RTC 校准方案

应用指导

文档版本　01

发布日期　2013-06-21

商标声明

 、HISILICON、海思和其他海思商标均为深圳市海思半导体有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受海思公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，海思公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

深圳市海思半导体有限公司

地址：　深圳市龙岗区坂田华为基地华为电气生产中心　邮编：518129

网址：　http://www.hisilicon.com

客户服务电话：　+86-755-28788858

客户服务传真：　+86-755-28357515

客户服务邮箱：　support@hisilicon.com

前 言

概述

本文档主要介绍 RTC 的校准方案，确保 RTC 计时准确。

产品版本

与本文档相对应的产品版本如下。

产品名称	产品版本
Hi3520D 芯片	V100
Hi3515A 芯片	V100
Hi3515C 芯片	V100

读者对象

本文档（本指南）主要适用于以下工程师：

- 技术支持工程师

修订记录

修订记录累积了每次文档更新的说明。最新版本的文档包含以前所有文档版本的更新内容。

修订日期	版本	修订说明
2013-06-21	01	第 1 章更新。 新增第 3 章。
2013-05-09	00B01	第 1 次版本发布。

目 录

1 概述

1.1 RTC 芯片分类

常见的 RTC 芯片，大致可分为三类：

- 非集成 RTC：只有 RTC 计时电路，不集成晶体、不集成温度补偿电路。这类芯片的计时精度主要取决于外接晶体的精度，而且受温度影响较大。通常在室温环境下，计时精度较高；随着温度升高或降低，计时偏差逐渐增大。
- 集成晶体的 RTC：将 RTC 计时电路与晶体集成，但没有温度补偿电路。这类芯片在室温环境下，计时精度更高。但仍然无法消除温度的影响。
- 集成 RTC：将 RTC 计时电路、晶体、温度补偿电路（含温度传感器）都集成在一颗芯片中，出厂时进行调教。这类 RTC 的计时精度可以做到很高，且由于温补电路的作用，受环境温度的影响很小。

1.2 RTC 工作模式

Hi3520D 内置 RTC 可支持两种工作模式：

- 固定分频模式

  与非集成 RTC 相同，Hi3520D 内置 RTC 的时钟直接采用外部晶体与振荡电路产生的经过分频后的时钟，工作时分频比固定不变。这种工作模式下，RTC 计时精度取决于外接晶体的频率精度，而且受环境温度影响。在非集成 RTC 这类芯片适用的场景，可以选择 Hi3520D 内置 RTC 替代外置非集成 RTC，节省一些器件成本。
- 温度补偿模式

  Hi3520D 内置了 RTC 计时电路，及温度补偿电路（含温度传感器），可校正因温度变化引起的计时偏差。但由于温度传感器集成在了 Hi3520D 内部，并不能真实反应外接晶体的实际温度，实际校正效果不是很理想。

1.3 温度与频率输出关系

RTC 计时时钟计算公式为：RTC 的计时时钟
=内部振荡电路产生的时钟/分频比（327.xx）。

当温度变化时，内部振荡电路产生的时钟受到影响也会变化，此时可通过调节分频比来确保 RTC 计时时钟恒定。32K 晶体的输出振荡频率与温度的关系如图 1-1 所示。

图1-1 温度与晶体频率输出关系

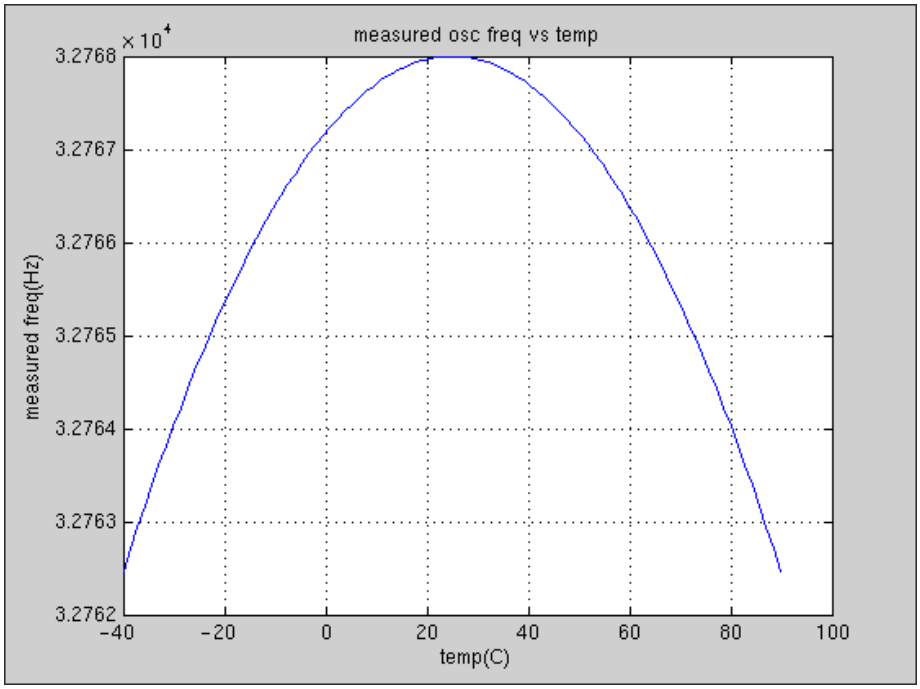


图 1-1 中的曲线可以公式表示：$F=(K_s*(T-T_0)^2+1+C)*32768$

其中：

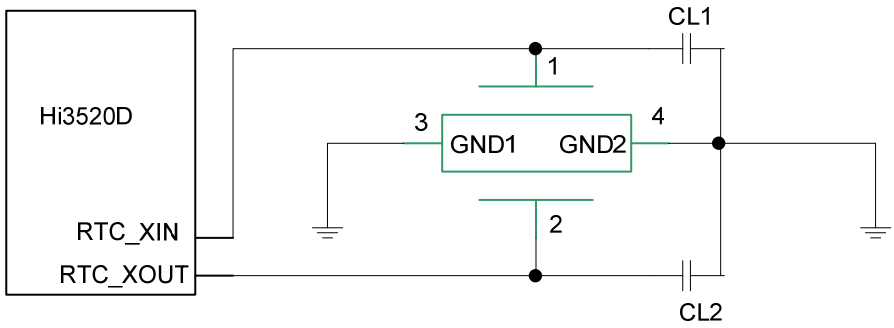
- F 为晶体在温度 T 下的振荡频率，单位为 Hz。
- Ks 为抛物线二次项系数，和选取的晶体相关，图 1-1 中的 Ks 为-4x10^{-8}/℃^{2}。
- T_0 是抛物线的转折温度点，一般为 25±5℃。图 1-1 中的值为 24.94℃。（RTC 中一个温度码字表示（140℃ -（-40℃））/255=0.705882℃，温度码字和温度的对应关系请参见《RTC 晶体校正参数生成表》）
- C 为晶体在转折温度点的频率偏差，图 1-1 中为 0。C 受以下两部分因素影响：
  - 负载电容
  - 晶体差异

2 RTC 的硬件参考电路

2.1 硬件参考电路

RTC 的硬件参考电路如图 2-1 所示，主要涉及晶体和电容的选择。

图2-1 RTC 晶体的硬件参考电路



2.1 选择晶体

选择晶体需要注意以下几个指标：

- 标准负载电容（Load capacitance/CL）：晶体的标准负载电容，晶体对负载电容有着严格的规定，只有实际负载电容和晶体的 SPEC 中的负载电容一致时，晶体频率才能达到标称频率。

  Hi3520D 芯片中的晶体振荡电路针对 CL=12.5pF 的晶体设计，且在 32.768K 晶体市场中，CL=12.5pF 的晶体为市场主流，请选用此规格晶体。如果想选用其他规格的晶体，需要按照影响 RTC 精度的因素选择匹配电容。

- 串联电阻（Series resistance/Rs/ESR）：晶体的谐振腔等效串联电阻，当 ESR 越大，表示晶体越难以驱动。晶体规格中会指出 Rs 的典型值与最大值。

  Hi3520D 芯片晶体振荡电路适用于 Rs 最大值<50KΩ 的晶体，请选用满足此规格的晶体。

- 最大驱动级别（Max Drive Level/DL）：表示晶体最大的振荡幅度，当振荡幅度超过一定幅度时，晶体容易发生损坏。

  Hi3520D 芯片晶体振荡电路内部限制了 RTC_XIN 与 RTC_XOUT 管脚振荡的振荡幅度，可以通过以下公式估计电路工作时的实际 Drive Level，并确定此值小于晶体规格中规定的最大 Drive Level。

  $DL\_actual=0.5*Rs\_max*(\pi*f*Vpp_{XIN}*CL*2)^2$

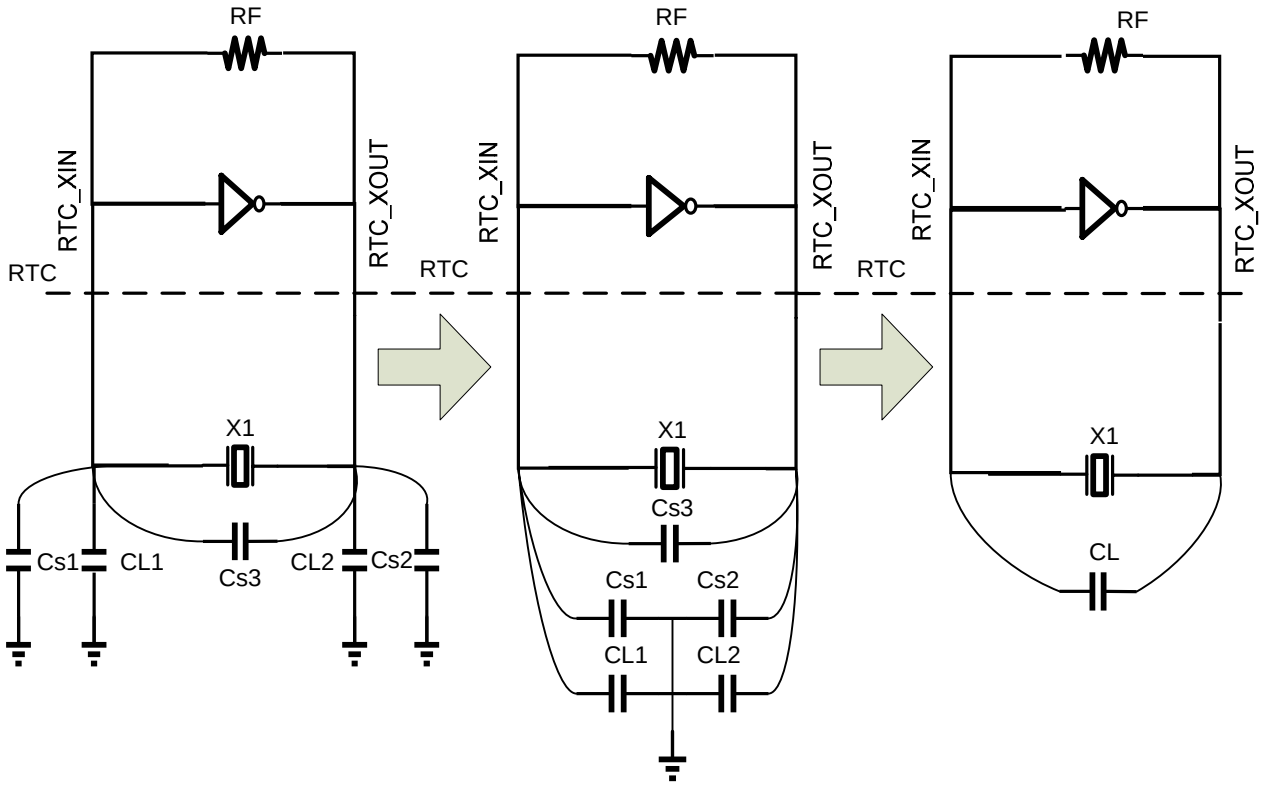
  其中：

  - Rs_max 为晶体规格书中的串联电阻的最大值
  - f 为晶体的谐振频率
  - Vpp_{XIN} 为示波器测量的 RTC_XIN 管脚的 peak to peak 电压
  - CL 为晶体规格书的标准负载电容

2.2 选择电容

实际 CL 的示意图如图 2-2 所示。

图2-2 实际 CL 的示意图



Pierce 振荡器中，一般将 CL1 与 CL2 取相同的值，可以通过以下公式来确定 CL1 与 CL2 的取值。

CL1=CL2=CL_SPEC*2-3~5pF

其中 CL_SPEC 为晶体规格书中规定的标准负载电容，3~5pF 代表的 PCB 板可能引入的杂散电容。以 12.5pF 的晶体为例，CL1 与 CL2 一般取值为 12.5pF*2-3pF=22pF。因为杂散电容随 PCB 板设计不同而变化，故亦可确定 PCB 板以后，通过选取不同容值的 CL1 来获得最接近 32.768K 的输出频率。

3 RTC 固定分频模式的实现

RTC 在固定分频模式下不进行温度补偿。RTC 的时钟直接采用外部晶体与振荡电路产生的进行 327.xx 分频后的时钟，RTC 精度取决于外部晶振提供时钟的准确性。小数分频的分频系数可以调整。本方案与非集成 RTC 芯片具有类似精度。

固定分频模式配置比较简单，涉及的 RTC 内部寄存器有 3 个：0x21，0x51 和 0x52。

- 0x21 配置为 0x6，把 RTC 的时钟设置为外部方波分频后的时钟，关闭温度补偿。
- 0x51 和 0x52，这两个寄存器连在一起为一个 16bit 的寄存器，它们的值决定小数分频的分频系数，具体计算方法如下：

  分频系数=327+（寄存器读取值/3052）

  例如：0x51 的值为 0x8，0x52 寄存器的值为 0x1b，它们连在一起的 16bit 的值为：0x81b。0x81b 的十进制为 2075，分频系数=327+（2075/3052）=327+0.68=327.68。

小数分频的分频系数可以微调是为了使分频后的时钟更加接近 100Hz，这样 RTC 的精度会有所提高。调节分频系数通常应用在时间统一偏快或者偏慢的情况。例如，假设晶振的输出频率为 32767.00Hz，若使用默认分频系数 327.68，则分频后的时钟是 99.97Hz，时钟会偏慢。若把分频系数设置为 327.67，则分频后的时钟为 100Hz，会改善时钟偏差的情况。

4 RTC 校正功能的实现

RTC 校正的频率范围为 32760~32776Hz (全温度范围下)。实现 RTC 校正功能需要以下步骤：

1. 获取晶体温度-频率曲线

   在厂家发布的晶体参数手册中，通常会有标称的曲线抛物线系数二次 Ks 和抛物线的转折温度点 T_0。只测量室温下的晶体频率输出值并记录晶振所处环境温度，可以直接使用单点法进行校正。如果对其标称参数不满意，可以按以下步骤测量以获得更真实的曲线。

   1) 准备必备的测试工具：温箱、频率计、外置温度传感器

   2) 准备好单板，上电启动 Linux 系统，配置 VI_ADC_CLK 管脚复用为 RTC 测试时钟输出（0x200F_0000 = 03b'110）；配置 RTC_CLK 寄存器为输出晶体的振荡时钟（02b'00）

   3) 使用多点测量法，测量并将数据记录到《RTC 晶体校正参数生成表》。

      如果直接使用晶体厂家发布的参数，则需计算多组温度-频率数据对，记录到《RTC 晶体校正参数生成表》。

   4) 点击《RTC 晶体校正参数生成表》中的生成按钮，会自动生成 rtc_temp_lut_tbl.h

2. 根据温度频率曲线配置 RTC

   1) 使用 rtc_temp_lut_tbl.h 覆盖 RTC 源代码目录下的同名文件，重新编译驱动

   2) 加载新生成的 hirtc.ko

   3) 驱动加载成功后，RTC 即成功启动；可使用附带的 test 程序，进行时间配置等操作

3. 配置 RTC 校正功能

   1) 上电时，如果使用内部 T-Sensor，建议选择固定温度值模式，并选择合适的温度（根据实际环境温度定，例如室温 25℃时，晶体温度可能会高出 10℃，可配置 RTC 温度 35℃）；如果使用外部 T-Sensor，建议参考 4.2 业务运行时 RTC 校正描述的配置流程。

   2) 下电时，无需任何操作，RTC 会自动切换到内部 T_sensor 的测温模式。

----结束

4.1 测量晶体温度频率曲线与配置 RTC

测量晶体的温度频率曲线

通过将测量的温度与晶体振荡输出频率写入《RTC 晶体校正参数生成表》，即可生成温度频率曲线。在测量温度频率曲线时，根据测量的精度和成本因素，选择以下几种测量方法：

- 多点测量法。

  在全温度范围内测量多个温度与晶体振荡输出频率的关系，该方案适用于晶体参数需要重新评估的情况。

- 三点测量法。

  只取-20°C、25°C 和 80°C 下测量到的频率值，该方案适用于快速确定晶体参数。

- 两点测量法。

  取-20°C 和 80°C 下测量到的频率值，该方法适合于已知晶体型号的情况下使用，因为对同一型号的晶体温度频率曲线抛物线二次系数 Ks 已知，且变化不明显（晶体一致性较好的情况下）。

- 单点测量法。

  只测量室温下的晶体频率输出值，该方法适合于使用同一型号同一批次的晶体，因为此时晶体温度频率曲线抛物线系数二次 Ks 和抛物线的转折温度点 T_0 已知，且变化不明显（晶体一致性较好的情况下）。

曲线测试步骤

- 测试工具：温箱、频率计、外置温度传感器。
- 测试时业务：内核态下测试最佳。
- 测试温度点：测试的方法有多点测量法、三点测量法、两点测量法、单点测量法。原则上是测试温度点越多，曲线的精度越高。

  通常情况下，同一批晶体的个体之间存在 20ppm 的个体差异，若需要把个体差异减小，可以测量多个晶体的“温度-频率曲线”然后取众多曲线中的比较靠中间的一条曲线；使用这种方法若拟合得好，理论上可以减小 10ppm 左右的误差；条件允许情况下一般会采用多点测量法，推荐的测试温度点如下：-20℃，-10℃、15℃、25℃、35℃、60℃、70℃。

- 测试环境准备：由于 RTC 的曲线测量对温度和干扰特别敏感，故对测试环境要求比较严格；环境准备时注意以下几点：
  - 外部温度传感器贴近晶振，最好用硅脂沾连在一起，保证测试温度准确。
  - 测试频率期间不能使用任何探头接触晶振管脚。
  - 测试频率时，请使用 RTC 频率测试复用管脚(VI_ADC 管脚)。
  - 使用抗干扰性好的铜轴线引出测试信号，RTC 输出频率通常会稳定在小数点的后两位，例如：32767.97xxHz，xx 表示跳动小数位，可以忽略；若达不到稳定频率说明环境存在干扰，需要检查接地情况或者排查、更换测试坏境。
- 测试与数据记录：调节温箱温度，待温箱温度稳定一段时间后读取外置温度传感器的温度（即晶体温度），记录 RTC 输出频率；频率和温度构成一条二次曲线，称该曲线为：频率-温度曲线。

- 条件允许情况下，可以测量同一批晶体的 4 个晶体以上的曲线，然后对这些“频率-温度曲线”进行拟合，选出最合适的一条。

配置 RTC 的温度频率曲线

晶体频率输出和温度的曲线对应 RTC 内部寄存器的寄存器，其中寄存器 LUT_n 与 $K_s*(T_n-24.94)^2$（$n \in [1,47], T_n = -40 + n*(180/255)$）相关，寄存器 TEMP_OFFSET 与 T_0 相关，寄存器 TOT_OFFSET 与 C 相关。

选择合适的方法后，将测量到的频率和温度测量数据填入相应的表格中，点击生成即可。将生成的 rtc_temp_lut_tbl.h 文件覆盖 RTC 驱动中相应的文件，重新编译 RTC 的驱动，这样芯片在每次启动时就会把曲线配置到 RTC 中。

由于晶体之间的差异，生成的温度和频率曲线不同。综合考虑精度和成本，配置 RTC 驱动文件的方案有以下三种：

- 每个晶体都进行测量，描述出自己特有的曲线进行配置。这种方法精度可以达到 5ppm，但是每个产品都要进行校正曲线，根据生成的 rtc_temp_lut_tbl.h 曲线配置文件。
- 对同一批次中一定数目的晶体进行测量，对拟合出的曲线进行再拟合，然后所有的晶体都按这个再拟合的曲线进行配置。这种方法的精度为 5ppm+晶体个体与一致性的偏差，精度受晶体生产的一致性的影响。
- 不做任何测量，采用芯片初始的曲线或者晶体厂商提供的数据生成的曲线配置文件。这种方案的精度完全由晶体的一致性以及晶体固有的温度频率曲线与 RTC 配置的晶体温度频率曲线的吻合度来决定。

影响 RTC 精度的因素

RTC 的误差=电容与晶体匹配误差+晶体个体差异+温度漂移误差+（温度偏差）。

- 电容与晶体匹配。对于晶体，晶体手册有电容规格（Load capacitance,CL）说明，电容之间有以下关系： C`+(C0+C1)/2=CL，其中 C`为单板走线电容，C0 和 C1 为晶体串联电容，CL 为晶体电容；晶体和电容匹配是保证 RTC 精度的前提，电容过大则晶体输出频率偏小，电容过小则晶体输出频率过大；电容过小还可能会引起倍频情况，故电容需要选与晶体匹配的。若有校正或者温度补偿方案，则电容与晶体不匹配带来的误差理论上可以消除，消除后这部分误差可以认为为 0。
- 晶体个体差异。在同一批晶体中，晶体与晶体之间也存在差异，个体差异一般为 20ppm，故若想精度统一性好就得选择正规的晶体；若对每一个晶体都进行测量温度频率曲线并校正，则这部分误差也是可以减小的，校正后该部分误差可以小于 5ppm，这也是高端 RTC 芯片的做法。次之的做法是取同一批晶体的一条曲线或者多条曲线的拟合曲线作为代表，这样做的好处是方便，但是会引入一定误差；若测试多个晶体的曲线然后进行拟合，则精度会好一些，引入误差在 10ppm 左右；若值测试一个晶体，则引入的误差是随机的，引入误差为 10ppm~20ppm。
- 温度漂移误差。机箱内温度变化会使得晶振的温度变化，晶振温度变化会引起晶体频率漂移。晶振的输出频率与温度强相关，特别是在高温和低温段。晶振温度每变化 1℃引起的频率误差在各个温度段是不一样的。在中心温度处受温度变化影响不明显，在高温或者低温受温度变化影响比较明显。若测试了晶体的曲线并进行温度补偿，则可以减小或者消除该部分误差。

- 温度偏差。这个偏差仅在使用内部 T_sensor 校正的时候存在。内部 T_sensor 校正的方法是通过读出芯片内部温度从而推算出来晶振的温度，得到晶振的温度后进行温度补偿。若所有的芯片均采用同一温度偏差，则这部分误差来源于 3 部分：
  - 内部 T_SENSOR 引起的误差，不同芯片的内部 T_SENSOR 的读数可能会存在差异，差异在 5℃左右。
  - 由于 T_sensor 实际测量的是主芯片温度，通过减去一个估计偏差值，得到晶体温度。在正常业务场景下，主芯片温度与晶体温度偏差不固定，且可能有较大幅度的波动，可能会引入较大的误差。但在下电放置场景下，主芯片温度与晶体温度基本一致，误差会非常小。
  - 晶振放置的位置、机箱内散热环境的差异，也会造成晶体的温度误差。

4.2 业务运行时 RTC 校正

校正温度的方案以下三种：

- 固定温度值。

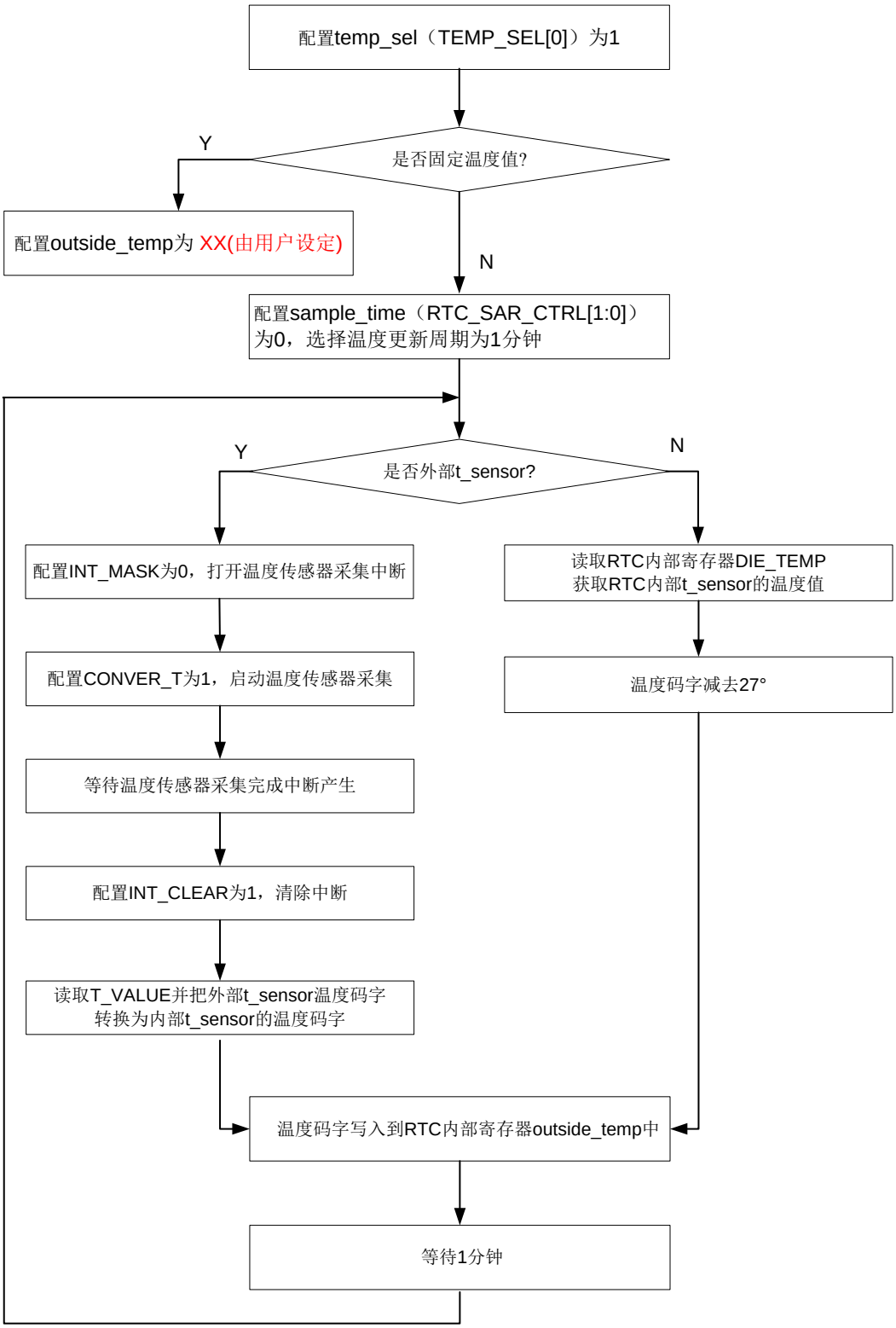
  适用于芯片工作后，晶体温度变化不明显的情况。
- 采用 RTC 内部 T_sensor 的测量值。

  将 T_sensor 的测量值，减去一个经验偏差值(例如 27℃)。在实际业务场景下，精度不高。
- 采用外部 T_sensor 的测量值。

  外部 T_sensor 贴近晶体表面，精确反应晶体的温度，如果 T_sensor 足够精度，RTC 校正的理论精度达 5ppm。

在配置好晶体振荡曲线相关的寄存器后，只需定时给 RTC 的温度寄存器刷新温度就可以实现校正。温度更新配置的流程如图 4-1 所示。

图4-1 温度更新配置流程图



4.3 单板下电时 RTC 自动校正

单板下电时，RTC 电路检测到单板下电信号，自动切换到电池供电。使用电池供电时，若有外部 T_sensor，则外部 T_sensor 将不再工作，温度采集的来源变为 RTC 内部寄存器 DIE_TEMP，此时认为环境温度、芯片温度相同，RTC 内部寄存器的值不变，使用的校正曲线与上电时的校正曲线相同。当板子重新上电时，RTC 会重新恢复到之前上电时配置的模式下工作。

5 RTC 驱动使用说明

5.1 准备

通过 4.1 测量晶体温度频率曲线与配置 RTC 的描述，在《RTC 晶体校正参数生成表》中得到晶体温度频率曲线，点击 Create rtc_temp_lut_tbl.h 生成 rtc_temp_lut_tbl.h 头文件，用此文件更新 RTC 目录中的同名文件。若使用固定分频模式，则该文件使用默认文件即可。

5.2 编译

在 RTC 目录下执行下述命令即可生成对应的驱动 hirtc.ko 及示例程序 test。

```
cd rtc
make
make test
```

5.3 使用

将 hirtc.ko 拷贝到单板，并执行如下命令插入驱动模块：

```
insmod hirtc.ko t_second=T
```

其中 t_second 模块参数表征温度采集的时间间隔，以秒为单位，驱动默认为 5s。如不必修改，则无需传入模块参数。若使用固定分频模式，直接使用 insmod hi_rtc.ko 命令插入 ko 即可，无需传入参数。

RTC 驱动提供的功能通过单板上运行的 test 示例程序说明，如图 5-1 所示。

图5-1 示例程序用法

```
# ./test

Usage: ./test [options] [parameter1] ...
Options:
        -s(set)         Set time/alarm,      e.g '-s time 2012/7/15/13/37/59'
        -g(get)         Get time/alarm,      e.g '-g alarm'
        -w(write)       Write RTC register,  e.g '-w <reg> <val>'
        -r(ead)         Read RTC register,   e.g '-r <reg>'
        -a(alarm)       Alarm ON/OFF',       e.g '-a ON'
        -reset          RTC reset
        -c(ompensation) temperature compensation ON/OFF, eg '-c ON'
        -f(requency)    frequency precise adjustment, eg '-f <val>'
        -m(mode)        Mode of temperature gather, e.g '-m <mode> <temp>, mode[0-2]'
```

设置获取时间

通过如下命令可设置 RTC 时间：

```
./test -s time <year/month/day/hour/minute/second>
```

通过如下命令可获取 RTC 时间：

```
./test -g time
```

设置获取闹钟时间

通过如下命令可设置 RTC 闹钟时间：

```
./test -s alarm <year/month/day/hour/minute/second>
```

通过如下命令可获取 RTC 闹钟时间：

```
./test -g alarm
```

通过如下命令设置闹钟到期是否产生中断，驱动中断例程由用户根据需求自由补充。

```
./test -a ON/OFF
```

读取、设置 RTC 内部寄存器

通过如下命令可读取 RTC 内部寄存器，此功能多用于辅助调试，比如读取内部温度传感器采集的温度值，读取设置的 RTC 更新温度值等。

```
./test -r <reg>
```

通过如下命令可设置 RTC 内部寄存器，此功能多用于辅助调试。

```
./test -w <reg> <value>
```

reg 取值，请参见《Hi3520D / Hi3515A H.264 编解码处理器用户指南.pdf》3.9 节实时时钟部分。

复位 RTC 模块

通过如下命令可复位 RTC 模块。

```
./test -reset
```

温度补偿模式开关

通过如下命令可设置温度补偿模式是否打开。 ON 表示打开 RTC 温度补偿，OFF 表示关闭温度补偿而采用固定分频模式。若单板复位重启，或者单板带有电池断电重启，重插 ko 后系统将运行在固定分频模式。若用户使用的是温度补偿模式，在重插 ko 后需要重新打开温度补偿模式。

```
./test -c ON/OFF
```

固定分频模式分频系数微调设置

通过如下命令可设置分频系数从而达到调整时钟的快慢效果。

```
./test -f <val>
```

< val>值为将要设置的分频系数的 10000 倍，例如要设置分频系数为 327.60，则 val=3276000。通过直接敲“./test –f”命令可以查看当前分频系数。分频系数可以配置范围为：327.60~327.70。

说明

频率系数仅在固定分频模式下设置，设置流程为：关闭温度补偿，然后设置分频系数（也可以采用默认值）。

设置温度采集方式

通过如下命令可设置温度采集方式：

```
./test -m <mode> <value>
```

Mode 和 value 取值如表 5-1 所示。

表5-1 Mode 和 value 取值表

温度采集方式	Mode	Value
固定模式，手动更新	0	晶体环境温度值。
外部 sensor 采集，自动更新	1	NA
内部 sensor 采集，自动更新	2	经验偏差值

说明

温度采集方式设置仅在温度补偿模式下设置。设置流程为：打开温度补偿，然后设置温度采集方式。

用户接口

请参看 hi_rtc.h 文件。

6 Q&A

6.1 振荡器不振

【现象】

32.768K 时钟无输出，RTC 计时电路中的秒寄存器值恒定不变。

【分析】

使用示波器探头观察 RTC_XIN 管脚振荡波形，引起不同的振荡波形的情况有以下几种：

- 如果无振荡波形，可能是晶体损坏。
- 如果有 32K 左右频率正弦波，且 peak to peak 幅度小于 600mV，则可能是因为 CL1 与 CL2 过大，导致振荡电路驱动不足，从而幅度偏小，振荡波形无法通过后续的施密特触发器。
- 如果有 200K 左右频率的正弦波，且 peak to peak 幅度小于 600mV，则可能是因为 CL1 与 CL2 偏小，导致振荡电路振荡到 200K 频点，且由于 200K 频点振幅较小，所以无法通过后续的施密特触发器。

【解决】

- 如果确定晶体损坏，请更换晶体。
- 如果为 32K 左右频率正弦波，幅度不够，请检查 CL1 与 CL2 是否偏大，并更换正确的电容。
- 如果为 200K 左右频率正弦波，幅度不够，请检查 CL1 与 CL2 是否偏小，并更换正确的电容。

6.2 振荡器的输出频率是 200K

【现象】

32.768K 时钟输出频率接近 200K，RTC 计时电路中的秒寄存器值每秒钟增加 6。

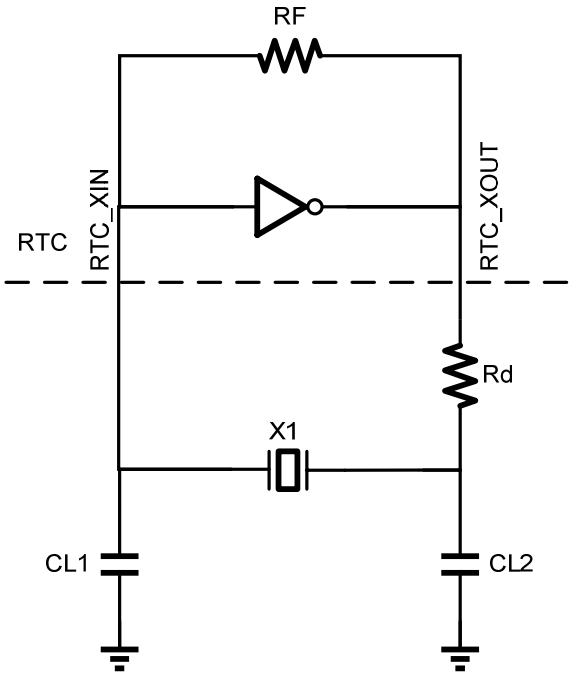
【分析】

因为 32.768K 晶体存在 6.1 倍频的谐振点，如果晶体有异常，则可能振荡到 6 倍频附近。

【解决】

建议首先检查 CL1 与 CL2 是否偏小；如果 CL1 于 CL2 为正确值，但振荡频率仍然为 200K，则可以在电路中添加如图 6-1 所示的 Rd，Rd 取值为 1/(2 π x 32768 x CL2)，Rd 与 CL2 可以形成一个 RC 滤波器，降低 6.1 倍频处的环路增益。

图6-1 200K 振荡解决方案电路



注意：一般情况不建议增加 Rd，如果采用增加 Rd 的方法，需要确定 RTC_XOUT 管脚信号幅度不会过小。

6.3 振荡频率虽然是 32.768K 附近，但是频率却不准

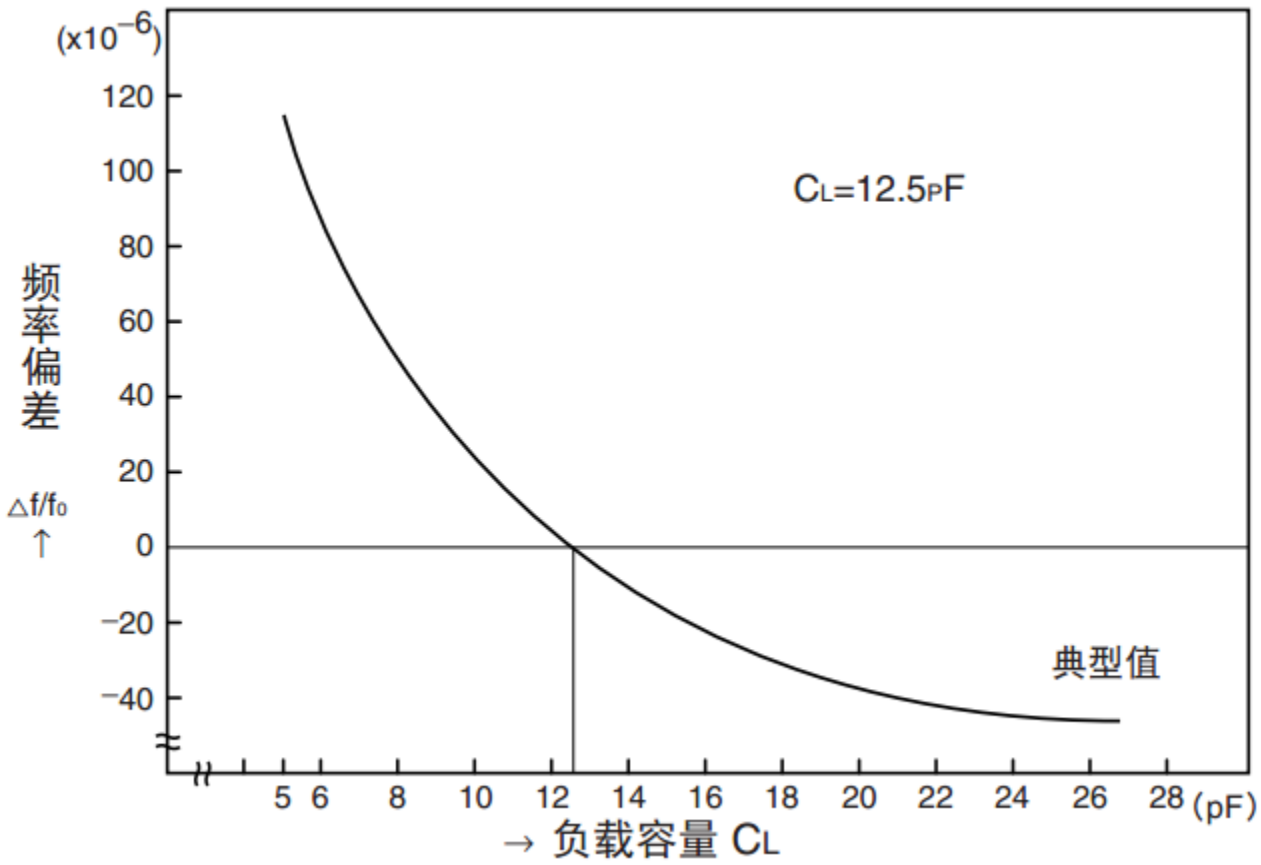
【现象】

32.768K 时钟输出频率偏离 32.768K。

【分析】

振荡电路振荡频率主要由晶体和负载电容共同保证，晶体本身确定了频率的大致范围（即图 6-2 所示中的 0 偏差对应的频率），而实际负载电容的大小则确定了频率的偏移量（即图 6-2 所示中的实际频率偏离 0 的值）。

图6-2 频率偏差和负载容量 C_L 的关系



【解决】

如果频率偏离了 32.768K，首先需要确认晶体管脚弯折不会对内部晶体部分施加应力，并确定焊接过程中的温度符合 datasheet 规范；其次，检查 CL1 与 CL2 取值是否正确，并更换正确的电容。