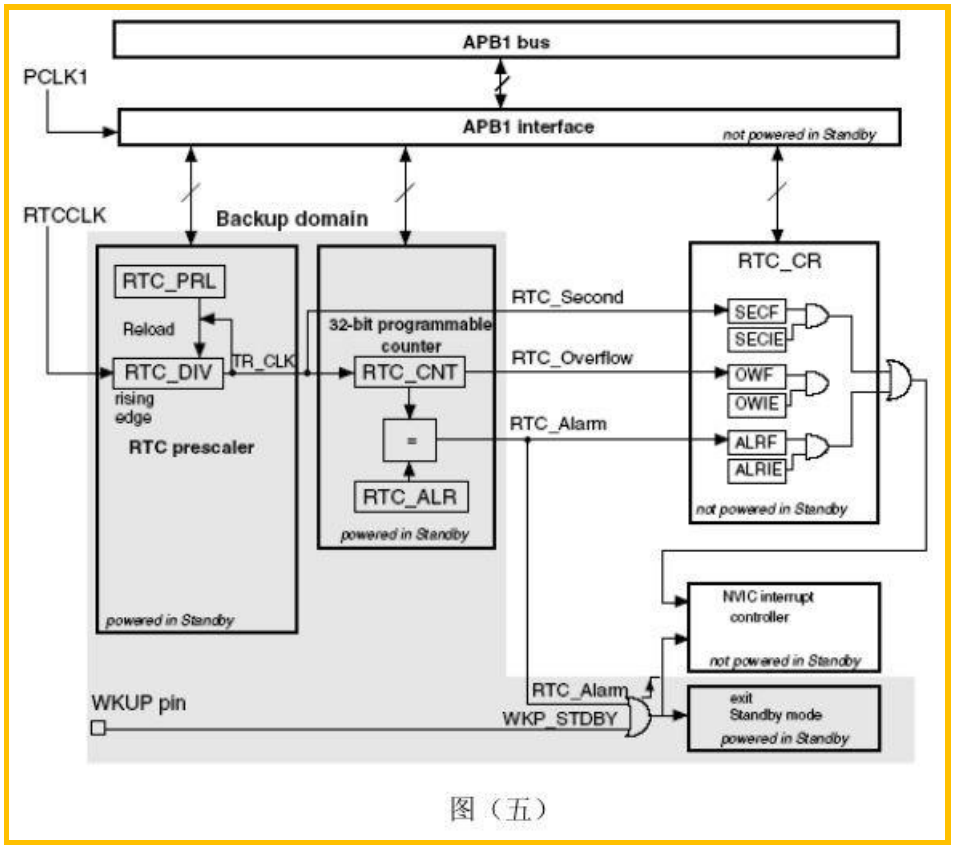
## 处理：

通过在读 RTC 的时间数据之前加入一定的延时，来保证其与 RTC 内部对时间数据的更新之间保持一 个固定的时间顺序，以便读回的都是更新后的数据。

## 建议：

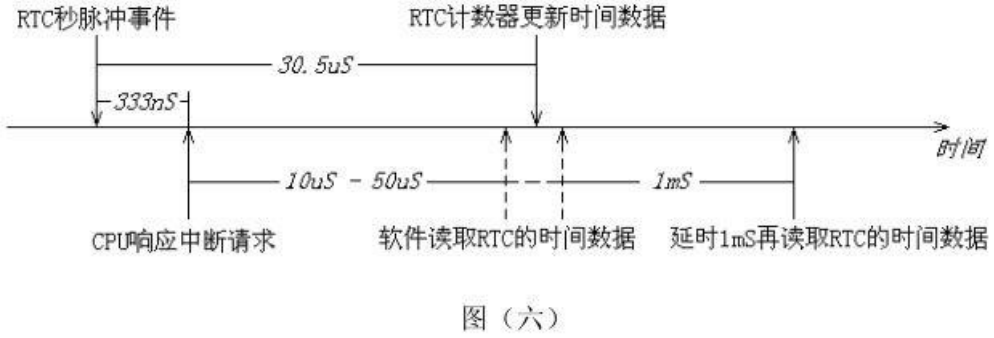
在 STM32 中，RTC 部分与 CPU 部分分别属于不同的时钟域，且两者的工作时钟频率相差较大，前者 工作在 32.768KHz 的频率下，而后者的工作频率通常在 36MHz 以上。这一差别导致两者对同一事件 做出响应的速度存在着明显的差距，CPU 的

响应速度快而 RTC 响应速度慢。从图（五）所示的 RTC 系统架构中可以看到，由前置分频器产生的秒脉冲信号送给 RTC 的计数器的同时，也通过 RTC 的中 断请求控制单元送到了 CPU 的 NVIC 单元。



图（五）

这样的信号传递关系决定了，在 RTC 的计数器接到更新计数的信号的同时，RTC 的秒中断请求信号即 已送达 CPU，而不是在 RTC 的计数器完成更新计数之后。这样一来，由于 RTC 对这一事件的响应时间 是 RTC 时钟域的 1 个时钟周期，即 $1/32.768KHz = 30.5uS$，而 CPU 对这一事件的响应时间几个到几 十 个 CPU 时 钟 域 的 时 钟 周 期 。 假 如 CPU 工 作在 36MHz 的 频 率 下 ， 一 般 来 说 ， 这 一 时 间 在 $12 \times (1/36MHz) = 333nS$ 左右，所以，CPU 要先于 RTC 的计数器响应这一事件。然而，一般情况下软 件的中断服务程序并非一开始就去读 RTC 的时间数据，而是要先做一些前期的判断和准备工作，甚至要等到中断服务程序结束后，由普通任务去读这一数据。由此，也会引发一个延时，通常这一延时的 时长在 10uS 到 50uS 之间。于是，读取 RTC 的时间数据的事件和 RTC 计数器更新计数值的事件在非 常邻近的时间点上发生，且无固定顺序。综上所述，几个事件在时间上的关系，如图（六）所示：



图（六）

为了消除 RTC 计数器更新时间数据和软件读取 RTC 时间数据这两个事件在时间上的竞争，可以让后 一事件延后一段时间之后再发生，以确保两个事件有固定的时间顺序。

## 重要通知 – 请仔细阅读

意法半导体公司及其子公司（“ST”）保留随时对ST 产品和/ 或本文档进行变更、更正、增强、修改和改进的权利，恕不另行通知。买方在订货之前应获取关于ST 产品的最新信息。ST 产品的销售依照订单确认时的相关ST 销售条款。

买方自行负责对ST 产品的选择和使用， ST 概不承担与应用协助或买方产品设计相关的任何责任。

ST 不对任何知识产权进行任何明示或默示的授权或许可。

转售的ST 产品如有不同于此处提供的信息的规定，将导致ST 针对该产品授予的任何保证失效。

ST 和ST 徽标是ST 的商标。所有其他产品或服务名称均为其各自所有者的财产。

本文档中的信息取代本文档所有早期版本中提供的信息。