AN5042
应用笔记

STM32 无线 MCU 的 HSE 频率和启动时间的精确调谐

引言

本应用笔记描述了如何使用 STM32WB 和 STM32WL 系列微控制器（以下简称 STM32 无线 MCU），对用于 RF 应用的 HSE 进行调谐。此类产品提供了一种具有成本效益的高效解决方案，其中通过使用其内部负载电容来控制振荡器精度，从而节省了外部电容的成本并减少了晶振限制。

STM32 无线 MCU 采用外部振荡器高速时钟源作为 RF 时钟生成的基础。HSE 精度对于 RF 系统性能至关重要，因此对外部振荡器进行精细调谐来实现最高时钟精度。

本文档的第一部分介绍了晶体振荡器解决方案。第二部分介绍并比较了三种 HSE 频率调谐方法，即手动调谐方法、自动调谐方法和基于 STM32CubeMonitor-RF 的调谐方法（仅适用于 STM32WB 系列）。以下各节描述了这些方法在 STM32WB 和 STM32WL Nucleo 板上的应用，提供了采用 STM32Cube 扩展包 X-CUBE-CLKTRIM 的固件和脚本样例。

其中一节专门描述了如何配置 HSE，使其启动阶段稳定可靠并针对 STM32WB 系列微控制器得到优化。

本文档必须与参考手册和数据手册（可从 *www.st.com* 下载）一起阅读。

目录

表格索引

图片目录

1 HSE 振荡器

RF 系统需要高频精度才能实现最佳性能。任何时钟偏差都可能导致系统故障和/或性能下降。

表 1 展示了 STM32WB 系列微控制器支持的两种 RF 协议的精度要求。有关其他协议和标准，请参阅相应的规范。

表 1. 射频协议的载波精度要求

RF 标准	载波精度
Bluetooth®低功耗	± 50 ppm
IEEE 802.15.4 / Thread	± 40 ppm

在基于 Arm® (a) Cortex®内核的 STM32 无线 MCU 中，RF 时钟由高频 VCO 提供，该 VCO 将使用外部晶体的嵌入式振荡器产生的信号作为参考。

该晶体是 RF 合成器和微控制器的 HSE（高速外部）时钟源。其标称频率可能会有所不同，具体取决于工艺变化、使用的晶体和 PCB 设计等因素。HSE 错误会直接传输到 RF 时钟，因此必须通过调整晶体端子处的负载电容进行精细调谐。

STM32 无线 MCU 具有采用内部负载电容的高效架构，使用户可对晶体频率进行微调，而无需额外的外部电容成本。

振荡器启动阶段的可靠性取决于其实际实现（相关晶体、外部组件、环境）。对于 STM32WB 系列，提出了一种调整其内部操作的方法，以便对这一阶段进行优化和巩固。较长的启动时间可提高可靠性，但会增加功耗。

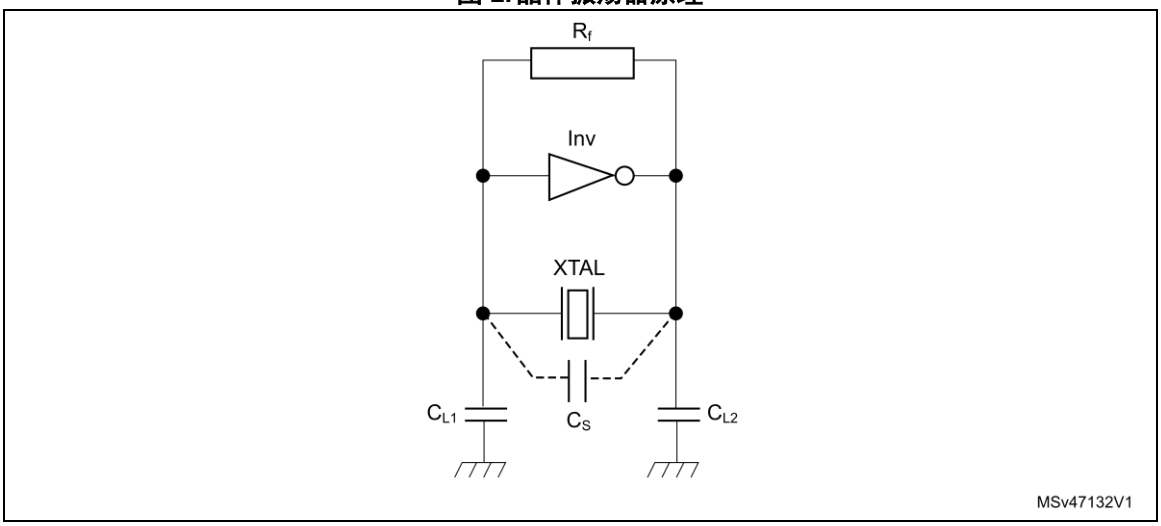
注意： *AN2867（意法半导体微控制器振荡器设计指南）总体上介绍了 STM32 产品的 HSE，由于 RF 限制，不适用于 STM32 无线 MCU。本文档是这些产品的正确参考文献。*

a. Arm 是 Arm Limited（或其子公司）在美国和/或其他地区的注册商标。

1.1 晶体振荡器

图 1 展示了晶体振荡器系统的原理。振荡器由反相放大器、反馈电阻（R_f）、晶振（XTAL）和两个负载电容器（C_{L1} 和 C_{L2}）组成。C_s 是杂散（寄生）电容，从 MCU 引脚电容之和（OSC_IN 和 OSC_OUT 之间）得出。该值可以忽略不计，无需确切掌握，因为微调和启动优化流程不依赖于该值。

图 1. 晶体振荡器原理



C_L 负载电容

负载电容是连接到晶体振荡器的电路的端子电容。该值由外部电容器 C_{L1} 和 C_{L2} 以及印刷电路板和连接件（C_s）的杂散电容决定。C_L 值由晶体制造商指定。为了使频率准确，振荡器电路必须向晶体显示出与调整晶体时相同的负载电容。

频率稳定性主要要求负载电容恒定。外部微调电容器 C_{L1} 和 C_{L2} 用于调整所需的 C_L 值，以达到晶体制造商指定的值。

以下等式给出了 C_L 的表达式

公式 1：负载电容

$$C_L = \frac{C_{L1} \times C_{L2}}{C_{L1} + C_{L2}} + C_s$$

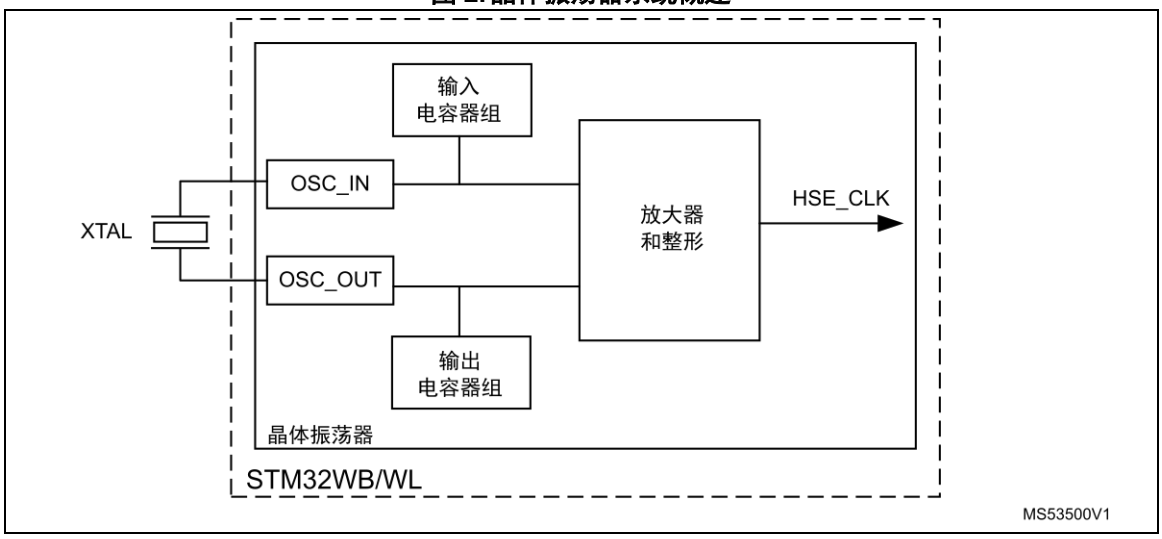
1.2 STM32 无线 MCU 架构

这些 MCU 嵌入了一个具有成本效益的高效晶体振荡器系统，具有用于微调的内部电容。用于负载电容调谐的内部机制具有双重优势：

- 减少了对外部晶体的精度限制
- 减少了 PCB 的全局 BOM（和封装）。

图 2 展示了嵌入在 STM32 无线 MCU 中的晶体振荡器系统。晶体是唯一的外部组件。无需额外的负载电容。

图 2. 晶体振荡器系统概述



晶体振荡器系统由两个焊盘（OSC_IN 和 OSC_OUT）及其各自的电容组，以及放大器级组成。

对于 STM32WB 系列，输入组和输出组的电容值相同。该值与振荡器增益和检测（用于优化启动阶段的参数）一起由寄存器驱动，并对系统行为进行控制。这些参数在*第 1.3 节*中进行了说明。

对于 STM32WL 系列，输入组和输出组的电容值是独立的。这两个值由两个 sub-GHz 射频寄存器驱动，并对系统行为进行控制。这些参数在*第 1.4 节*中进行了说明。

1.3 HSE 配置参数-STM32WB 系列

可以设置三个参数来控制振荡器模块。这些参数可在下述 RCC_HSECR 寄存器中访问。

RCC_HSECR

地址 0x09C

复位 0x0000 0030

访问 该寄存器具有实时修改保护。
在任何单次写入访问之前，必须在寄存器地址写入密钥（0xCAFECAFE）以将其解锁，之后会再次锁定。
在寄存器访问流程中，必须关闭 HSE 时钟，以免发生不可预知的行为。请注意，在该步骤中，不得将 HSE 用作 CPU 时钟源。
在本文档中，复位启动后，使用默认 MSI 时钟作为系统时钟源。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	HSETUNE[5:0]						Res.	HSEGMC[2:0]			HSES	Res.	Res.	未锁定
		rw	rw	rw	rw	rw	rw		rw	rw	rw	rw			rw

负载电容：HSETUNE[5:0]

这是负责时钟精度的参数。它选择在输入和输出焊盘上添加的电容值。可调范围设置为具有 12 至 16 pf 的全局负载电容。最小值（0x00）和最大值（0x3F）分别对应于最小和最大负载电容。

默认值为 0x00（最小负载电容）。

电流控制：HSEGMC[2:0]

该参数（在其他 STM32 产品文档中称为 Gm 或 Gm_crit_max，其具有固定值）是振荡器的最大临界晶体跨导。它控制系统的启动性能，其值必须大于 gmcrit（请参阅*第 8 节*）。值越小，功耗越低，而值越高，启动时间越短。

最小值（0b000）对应于 Gm=0.18 mA/V，最大值（0b111）对应于 Gm=2.84 mA/V。默认值为 0x3（Gm=1.13 mA/V）。

HSES 检测放大器阈值

该参数控制振荡器启动的内部比较阈值。当将此位置（1）时，启动时间减少（从大约 15 µs），但电流功耗更高，因为 HSE 启动得更早。

默认值为 0x0（1/2 比率）。

1.4 HSE 配置参数-STM32WL 系列

对于 STM32WL 系列，可以设置两个参数来控制振荡器模块。它们可在 SUBGHZ_HSEINTRIMR 和 SUBGHZ_HSEOUTTRIMR sub-GHz 射频寄存器中访问，这些寄存器分别包含输入组和输出组的电容值。相关值由其六个较低位表示（这些寄存器的其余位必须保持在其复位值）。

对于两个寄存器：

- 0x00 对应于最小电容（~11.3 pF）
- 0x2F 对应于最大电容（~33.4 pF）
- 值不得超过 0x2F，并且微调步长为~0.47 pF
- 复位值为 0x12，对应于~20.3 pF。

如前所述，SUBGHZ_HSEINTRIMR 和 SUBGHZ_HSEOUTTRIMR 是 sub-GHz 射频的一部分，而不是系统 CPU 的一部分。要修改其值，用户代码必须通过其 SPI 接口与 sub-GHz 射频通信。考虑到该 SPI 接口，这些寄存器的地址如下：SUBGHZ_HSEINTRIMR 为 0x911，SUBGHZ_HSEOUTTRIMR 为 0x912。

1.5 板实现

振荡器焊盘采用不同的引脚（命名为 OSC_IN 和 OSC_OUT），具体取决于封装。*表 2* 展示了 STM32WB 系列 Nucleo 和 USB Dongle 板中使用的不同封装的引脚号。

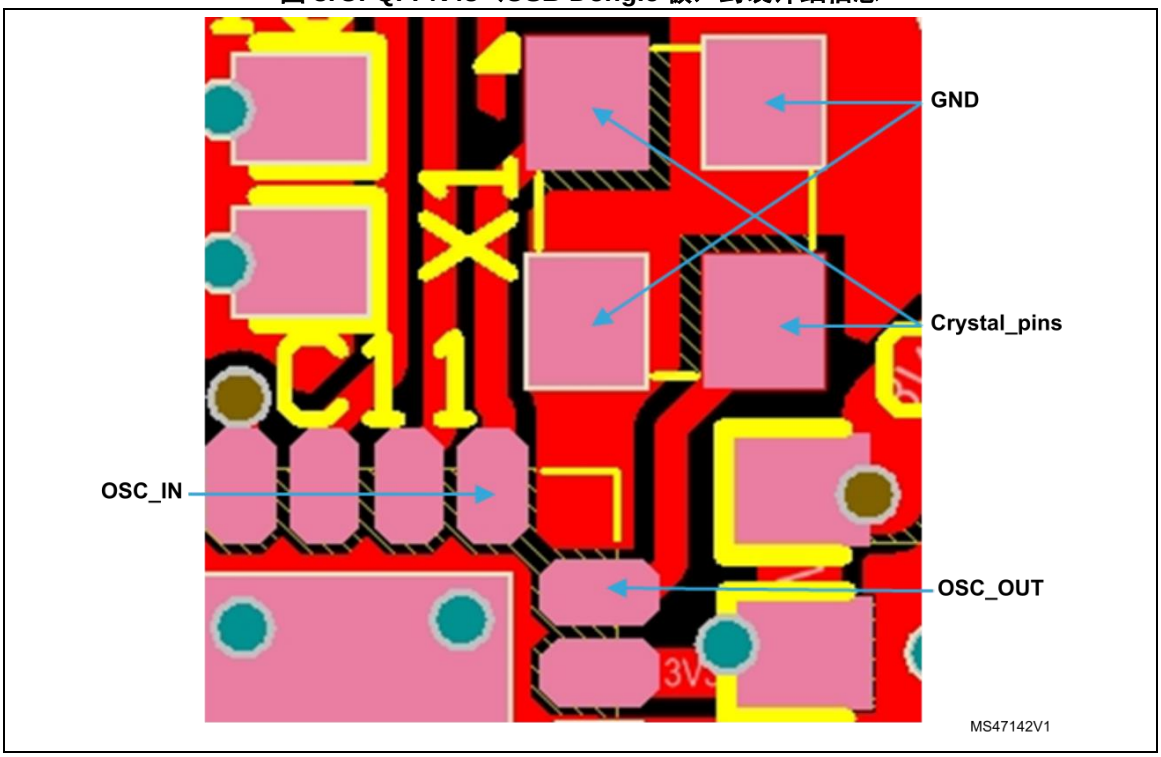
表 2. STM32WB 系列振荡器引脚数

封装	OSC_IN	OSC_OUT
UFQFPN48	25	24
WLCSP49	F1	F2
VFQFPN68	35	34
WLCSP100	J1	J2
BGA129	M13	N13

晶体直接插到焊盘上，没有额外的电容，并且尽可能靠近器件，以尽量减少寄生电容。

图 3 展示了 STM32WB 系列的典型 UFQFPN48 封装。

图 3. UFQFPN48（USB Dongle 板）封装详细信息



在 AN5165“使用 STM32WB 微控制器开发 RF 硬件”和 AN5407“STM32WL 系列优化 RF 板布局”（均可从 *www.st.com* 下载）中，分别详细介绍了 STM32WB 和 STM32WL 系列参考设计的 PCB 布局。

1.6 晶体参考

表 3 展示了用于验证参考设计的晶体规格。

表 3. 晶体规格

系列	NDK 晶振	参数	值
STM32WB	NX2016SA 32 MHz EXS00A-CS06654	负载电容	8 pF
		频率容差	(25 ± 3°C) ± 10 × 10^{-6}
		频率与温度（参考+25°C）	± 25 × 10^{-6}
STM32WL	NX2016SA 32 MHz EXS00A-CS06465	负载电容	10 pF
		频率容差	25°C 时为±10 ppm
		频率与温度	±10 ppm（-25-+70°C） ±30 ppm（-40-+85°C）

可以使用其他晶体，前提是这些晶体符合载波精度要求（参见*表 1*）。更精确地说，所选晶体的以下参数的最大值之和必须低于载波精度要求：

- 频率容差
- 频率与温度特性（对于必须使用晶体的范围）
- 老化（在必须使用晶体的持续时间内）。

注意： *此类数据可以在晶体文档中以不同方式命名或者不存在。*

如果按照*第 1.5 节*中提到的要求基于所选晶体进行 PCB 设计，并使用本文档下一节中描述的 HSE 微调方法之一进行精细调谐，则先前的验证是可靠的。

请注意，用于 HSE 微调电容组的精度可能因产品而异。如果整个生产链需要载波精度验证，请参阅*第 1.7 节*。

1.7 生产中的调谐

重要的一点是要知道，为测试 PCB 定义的微调参数在应用于所有要生产的 PCB 时，是否具有相同的效率。

为了使生产链更快、更高效，并不强制要求在每个 PCB 上运行微调过程。

在 STM32WB Nucleo 板的生产过程中收集的数据表明，对于固定的 HSE 值（HSETUNE 精度），HSE 频率有一些变化。

除此之外，典型的 HSETUNE 粒度为 1 ppm（请参阅 STM32WB 数据手册中的 XOTUNE 粒度）。为了对这些变化进行补偿，建议使用以下微调方法进行生产：

1. 调谐一些 PCB（不超过几十个），使 HSETUNE 值具有明显的跨度
2. 计算找到的值的中位数，使用结果微调所有其他 PCB
3. 找到的值的中位数和最大值/最小值之间的间隔与经调谐设计的 HSETUNE 精度对应。

遵循此过程后，必须考虑由 HSETUNE 精度（第 3 点）导致的潜在频率偏差，并将其添加到*第 1.6 节*中介绍的载波精度验证计算中。

此外，也可以将类似的方法用于 STM32WL 系列。测试板的数量必须更大，因为这些产品采用两个微调参数。

2 微调方法比较

本文档中描述的三种微调方法详见*表 4*，并在*表 5* 中进行了比较。

表 4. 微调方法

方法	说明
手动	精密频率计用于测量其中一个 STM32 引脚上的 HSE 频率输出。然后，用户使用 Nucleo 板的按钮调谐 HSE 频率。一个按钮专用于将调谐参数保存在 STM32 非易失性存储器中。
自动	一个 STM32 定时器由用户通过其中一个 STM32 引脚提供的精密外部时钟源进行时钟控制。该参考时钟使用户可测量 STM32 系统内部 HSE 频率。然后，STM32 可以将测量的频率与预期的频率进行比较，以测试和确定最佳调谐参数。最后，STM32 将这些参数保存在其非易失性存储器中。
STM32CubeMonitor-RF(1)	精密频率计用于测量其中一个 STM32 引脚上的 HSE 频率输出。然后，用户使用在 STM32CubeMonitor-RF 中运行的脚本调谐 HSE 频率。用户必须更改脚本中的调谐参数值，以便对其进行测试。找到正确的值后，由另一个脚本将这些值保存在 STM32 非易失性存储器中。

1. 仅与 STM32WB 系列兼容。

表 5. 微调方法比较

方法	优点	缺点
手动	在 SRAM 中运行（Flash 存储器中的用户程序不受影响）。	每个产品都需要使用按钮和频率计。
自动	在 SRAM 中运行（Flash 存储器中的用户程序不受影响）。 用户只需设置一次参考时钟，即可根据需要微调任意数量的器件。	基于更复杂原理的方法。
STM32CubeMonitor-RF(1)	方便熟悉 STM32CubeMonitor-RF、希望实现最大功能的用户。	必须在器件中刷写 BLE 协议栈和透传模式固件。需要用户对每个产品执行多项操作（修改和运行脚本、使用频率计）。

1. 仅与 STM32WB 系列兼容。

注意： *所提出的方法需要使用引脚（MCO）来输出 HSE 信号，以进行频率测量。如果没有可用于此目的的 MCO 引脚，AN5378（可从 www.st.com 下载）介绍了另一种技术：该技术提出对 STM32WB 射频发射的音调频率进行测量，而不是在 MCO 引脚上进行频率测量。然而，在发出此音调时，无法微调 HSE，因为 HSE 被射频使用（实际上，这更像是一种验证，而不是如本文档所公开的真正的微调方法）。*

3 STM32WB 系列的手动频率微调流程样例

与本文档相关的固件和脚本在 STM32Cube 扩展包（X-CUBE-CLKTRIM_vx.y\Projects\P-NUCLEO-WB55.Nucleo\RCC_HSE_Calib）中提供。

3.1 流程说明

该流程包括测量器件内部从外部晶体产生的 HSE 时钟。该时钟在引脚 PA8 上输出，并由精密频率计测量。

注意： *外部基准为必选项，因为器件中未集成如此精确的基准。*

对负载电容进行逐步调谐，以达到 HSE 时钟的最佳精度。然后，将负载电容值存储在器件的非易失性位置中，即用户 Flash 存储器的专用区域或一次性编程（OTP）区域。

以双字粒度（64 位）完成 Flash 存储器或 OTP 编程。为了节省 OTP 字节（STM32WB 系列为 1 K），可以将 6 位的负载电容值与其他个性化数据（如 Bluetooth®器件地址、MAC 短地址、产品特定代码、密钥）一起附加到 64 位宽结构中。

该流程可以多次执行，只有上一次设置处于活动状态。

流程完成后，可以在启动时（在时钟配置功能中）检索有源负载电容值，并相应地设置 HSE 配置寄存器。

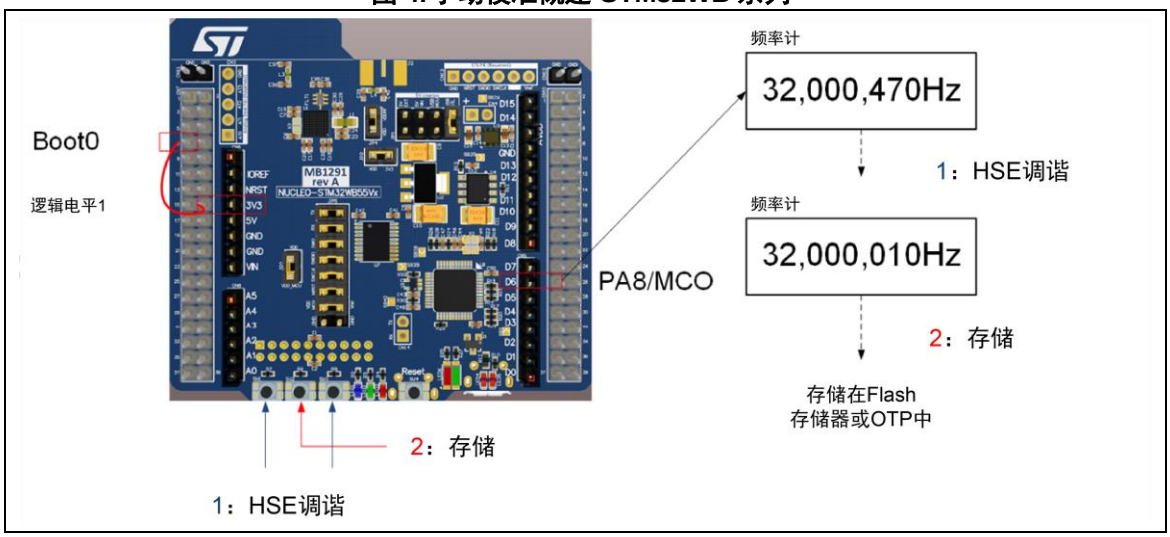
3.2 实现方法

该流程在 SRAM 中执行，因此可以在已编程的器件上运行，而无需修改 Flash 存储器内容。

3.2.1 硬件设置

图 4 展示了 STM32WB 系列 Nucleo-68 板的手动校准流程。

图 4. 手动校准概述-STM32WB 系列



必须将精密频率计（优于 0.1 ppm）连接到引脚 PA8/MCO，并设置成检测 32 MHz 3.3 V 方波峰对峰信号。

注意： *对于这种测量而言，标准示波器不够精确。*

从 SRAM 启动

启动选择可通过 BOOT0 引脚和用户选项（FLASH_OPTR）中的 nBOOT1 位完成。

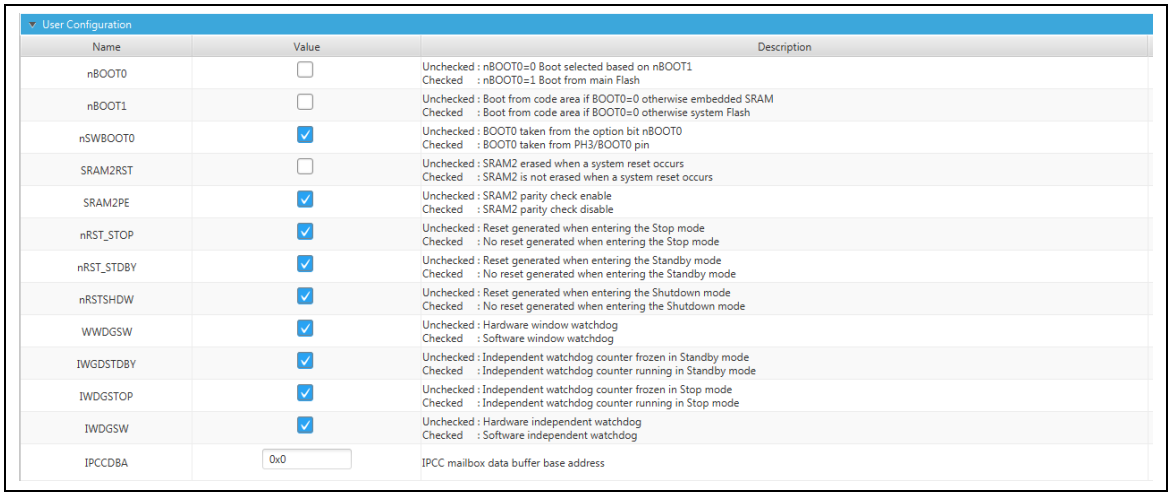
从 SRAM 配置启动通过以下条件配置：Boot0=1 且 nBoot1=0，并且 nBoot1 仅由选项位 FLASH_OPTR[23]设置。

可以通过以下方式选择 Boot0

- 如果选项位 nSWBOOT0=1（FLASH_OPTR[26]=1），则通过启动时引脚 PH3 的值，请参见 *图 5* 中的选项字节面板
- 如果选项位 nSWBOOT0=0（FLASH_OPTR[26]=0），则通过选项位值 nBOOT0，请参见*图 6* 中的选项字节面板

选项位可以通过 STM32CubeProgrammer 选择。

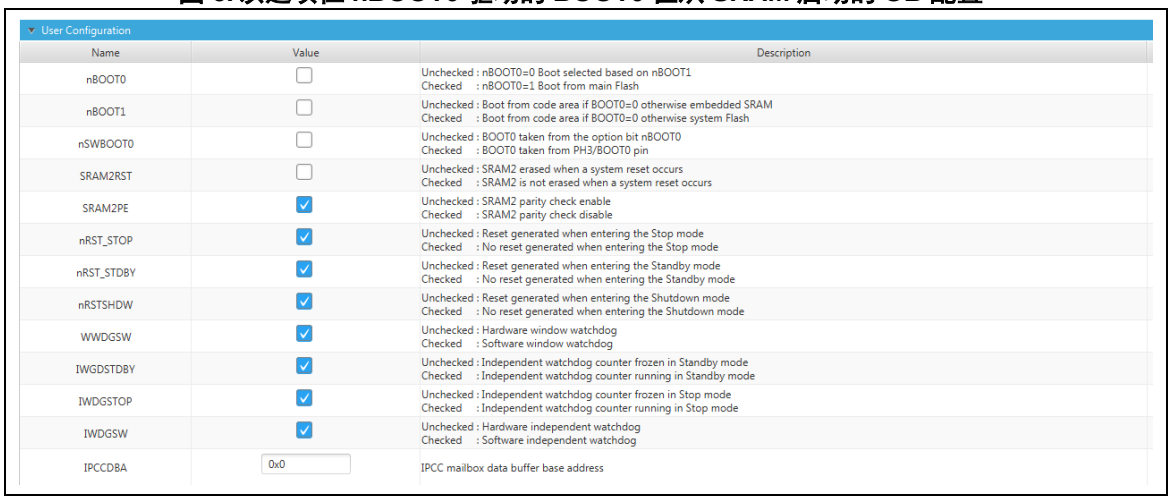
图 5. 以 PH3 引脚驱动的 BOOT0 值从 SRAM 启动的 OB 配置



User Configuration

Name	Value	Description
nBOOT0	☐	Unchecked : nBOOT0=0 Boot selected based on nBOOT1 Checked : nBOOT0=1 Boot from main Flash
nBOOT1	☐	Unchecked : Boot from code area if BOOT0=0 otherwise embedded SRAM Checked : Boot from code area if BOOT0=0 otherwise system Flash
nSWBOOT0	☑	Unchecked : BOOT0 taken from the option bit nBOOT0 Checked : BOOT0 taken from PH3/BOOT0 pin
SRAM2RST	☐	Unchecked : SRAM2 erased when a system reset occurs Checked : SRAM2 is not erased when a system reset occurs
SRAM2PE	☑	Unchecked : SRAM2 parity check enable Checked : SRAM2 parity check disable
nRST_STOP	☑	Unchecked : Reset generated when entering the Stop mode Checked : No reset generated when entering the Stop mode
nRST_STDBY	☑	Unchecked : Reset generated when entering the Standby mode Checked : No reset generated when entering the Standby mode
nRSTSHDW	☑	Unchecked : Reset generated when entering the Shutdown mode Checked : No reset generated when entering the Shutdown mode
WWDGSW	☑	Unchecked : Hardware window watchdog Checked : Software window watchdog
IWGDSTDBY	☑	Unchecked : Independent watchdog counter frozen in Standby mode Checked : Independent watchdog counter running in Standby mode
IWDGSTOP	☑	Unchecked : Independent watchdog counter frozen in Stop mode Checked : Independent watchdog counter running in Stop mode
IWDGSW	☑	Unchecked : Hardware independent watchdog Checked : Software independent watchdog
IPCCDBA	0x0	IPCC mailbox data buffer base address

图 6. 以选项位 nBOOT0 驱动的 BOOT0 值从 SRAM 启动的 OB 配置



User Configuration

Name	Value	Description
nBOOT0	☐	Unchecked : nBOOT0=0 Boot selected based on nBOOT1 Checked : nBOOT0=1 Boot from main Flash
nBOOT1	☐	Unchecked : Boot from code area if BOOT0=0 otherwise embedded SRAM Checked : Boot from code area if BOOT0=0 otherwise system Flash
nSWBOOT0	☐	Unchecked : BOOT0 taken from the option bit nBOOT0 Checked : BOOT0 taken from PH3/BOOT0 pin
SRAM2RST	☐	Unchecked : SRAM2 erased when a system reset occurs Checked : SRAM2 is not erased when a system reset occurs
SRAM2PE	☑	Unchecked : SRAM2 parity check enable Checked : SRAM2 parity check disable
nRST_STOP	☑	Unchecked : Reset generated when entering the Stop mode Checked : No reset generated when entering the Stop mode
nRST_STDBY	☑	Unchecked : Reset generated when entering the Standby mode Checked : No reset generated when entering the Standby mode
nRSTSHDW	☑	Unchecked : Reset generated when entering the Shutdown mode Checked : No reset generated when entering the Shutdown mode
WWDGSW	☑	Unchecked : Hardware window watchdog Checked : Software window watchdog
IWGDSTDBY	☑	Unchecked : Independent watchdog counter frozen in Standby mode Checked : Independent watchdog counter running in Standby mode
IWDGSTOP	☑	Unchecked : Independent watchdog counter frozen in Stop mode Checked : Independent watchdog counter running in Stop mode
IWDGSW	☑	Unchecked : Hardware independent watchdog Checked : Software independent watchdog
IPCCDBA	0x0	IPCC mailbox data buffer base address

时钟输出

HSE 时钟在引脚 PA8（MCO）上输出，该引脚设置于连接器 CN9/D6 和 CN10/25 上。在固件中进行 PA8 配置。频率计探头连接到这些连接器之一，接地可以从连接器 CN11 或 CN12 获取。根据所使用的频率计类型，可能需要 AC 耦合而不是 DC 耦合。

需要下一时序在 MCO 引脚上输出 HSE。

1. 打开 HSE 振荡器：
   a) 设置时钟控制寄存器 RCC_CR[16] = 1
2. 将 PA8 引脚配置为时钟输出功能
   a) 将 GPIO 复用功能选择为 MCO（=0x0）
   b） 将 GPIO 速度选择为极高频率（=0x3）
3. 将 HSE 选择为无分频系数的输出时钟：
   a) 时钟配置寄存器 RCC_CFGR[30:24] = 0b000_0100

负载电容设置

RCC_HSECR[13:8]寄存器驱动负载电容。

根据提出的流程，使用 Nucleo68 板上设置的三个按钮来修改寄存器值：

- 按 SW1 按钮将该值加 1。
- 按 SW2 按钮将其保存在非易失性存储器中。
- 按 SW3 按钮将其减 1。
  – 复位后初始值置为 0，不能增加到 0x3F 以上（最大负载电容），也不能降低到 0x0 以下（最小负载电容）。每次操作后，都必须测量频率。

每个测试值都需要下一时序：

1. 禁用 HSE 时钟：
   – 时钟控制寄存器 RCC_CR[16] = 0
2. 解锁 RCC_HSECR 寄存器：
   – WRITE_REG(RCC_HSECR, 0xCAFECAFE);
3. 用 RCC_HSECR[13:8]写入六位负载电容
4. 打开 HSE 振荡器：
   – 时钟控制寄存器 RCC_CR[16] = 1

在该流程中，寄存器的其他字段保持不变，并保持其初始值：

- HSE 电流控制（HSEGMC=RCC_HSECR[6:4]）设置为 0x3 →电流最大限值 1.13 mA/V。
- HSE 检测放大器阈值（HSES = RCC_HSECR[3]）设置为 0 → HSE 偏置电流系数 1/2

3.2.2 软件实现

项目配置

扩展包中提供了两种项目配置，一种用于校准流程（STM32WBxx_Nucleo_Set_Calibration），另一种用于测试所存储的值（STM32WBxx_Nucleo_Test_Calibration）。在 RF 应用中，给出上一次配置作为实现 HSE 时钟初始化的样例。

固件基于 STM32WB HAL 驱动程序构建。

1. STM32WBxx_Nucleo_Set_Calibration：将器件编程为：
   - 在 PA8 引脚上发送 HSE 时钟。
   - 修改并设置操作按钮 SW1 和 SW3 时的负载电容值。
   - 按下 SW2 按钮时，将负载电容值与 48 位附加数据（蓝牙器件地址）一起存储在 OTP 或所选 Flash 存储器区域中。
2. STM32WBxx_Nucleo_Test_Calibration：测试实际 HSE 设置的配置：
   - 配置 HSE 时钟并在 PA8 引脚上将其输出。
   - 从 OTP/Flash 存储器区域获取负载电容值。
   - 相应地对 RCC_HSECR 寄存器进行编程。

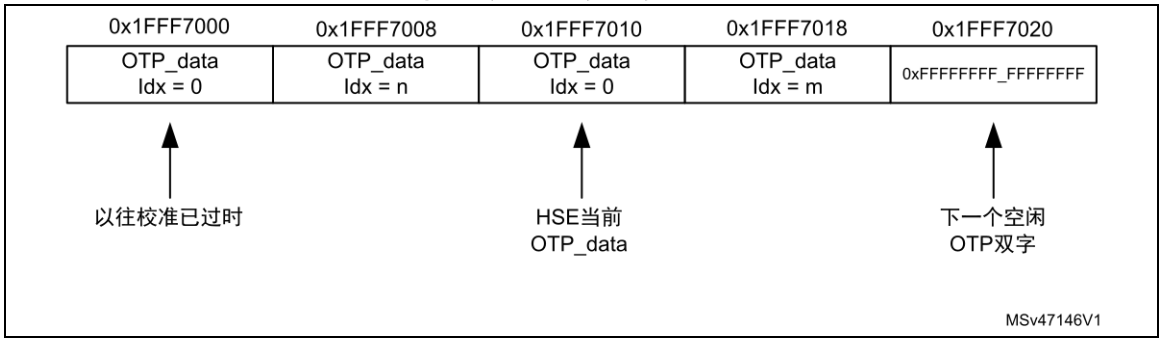
以 OTP（一次性可编程）字节进行存储

负载电容值包含在 64 位结构中。每个 OTP 结构类型都由其索引（一个字节）指示。本应用笔记中用于结构的索引为 0。剩下六个字节用于存储附加数据，如 MAC、蓝牙器件地址或密码密钥。

```
typedef packed struct
    {
      uint8_t    additional _data[6]; /* 48 bits */
      uint8_t    hse_tuning; /* Load capacitance value */
      uint8_t    index; /* structure index ==0x00*/
    } OTP_BT_t;
```

配置阶段通常不会重复，但在存在必须覆写的情况。使用 OTP 字节时，无法删除当前配置，因此新配置将放置在下一个空闲双字间隙中（参见*图 7*）。

图 7. 以 OTP 字节存储配置



校准阶段完成后，应用初始化阶段必须从该 OTP 区域检索负载电容（和其他附加数据）。然后，保留的值是上一个具有正确索引的值。

3.2.3 脚本

提供了两种批处理脚本来运行每个固件配置：

1. STM32WBxx_Nucleo_Set_HSE_Calibration_OTP.bat
2. STM32WBxx_Nucleo_Test_HSE_Calibration_OTP.bat

这些脚本在命令行模式下调用 STM32CubeProgrammer，并且必须相应地设置到工具的路径：SET CLI=“C:\Program Files (x86)\STMicroelectronics\STM32Cube\STM32CubeProgrammer\bin\STM32_Programmer_ CLI.exe.

此外还必须设置配置（例如 Set_Calibration）的二进制或十六进制文件

1. For Keil®：
   SET HEXFILE=“STM32WBxx_Nucleo\STM32WBxx_Nucleo_Set_Calibration.hex”
2. For IAR™：
   SET HEXFILE=“STM32WBxx_Nucleo_Set_Calibration\Exe\STM32WBxx_Nucleo_Set_Calibration.hex”
3. For STM32CubeIDE：
   SET HEXFILE=“STM32WBxx_Nucleo_Set_Calibration\Debug\STM32WBxx_Nucleo_Set_Calibration.hex”

48 位附加数据（上一个参数）通过这些脚本传输到器件。它们存储在 SRAM 内并由固件读取。

```
%CLI% -c port=swd -w32 0x2002FFF0    0x33445566
%CLI% -c port=swd -w32 0x2002FFF4    0x00001122
```

4 STM32WL 系列的手动频率微调流程样例

与本应用笔记相关的固件在 STM32Cube 扩展包（X-CUBE-CLKTRIM_vx.y\Projects\Nucleo-WL55JC\RCC_HSE_Calib_SingleCore）中提供。

注意： *微调的 NUCLEO-WL55JC 板需要配置为使用板载晶体作为 HSE，而不是 TCXO（TCXO 根据温度自主调整其频率）。详情请参考 www.st.com 上的 UM2592。*

4.1 流程说明

该流程包括测量器件内部从外部晶体产生的 HSE 时钟。该时钟在引脚 PA8 上输出，并由精密频率计测量。

注意： *外部基准为必选项，因为器件中不存在这种精确的内部基准。*

对负载电容进行逐步调谐，以达到 HSE 时钟的最佳精度。然后，输入组和输出组负载电容值存储在器件的非易失性位置，即用户 Flash 存储器的专用区域或 OTP 编程区域。

以双字粒度（64 位）完成 Flash 存储器或 OTP 编程。为了节省 OTP 字节（STM32WL 系列为 1 K），可以将输入和输出负载电容值（每个值均为 6 位）与个性化数据（如器件地址、产品特定代码、密钥）一起附加到 64 位宽结构中。

该流程可以多次执行，只有上一次设置处于活动状态。

流程完成后，可以在启动时（在时钟配置功能中）检索有源负载电容值，并相应地设置 HSE 配置寄存器。

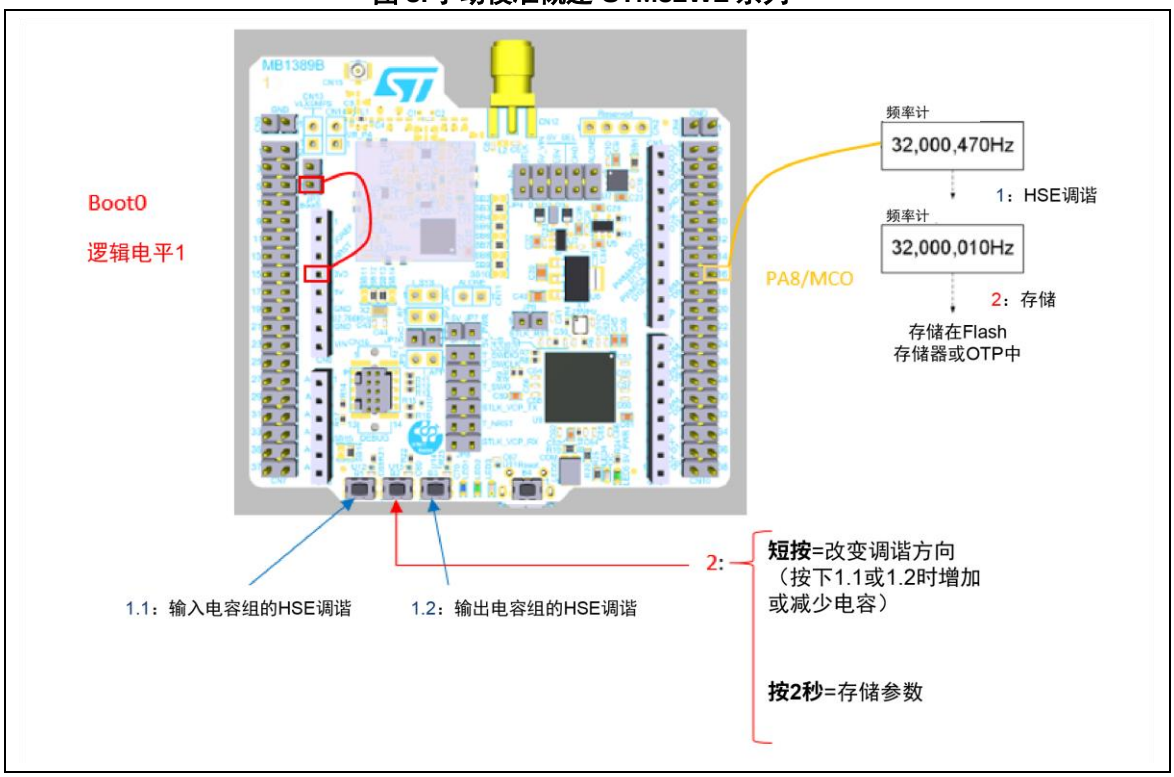
4.2 实现方法

该流程在 SRAM 中运行，因此可以在已编程的器件上执行，而无需修改 Flash 存储器内容。

4.2.1 硬件设置

图 8 展示了 STM32WL 系列 Nucleo 板的手动校准流程。

图 8. 手动校准概述-STM32WL 系列



必须将精密频率计（优于 0.1 ppm）连接到引脚 PA8/MCO，并设置为检测 32 MHz 方波 3.3 V 峰对峰信号。

注意： *对于这种测量而言，标准示波器不够精确。*

从 SRAM 启动

启动选择可通过 BOOT0 引脚和用户选项（FLASH_OPTR）中的 nBOOT1 位完成。

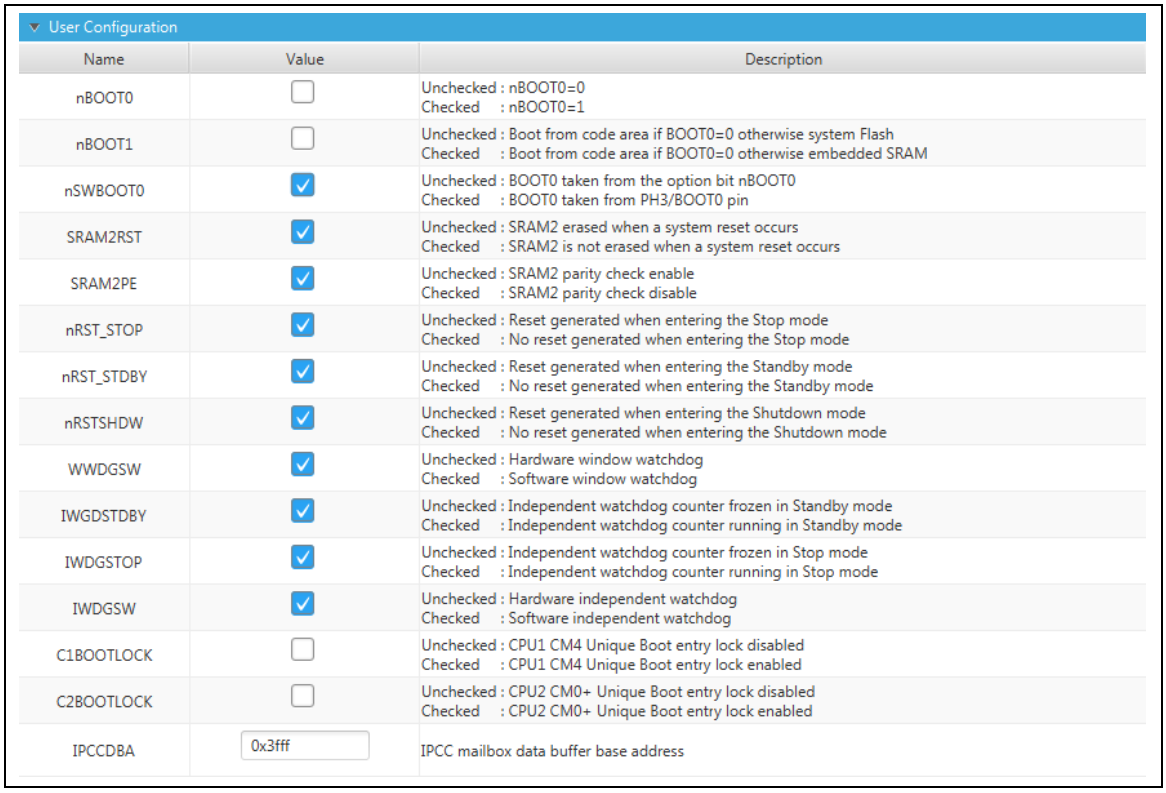
从 SRAM 配置启动通过以下条件配置：BOOT0=1 且 nBOOT1=0，并且 nBOOT1 仅由选项位 FLASH_OPTR[23]设置。

可以通过以下方式选择 BOOT0

- 如果选项位 nSWBOOT0=1（FLASH_OPTR[26]=1），则通过启动时引脚 PH3 的值，请参见 *图 9* 中的选项字节面板
- 如果选项位 nSWBOOT0=0（FLASH_OPTR[26]=0），则通过选项位值 nBOOT0，请参见*图 10* 中的选项字节面板

选项位可以通过 STM32CubeProgrammer 选择。

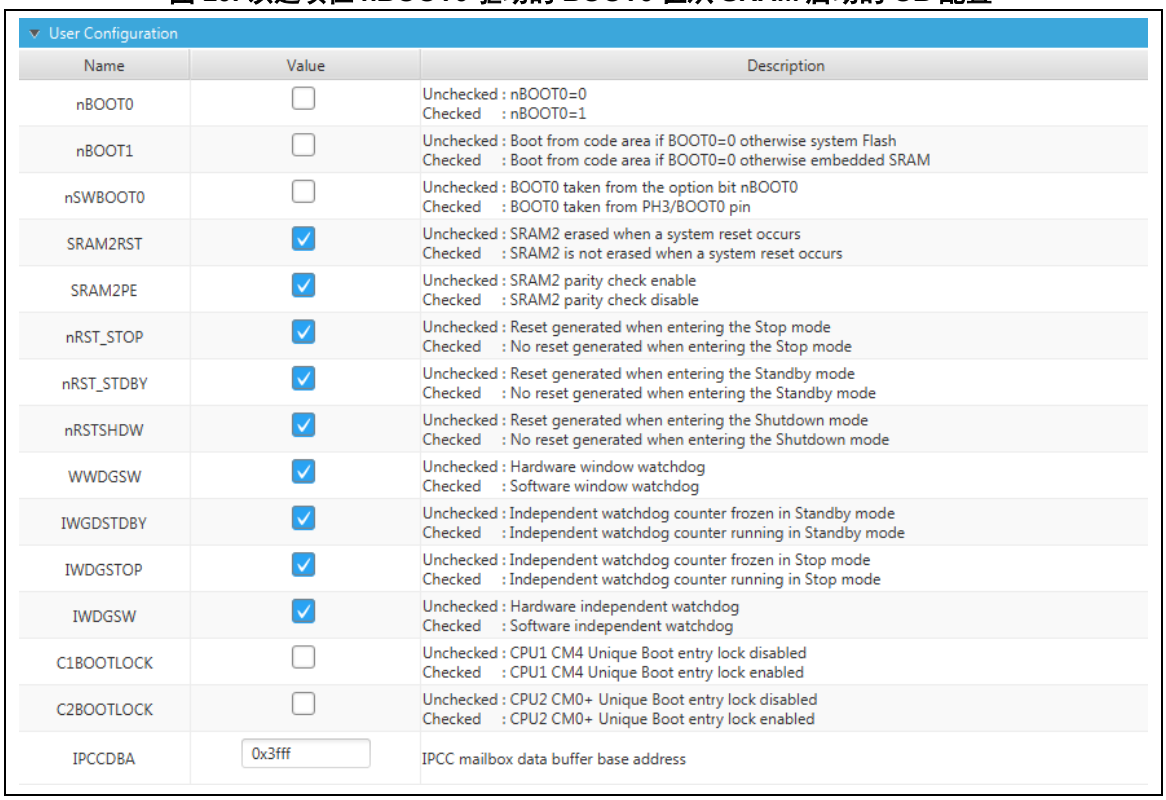
图 9. 以 PH3 引脚驱动的 BOOT0 值从 SRAM 启动的 OB 配置



User Configuration

Name	Value	Description
nBOOT0	☐	Unchecked : nBOOT0=0 Checked : nBOOT0=1
nBOOT1	☐	Unchecked : Boot from code area if BOOT0=0 otherwise system Flash Checked : Boot from code area if BOOT0=0 otherwise embedded SRAM
nSWBOOT0	☑	Unchecked : BOOT0 taken from the option bit nBOOT0 Checked : BOOT0 taken from PH3/BOOT0 pin
SRAM2RST	☑	Unchecked : SRAM2 erased when a system reset occurs Checked : SRAM2 is not erased when a system reset occurs
SRAM2PE	☑	Unchecked : SRAM2 parity check enable Checked : SRAM2 parity check disable
nRST_STOP	☑	Unchecked : Reset generated when entering the Stop mode Checked : No reset generated when entering the Stop mode
nRST_STDBY	☑	Unchecked : Reset generated when entering the Standby mode Checked : No reset generated when entering the Standby mode
nRSTSHDW	☑	Unchecked : Reset generated when entering the Shutdown mode Checked : No reset generated when entering the Shutdown mode
WWDGSW	☑	Unchecked : Hardware window watchdog Checked : Software window watchdog
IWGDSTDBY	☑	Unchecked : Independent watchdog counter frozen in Standby mode Checked : Independent watchdog counter running in Standby mode
IWDGSTOP	☑	Unchecked : Independent watchdog counter frozen in Stop mode Checked : Independent watchdog counter running in Stop mode
IWDGSW	☑	Unchecked : Hardware independent watchdog Checked : Software independent watchdog
C1BOOTLOCK	☐	Unchecked : CPU1 CM4 Unique Boot entry lock disabled Checked : CPU1 CM4 Unique Boot entry lock enabled
C2BOOTLOCK	☐	Unchecked : CPU2 CM0+ Unique Boot entry lock disabled Checked : CPU2 CM0+ Unique Boot entry lock enabled
IPCCDBA	0x3fff	IPCC mailbox data buffer base address

图 10. 以选项位 nBOOT0 驱动的 BOOT0 值从 SRAM 启动的 OB 配置



User Configuration

Name	Value	Description
nBOOT0	☐	Unchecked : nBOOT0=0 Checked : nBOOT0=1
nBOOT1	☐	Unchecked : Boot from code area if BOOT0=0 otherwise system Flash Checked : Boot from code area if BOOT0=0 otherwise embedded SRAM
nSWBOOT0	☐	Unchecked : BOOT0 taken from the option bit nBOOT0 Checked : BOOT0 taken from PH3/BOOT0 pin
SRAM2RST	☑	Unchecked : SRAM2 erased when a system reset occurs Checked : SRAM2 is not erased when a system reset occurs
SRAM2PE	☑	Unchecked : SRAM2 parity check enable Checked : SRAM2 parity check disable
nRST_STOP	☑	Unchecked : Reset generated when entering the Stop mode Checked : No reset generated when entering the Stop mode
nRST_STDBY	☑	Unchecked : Reset generated when entering the Standby mode Checked : No reset generated when entering the Standby mode
nRSTSHDW	☑	Unchecked : Reset generated when entering the Shutdown mode Checked : No reset generated when entering the Shutdown mode
WWDGSW	☑	Unchecked : Hardware window watchdog Checked : Software window watchdog
IWGDSTDBY	☑	Unchecked : Independent watchdog counter frozen in Standby mode Checked : Independent watchdog counter running in Standby mode
IWDGSTOP	☑	Unchecked : Independent watchdog counter frozen in Stop mode Checked : Independent watchdog counter running in Stop mode
IWDGSW	☑	Unchecked : Hardware independent watchdog Checked : Software independent watchdog
C1BOOTLOCK	☐	Unchecked : CPU1 CM4 Unique Boot entry lock disabled Checked : CPU1 CM4 Unique Boot entry lock enabled
C2BOOTLOCK	☐	Unchecked : CPU2 CM0+ Unique Boot entry lock disabled Checked : CPU2 CM0+ Unique Boot entry lock enabled
IPCCDBA	0x3fff	IPCC mailbox data buffer base address

时钟输出

HSE 时钟通过连接器 CN10/16 上设置的引脚 PA8（MCO）输出。由固件执行 PA8 配置。频率计探头连接到该连接器，接地可以从连接器 CN3 或 CN4 获取。根据所使用的频率计类型，可能需要使用 AC 耦合而不是 DC 耦合。

需要下一时序在 MCO 引脚上输出 HSE。

1. 打开 HSE 振荡器：
   a) 设置时钟控制寄存器 RCC_CR[16] = 1
2. 将 PA8 引脚配置为时钟输出功能：
   a) 将 GPIO 复用功能选择为 MCO（=0x0）
   b） 将 GPIO 速度选择为极高频率（=0x3）
3. 将 HSE 选择为无分频系数的输出时钟：
   a) 时钟配置寄存器 RCC_CFGR[30:24] = 0b000_0100

负载电容设置

SUBGHZ_HSEINTRIMR[5:0]位设置输入组的电容。

SUBGHZ_HSEOUTTRIMR[5:0]位设置输出组的电容。

根据提出的流程，使用 Nucleo 板上设置的三个按钮来修改寄存器值：

- 启动后，每次按下 SW1 或 SW3 分别使 SUBGHZ_HSEINTRIMR 和 SUBGHZ_HSEOUTTRIMR 的值加 1。
- 短按一下 SW2 可改变变化方向。例如，如果我们在启动后按下一次 SW2，则每次按下 SW1/SW3 分别使 SUBGHZ_HSEINTRIMR/SUBGHZ_HSEOUTTRIMR 的值减 1。再按一下 SW2 可设置回递增变化方向，另按一下可设置回递减变化方向，依此类推。
- 按下 SW2 两秒（在按下按钮时三个 Nucleo LED 同时闪烁）将参数存储在存储器中（OTP 或 Flash 存储器，取决于用户定义的常数）。

对于这两个寄存器，复位后初始值设置为 0x12（20.3 pF）。不能增加到 0x2F 以上（最大负载电容 33.4 pF），也不能降低到 0x0 以下（最小负载电容 11.3 pF）。每次操作后，必须测量频率。

每个测试值都需要下一时序：

1. 在 HSE32 模式下 sub-GHz 射频进入待机模式：
   – 将不带参数的指令代码 0x80 发送到 sub-GHz 射频 SPI 接口（默认将参数设置为 0）。
2. 将负载电容的六位写入 SUBGHZ_HSEINTRIMR[5:0]或 SUBGHZ_HSEOUTTRIMR[5:0]：
   – 将指令代码 0x0D（写入寄存器指令）发送到 sub-GHz 射频 SPI 接口，然后是寄存器地址（SUBGHZ_HSEINTRIMR 为 0x0911，SUBGHZ_HSEOUTTRIMR 为 0x0912）以及它们必须取的值。
3. sub-GHz 射频进入 FS（频率合成）模式下（以退出待机模式）：
   – 将指令代码 0xC1 发送到不带参数的 sub-GHz 射频 SPI 接口（默认将参数设置为 0）。

4.2.2 软件实现

项目配置

软件包中提供了两种项目配置，一种用于校准流程（被称为 RCC_HSE_Calib_SingleCore 的测定），另一种用于测试存储值（称为 RCC_HSE_Calib_SingleCore_Test）。在 RF 应用中，给出后一次配置作为实现 HSE 时钟初始化的样例。

固件基于 STM32WL HAL 驱动程序构建。

1. RCC_HSE_Calib_SingleCore：将器件编程为：
   a) 在 PA8 引脚上发送 HSE 时钟。
   b） 当操作按钮 SW1 和 SW3 时（短按 SW2 可改变变化方向），修改和设置输入和输出负载电容值。
   c） 当按下 SW2 两秒钟时，将负载电容值与 40 位附加数据（器件地址）一起存储在 OTP 或所选 Flash 存储器区域中。
2. RCC_HSE_Calib_SingleCore_Test，测试保存的 HSE 设置的配置：
   a) 配置 HSE 时钟并在 PA8 引脚上将其输出。
   b） 从 OTP/Flash 存储器区域获取输入和输出负载电容值。
   c） 相应地对 SUBGHZ_HSEINTRIMR 和 SUBGHZ_HSEOUTTRIMR 寄存器进行编程。

以 OTP 字节存储

负载电容值包含在 64 位结构中。每个 OTP 结构类型都由其索引（一个字节）指示。本应用笔记中用于结构的索引为 0。剩余五个字节用来存储附加数据，如器件地址或密码密钥。

```
typedef packed struct
{
uint8_t additional _data[5]; /* 48 bits */
uint8_t hse_tuning_in; /* IN bank load capacitance value */
uint8_t hse_tuning_out; /* OUT bank load capacitance value */
uint8_t index; /* structure index ==0x00*/
} OTP_BT_t
```

即使不应重复配置阶段，但也可能存在应对其进行覆写的情况。使用 OTP 字节时，无法删除当前配置，因此新配置将放置在下一个空闲的双字间隙中（参见*图 7*）。

校准阶段完成后，应用初始化阶段必须从该 OTP 区域检索输入和输出负载电容（以及其他附加数据）。保留的值是上次具有正确索引的值。

4.2.3 脚本

提供了两种批处理脚本来运行每个固件配置：

- RCC_HSE_Calib_SingleCore_OTP.bat
- RCC_HSE_Calib_SingleCore_Test_OTP.bat

这些脚本调用 STM32CubeProgrammer 命令行接口，并且必须相应地设置到工具的路径：

SET CLI=“C:\Program Files (x86)
\STMicroelectronics\STM32Cube\STM32CubeProgrammer\bin\STM32_Programmer_CLI.exe”

此外还必须设置配置的二进制或十六进制文件

对于 IAR™：

SET BINFILE=“RCC_HSE_Calib_SingleCore\Exe\RCC_HSE_Calib_SingleCore.bin”

For Keil®：

SET HEXFILE=“RCC_HSE_Calib_SingleCore\Exe\RCC_HSE_Calib_SingleCore.hex”

40 位附加数据（上一个参数）通过这些脚本传输到器件。它们存储在 SRAM 内并由固件读取。

STM32_Programmer_CLI.exe -c port=swd -w32 0x20006FF0 0x22334455
STM32_Programmer_CLI.exe -c port=swd -w32 0x20006FF4 0x00000011

5 STM32WB 系列自动频率微调流程样例

与本文档相关的固件和脚本在 STM32Cube 扩展包（X-CUBE-CLKTRIM_vx.y\Projects\P-NUCLEO-WB55.Nucleo\RCC_HSE_AutoCalib）中提供。

5.1 流程说明

该流程包括测量器件内部从外部晶体产生的 HSE 时钟。测量是在 STM32 系统内部完成，并且基于用户在 PA9 上提供的精确的外部 16 MHz 时钟参考。

注意： *外部基准为必选项，因为器件中未集成如此精确的基准。*

由特定的 STM32 应用对负载电容进行自动调谐，以达到 HSE 时钟的最佳精度。然后，将找到的负载电容值存储在器件的非易失性位置、用户 Flash 存储器或 OTP 的专用区域中。

以双字粒度（64 位）完成 Flash 存储器或 OTP 编程。为了节省 OTP 字节（STM32WB 系列为 1 K），可以将 6 位负载电容值与其他个性化数据（如蓝牙器件地址、MAC 短地址、产品特定代码、密钥）一起附加到 64 位宽结构中。

该流程可以多次执行，只有最新的设置是有效的。

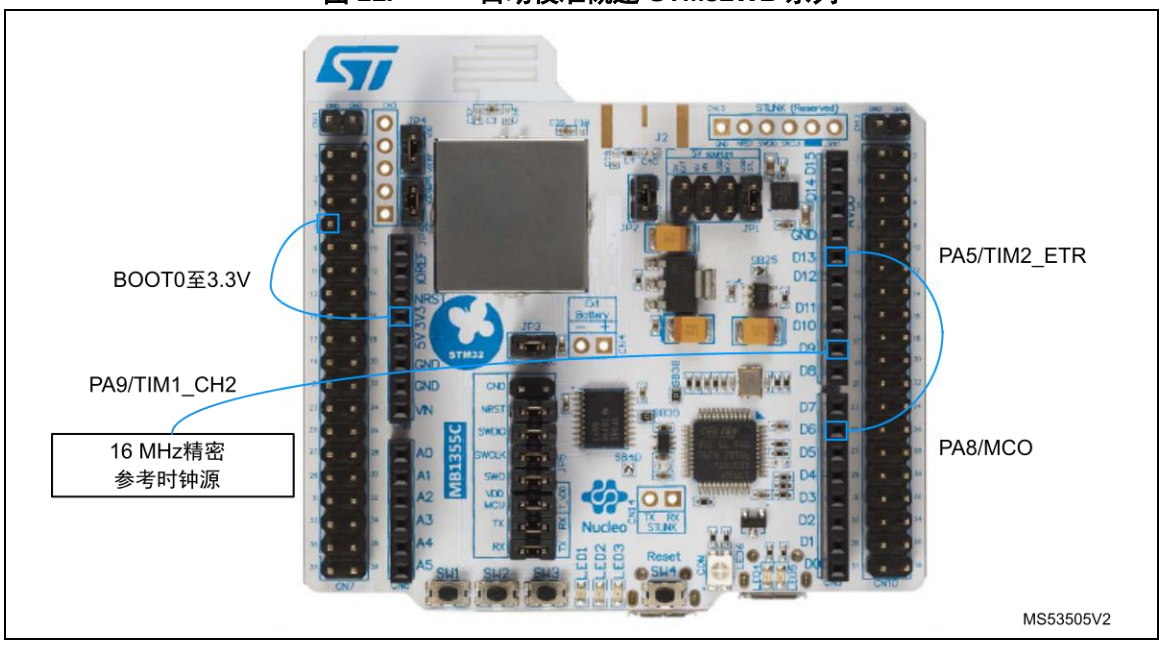
流程完成后，可以在启动时（在时钟配置功能中）检索有效负载电容值，并相应地设置 HSE 配置寄存器。

5.2 实现方法

5.2.1 硬件设置

图 11 展示了 STM32WB 系列 Nucleo-68 板的自动校准流程。

图 11. 自动校准概述-STM32WB 系列



精密时钟发生器（优于 0.1 ppm）必须连接到引脚 PA9/TIM1_CH2，并设置为产生 16 MHz、3.3 VPP 方波。要了解如何将板配置成从 SRAM 启动，请参阅*第 3.2.1 节：硬件设置*。

注意： *标准信号发生器无法为该应用提供足够的精度。*

硬件连接原理

MCU 使用两个定时器（TIMER1 和 TIMER2）来利用外部参考时钟，从而对 HSE 频率进行测量和调谐。

TIMER1 由 16 MHz 精密外部参考时钟进行时钟控制，并在 OFF 时间（固定为 16 ms）期间产生低电平 PWM 信号，在 REF 时间（100 或 1000 ms，取决于校准程序的阶段）期间产生高电平 PWM 信号。

该 PWM 信号用于在高电平时启用 TIMER2 计数器。

TIMER2 由 MCO 引脚上的 HSE/2 输出进行时钟控制。

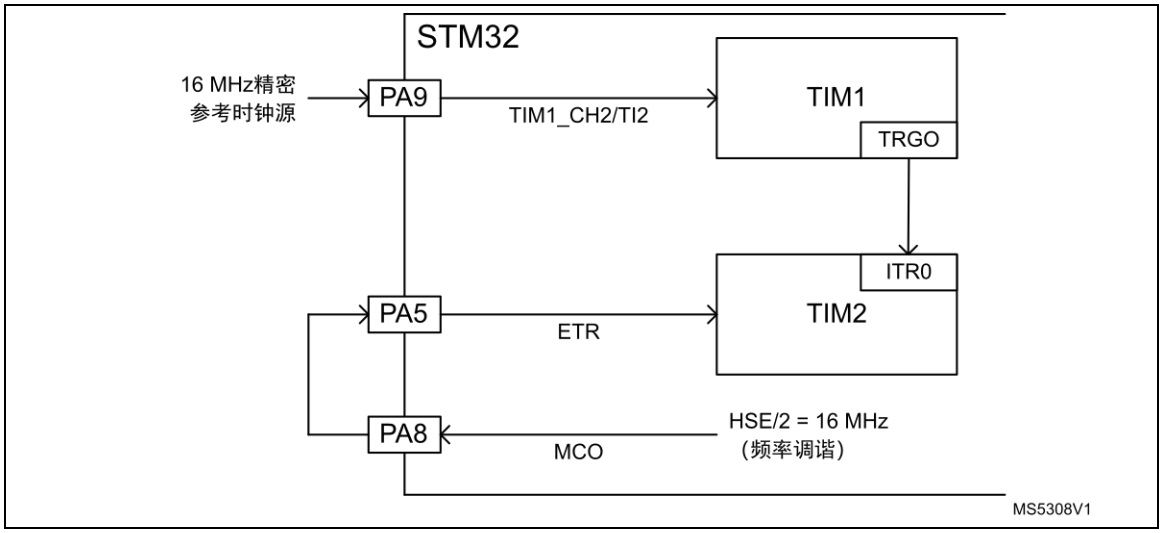
然后，在一个 PWM 信号周期之后，TIMER2 计数器对应于 REF 时间期间经过的 HSE/2 周期数。

可以将该值与预期值进行比较，以调整调节 HSE 频率的负载电容。

注意： *当依次进行多次 HSE 测量时，需要 OFF 时间来延迟并等待 HSE 稳定。*

图 12 展示了流程中涉及的元件之间的连接。

图 12. 流程实现



TIMER1 配置

TIM1 配置为由 PA9 上提供的 16 MHz 外部信号进行时钟控制。PA9 与 TIM1 CH2 相关联，后者提供 TI2（触发输入 2）输入。因此，必须选择 TI2 作为 TIM1 时钟源。

- TIM1_SMCR 寄存器的 SMS 位=b0111（外部时钟模式 1）
- TIM1_SMCR 寄存器的 TS 位=b00110（触发输入=TI2，还必须设置输入的滤波器和极性）。

TIM1 必须在 OFF 时间期间产生低电平 PWM 信号，并在 REF 时间期间产生高电平 PWM 信号。

为此，TIM1 CH3 配置为 PWM 模式 2（低电平周期，然后是高电平周期），预分频器为 16000，以获得 1 ms 的粒度（APB2 频率配置为 16 MHz 和 16 MHz/16000=1000 Hz）。周期设置为 REF 时间 + OFF 时间 - 1，并且脉冲持续时间设置为 OFF 时间，以获得所需的信号（REF 和 OFF 时间以毫秒为单位）。

- TIM1_ARR 寄存器=0x3E7F（预分频器=16000）
- TIM1_PSC 寄存器=REF 时间 + OFF 时间 - 1（周期）
- TIM1_CCMR2 寄存器的 OC3M 位=b111（PWM 模式 2）
- TIM1_CCR3 寄存器=OFF 时间（通道 3 的脉冲）

最后，TIM1 必须设置为主模式 OC3REF，以便其 TRGO（触发输出）对通道 3 产生的 PWM 信号进行中继。

- TIM1_CR2 寄存器的 MMS 位=b110（OC3REF 主模式）

TIMER2 配置

TIM2 必须由 HSE/2 进行时钟控制。

为此，HSE/2 输出在 MCO 引脚 PA8 上：

- RCC_CFGR[30:24]=b001_0100（HSE 作为 MCO 输出，分频系数为 2）。

PA8 从外部链接到 PA5，对应于 TIM2 ETR（外部触发）输入。

然后，必须选择外部时钟模式 2（计数器由 ETRF 信号上的任何有源边沿进行时钟控制）作为 TIM2 时钟源。

- TIM2_SMCR 中的 ECE 位=b1

TIM2 CH2 配置为简单的时基来对 HSE/2 周期进行计数。时基周期设置为其最大值 0xFFFFFFFF，以确保 HSE/2 测量不会溢出。

- TIM2_ARR 寄存器=0（预分频器=0）
- TIM2_PSC 寄存器=0xFFFFFFFF（周期）

最后，TIM2 必须设置为 GATED 从模式，ITR0 作为触发输入。由于 TIM2 ITR0 和 TIM1 TRGO 已连接，因此 GATED 模式仅在 TRGO 上显示的 TIM1 CH3 PWM 信号为高电平时，才会启用 TIM2 CH2 计数器。

- TIM2_SMCR 寄存器中的 SMS 位=b0101（GATED 模式）
- TIM2_SMCR 寄存器中的 TS 位= b00000（ITR0 作为触发输入）

5.2.2 软件实现

固件调谐算法原理

通过以上介绍的 TIMER 链路，可采用外部参考时钟精度，对 REF 时间期间经过的 HSE/2 周期数进行计数。

随该 AN 提供的固件的 HSE_Measurement 函数可通过启动一个 TIM1 CH3 PWM 循环并返回 TIM2 CH2 计数器的值，来获取该周期数。

HSE_Tuning 函数利用 HSE_Measurement 来调谐 HSE，使用二分法逻辑找到 HSE 的最接近 32 MHz 的最佳负载电容值。

设置负载电容的调谐参数可以在 0 到 63 之间取值。因此，该函数首先将其设置为 32（RCC_HSECR[13:8]=HSETUNE=32）。然后，进行 HSE/2 测量。

如果测得的频率（根据 HSE_Measurement 返回的周期数算得）高于 16 MHz，则该函数会增加 HSETUNE 以降低频率。如果低于 16 MHz，则降低，以增加频率。

当 HSETUNE 为 32 时，使用从值 16 开始的步长完成 HSETUNE 的改变。每次测量后，将该步长除以 2，以计算待测的新 HSETUNE 值。

到目前为止，测量持续 100 ms。当步长值达到 1 并且负载电容接近最佳值时，持续时间设置为 1000 ms，从而提高精确度。

当步长达到 1 时，如果测得的 HSE/2 频率低于 16 MHz，则该函数递减 HSETUNE，并进行新的测量和调整，直到测得的频率超过 16 MHz。然后，该函数使用最佳 HSE 精度，在 16 MHz 以上的值和最后一个低于 16 MHz 的值之间取 HSETUNE 的值。

如果上一次 100 ms 测量（步长达到 1 之前的最后一次测量）高于 16 MHz，则推理相同。

最后，将找到的 HSETUNE 值保存在非易失性存储器中。

在该流程中，RCC_HSECR 寄存器的其他字段保持不变，并保持其初始值：

- HSE 电流控制（HSEGMC=RCC_HSECR[6:4]）设置为 0x3 →电流最大限值 1.13 mA/V
- HSE 检测放大器阈值（HSES = RCC_HSECR[3]）设置为 0 → HSE 偏置电流系数 1/2

固件配置

通过 hse_trimming.h 中的常数 STORE_ADDRESS，用户可选择将 HSETUNE 保存在 OTP（STORE_ADDRESS=0x1FFF7000U）或 Flash 存储器（例如，STORE_ADDRESS=0x080A0000）中。

扩展包中提供了两种项目配置，一种（RCC_HSE_AutoCalib）用于校准流程，另一种（RCC_HSE_AutoCalib_Test）用于测试存储的值。在 RF 应用中，给出最后一次配置作为实现 HSE 时钟初始化的样例。

注意： *在 STM32CubeIDE 下，两个项目分别本称为 STM32WBxx_Nucleo_Set_AutoCalibration 和 STM32WBxx_Nucleo_Test_AutoCalibration。*

固件基于 STM32WB HAL 驱动程序构建。

- RCC_HSE_AutoCalib：将器件编程为：
  - 在 PA8 引脚上发送 HSE 时钟，配置定时器并利用外部时钟源
  - 按照前面介绍的微调算法修改并设置负载电容值
  - 将负载电容值与 48 位附加数据（蓝牙器件地址）一起存储在 OTP 存储器或所选 Flash 存储器区域中。
- RCC_HSE_AutoCalib_Test：将器件编程为对实际 HSE 设置进行测试：
  - 配置 HSE 时钟并在 PA8 引脚上将其输出
  - 从 OTP/Flash 存储器区域获取负载电容值
  - 相应地对 RCC_HSECR 寄存器进行编程。

以 OTP 字节存储

在 OTP 存储器中存储调谐参数的方式与 STM32WB 系列的手动微调流程相同（参见*第 3.2.2 节：软件实现*）。

5.2.3 脚本

提供了两种批处理脚本来运行每个固件配置：

1. STM32WBxx_Nucleo_Set_HSE_AutoCalibration_OTP.bat
2. STM32WBxx_Nucleo_Test_HSE_AutoCalibration_OTP.bat

这些脚本在命令行模式下调用 STM32CubeProgrammer，并且必须相应地设置到工具的路径：
SET CLI=“C:\Program Files (x86)\STMicroelectronics\STM32Cube\
STM32CubeProgrammer\bin\STM32_Programmer_ CLI.exe”

还必须设置配置（例如 Set Calibration 应用）的二进制或十六进制文件

- 用于 IAR™：SET HEXFILE=“RCC_HSE_AutoCalib \Exe\RCC_HSE_AutoCalib.hex”
- 用于 Keil®：SET HEXFILE=“RCC_HSE_AutoCalib\ RCC_HSE_AutoCalib.hex”
- 对于 STM32CubeIDE：
  SET HEXFILE=“STM32WBxx_Nucleo_Set_AutoCalibration\
  Debug\STM32WBxx_Nucleo_Set_AutoCalibration.hex”

48 位附加数据（上一个参数）通过这些脚本传输到器件。它们存储在 SRAM 内并由固件读取。

```
%CLI% -c port=swd -w32 0x2002FFF0    0x33445566
%CLI% -c port=swd -w32 0x2002FFF4    0x00001122
```

6 STM32WL 系列自动频率微调流程样例

与本文档相关的固件在 STM32Cube 扩展包（X-CUBE-CLKTRIM_vx.y\Projects\NUCLEO-WL55JC\RCC_HSE_AutoCalib_SingleCore）中提供。

注意： *微调的 NUCLEO-WL55JC 板需要配置为使用板载晶体作为 HSE，而不是 TCXO（TCXO 根据温度自主调整其频率）。详情请参考 www.st.com 上的 UM2592。*

6.1 流程说明

该流程包括测量器件内部从外部晶体产生的 HSE 时钟。测量是在 STM32 系统内部完成，并且基于用户在 PA9 上提供的精确的外部 16 MHz 时钟参考。

注意： *外部基准为必选项，因为器件中不存在这种精确的内部基准。*

由特定的 STM32 应用对负载电容进行自动调谐，以达到 HSE 时钟的最佳精度。然后，负载电容值存储在器件的非易失性位置（用户 Flash 存储器的专用区域或 OTP 编程区域）。

以双字粒度（64 位）完成 Flash 存储器或 OTP 内存。为了节省 OTP 字节（STM32WL 系列为 1 K），可以将输入和输出负载电容值（每个值均为 6 位）与个性化数据（如器件地址、产品特定代码、密钥）一起附加到 64 位宽结构中。

该流程可以重复多次，只有最后一个设置是有效的。

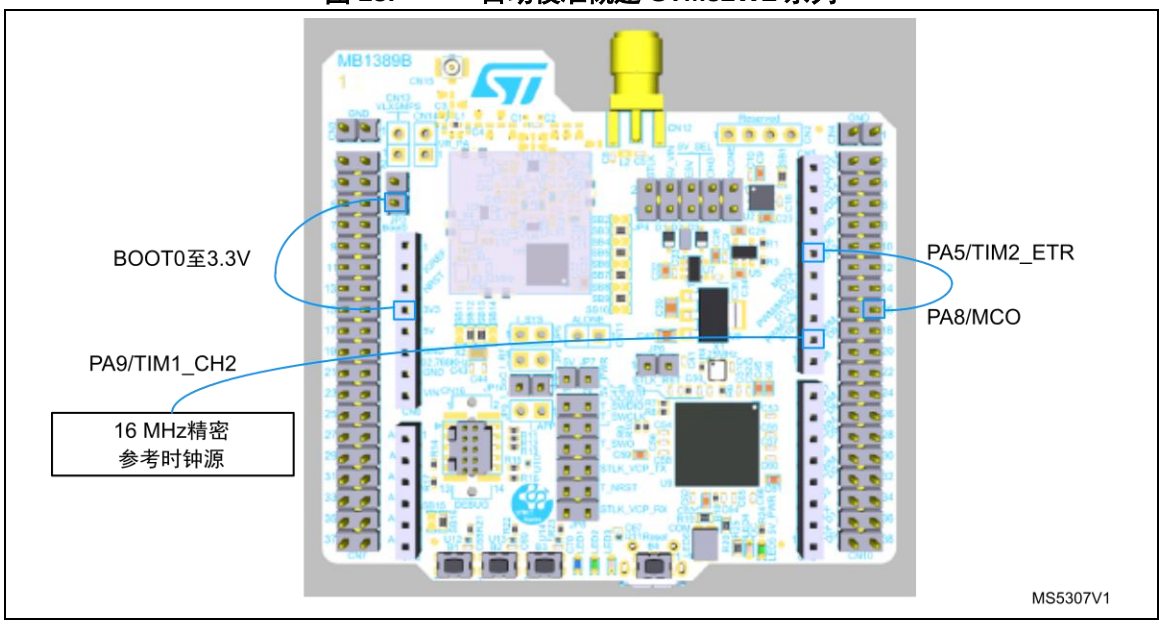
流程完成后，可以在启动时（在时钟配置功能中）检索有源负载电容值，并相应地设置 HSE 配置寄存器。

6.2 实现方法

6.2.1 硬件设置

图 13 展示了 STM32WL 系列 Nucleo 板的自动 HSE 校准流程。

图 13. 自动校准概述-STM32WL 系列



精密时钟发生器（优于 0.1 ppm）必须连接到引脚 PA9/TIM1_CH2，并设置为产生 16 MHz、3.3 VPP 方波。要了解如何将板配置成从 SRAM 启动，请参阅*第 4.2.1 节：硬件设置*。

注意： *标准信号发生器无法为该应用提供足够的精度。*

硬件连接原理

建立连接和实现 TIMER1 和 TIMER2 的方式与用于 STM32WB 系列的自动微调流程的方式相同（参见*第 5.2.1 节*）。

6.2.2 软件实现

固件调谐算法原理

通过*第 5.2.1 节*中介绍的 TIMER 链路，可采用外部参考时钟精度，对 REF 时间期间经过的 HSE/2 周期数进行计数。

随本文档提供的 HSE_Measurement 函数可通过启动一个 TIM1 CH3 PWM 循环并返回 TIM2 CH2 计数器的值，来获取该周期数。

HSE_Tuning 函数利用 HSE_Measurement 对 HSE 进行调谐。

其中遵循二分法逻辑找到 HSE 的最接近 32 MHz 的最佳负载电容值。

设置输入和输出负载电容的调谐参数可以在 0 到 47 之间取值。因此，该函数首先将其均设置为 24（SUBGHZ_HSEINTRIMR[5:0]=24 且 SUBGHZ_HSEOUTTRIMR[5:0]=24）。然后，进行 HSE/2 测量。

如果测得的频率（根据 HSE_Measurement 返回的周期数算得）高于 16 MHz，则该函数会增加 SUBGHZ_HSEINTRIMR 以降低频率。如果低于 16 MHz，则降低，以增加频率。

当 SUBGHZ_HSEINTRIMR 为 24 时，使用从值 12 开始的步长完成 SUBGHZ_HSEINTRIMR 的改变。每次测量后，该步长除以 2 来计算要测试的新的 SUBGHZ_HSEINTRIMR 值。

对于每次测量，更改调谐的参数。也就是说，进行第一次测量并调整 SUBGHZ_HSEINTRIMR，进行另一次测量并调整 SUBGHZ_HSEOUTTRIMR（并且遵循以上针对 SUBGHZ_HSEINTRIMR 说明的调整原理），然后进行又一次测量并再次调整 SUBGHZ_HSEINTRIMR，依此类推。

到目前为止，测量持续 100 ms。当两个参数之一的步长值达到 1（负载电容接近最佳值）时，持续时间设置为 1000 ms，从而提高精确度。

当步长达到 1 时，如果测得的 HSE/2 频率低于 16 MHz，则该函数递减 SUBGHZ_HSEINTRIMR 和 SUBGHZ_HSEOUTTRIMR，并进行新的测量和调整，直到测得的频率超过 16 MHz。一旦超过 16 MHz，该函数将 SUBGHZ_HSEINTRIMR 和 SUBGHZ_HSEOUTTRIMR 对保持在获得最佳 HSE 精确度时的数值以上和最后一个在该数值以下的数值之间。

如果最后一次 100 ms 测量（步长达到 1 之前的最后一次测量）高于 16 MHz，则理由相同。

最后，将找到的 SUBGHZ_HSEINTRIMR 和 SUBGHZ_HSEOUTTRIMR 对保存在非易失性存储器中。

固件配置

通过 hse_trimming.h 中的常数 STORE_ADDRESS，用户可选择将 SUBGHZ_HSEINTRIMR 和 SUBGHZ_HSEOUTTRIMR 对保存在 OTP（STORE_ADDRESS=0x1FFF7000U）或 Flash 存储器（例如，STORE_ADDRESS=0x0803F800）中。

扩展包中还提供两种项目配置，一种用于校准流程（RCC_HSE_AutoCalib_SingleCore），另一种用于测试所存储的值（RCC_HSE_AutoCalib_SingleCore_Test）。在 RF 应用中，给出最后一次配置作为实现 HSE 时钟初始化的样例。

固件基于 STM32WL HAL 驱动程序构建。

- RCC_HSE_AutoCalib_SingleCore：将器件编程为
  - 在 PA8 引脚上发送 HSE 时钟，配置定时器并利用外部时钟源
  - 按照前面介绍的微调算法修改并设置输入和输出负载电容值
  - 将输入和输出负载电容值与 40 位附加数据一起存储在 OTP 或所选 Flash 存储器区域中。
- RCC_HSE_AutoCalib_SingleCore_Test，用于测试所保存的实际 HSE 设置：
  - 配置 HSE 时钟并在 PA8 引脚上将其输出
  - 从 OTP/Flash 存储器区域获取负载电容值
  - 相应地对 SUBGHZ_HSEINTRIMR 和 SUBGHZ_ HSEOUTTRIMR 寄存器进行编程

以 OTP 存储

在 OTP 存储器中存储调谐参数的方式与用于 STM32WL 系列的手动微调流程的方式相同（参见*第 4.2.2 节*）。

6.2.3 脚本

提供了两种批处理脚本来运行每个固件配置：

- RCC_HSE_AutoCalib_SingleCore_OTP.bat
- RCC_HSE_AutoCalib_SingleCore_Test_OTP.bat

这些脚本调用 STM32CubeProgrammer 命令行接口，并且必须相应地设置到工具的路径：

SET CLI=“C:\Program Files (x86)
\STMicroelectronics\STM32Cube\STM32CubeProgrammer\bin\STM32_Programmer_CLI.exe”

还必须设置配置的二进制或十六进制文件

- 用于 IAR：SET
  BINFILE=“RCC_HSE_AutoCalib_SingleCore\Exe\RCC_HSE_AutoCalib_SingleCore.bin”
- 用于 Keil：SET
  HEXFILE=“RCC_HSE_AutoCalib_SingleCore\Exe\RCC_HSE_AutoCalib_SingleCore.hex”

40 位附加数据通过这些脚本传输到器件。它们存储在 SRAM 内并由固件读取。

STM32_Programmer_CLI.exe -c port=swd -w32 0x20006FF0 0x22334455

STM32_Programmer_CLI.exe -c port=swd -w32 0x20006FF4 0x00000011

7 STM32WB 系列 STM32CubeMonitor-RF 频率微调流程样例

与本文档相关的固件和脚本在 STM32Cube 扩展包（X-CUBE-CLKTRIM_vx.y\Projects\P-NUCLEO-WB55.Nucleo\RCC_HSE_MonitorRFCalib）中提供。

注意： *STM32CubeMonitor-RF 流程专为 STM32WB 系列而开发，因为该软件不支持 STM32WL 系列。*

7.1 流程说明

该流程与*第 3.1 节：流程说明*中详细说明的流程相同。

7.2 流程步骤

*第 7.3.2 节*中提供了这些步骤的详细信息。

该流程使用 STM32CubeMonitor-RF 对 STM32WB HSE 时钟进行调谐。因此，必须建立两个实体间的连接。

要实现该连接，必须在 STM32WB 内核中刷写两个软件：

- Cortex® M4 内核中的 BLE 透传模式固件
- Cortex® M0+内核中的 BLE 协议栈。

刷写完成后，将 STM32WB 连接到运行 STM32CubeMonitor-RF 的电脑。

转到 STM32CubeMonitor-RF 中的脚本部分。

首先运行 mco_output_config.txt，以在引脚 PA8 上输出 HSE。

使用 HSETUNE 调谐参数修改 hse_tunning.txt 进行测试，然后运行 hse_tunning.txt。

测量 PA8 上的 HSE 频率，再次运行前，根据需要调整 hse_tunning.txt 中的调谐参数。

找到符合要求的参数后，将其插入 trim_param_flash.txt 或 trim_param_otp.txt，具体取决于保存该参数的存储器（Flash 存储器或 OTP）。然后运行所选脚本。

若要保存在 Flash 存储器中，可能必须使用 erase_flash_page.txt 脚本擦除所选页面（只能在页面级别执行擦除操作）。

最后，将调谐参数保存在非易失性存储器中。

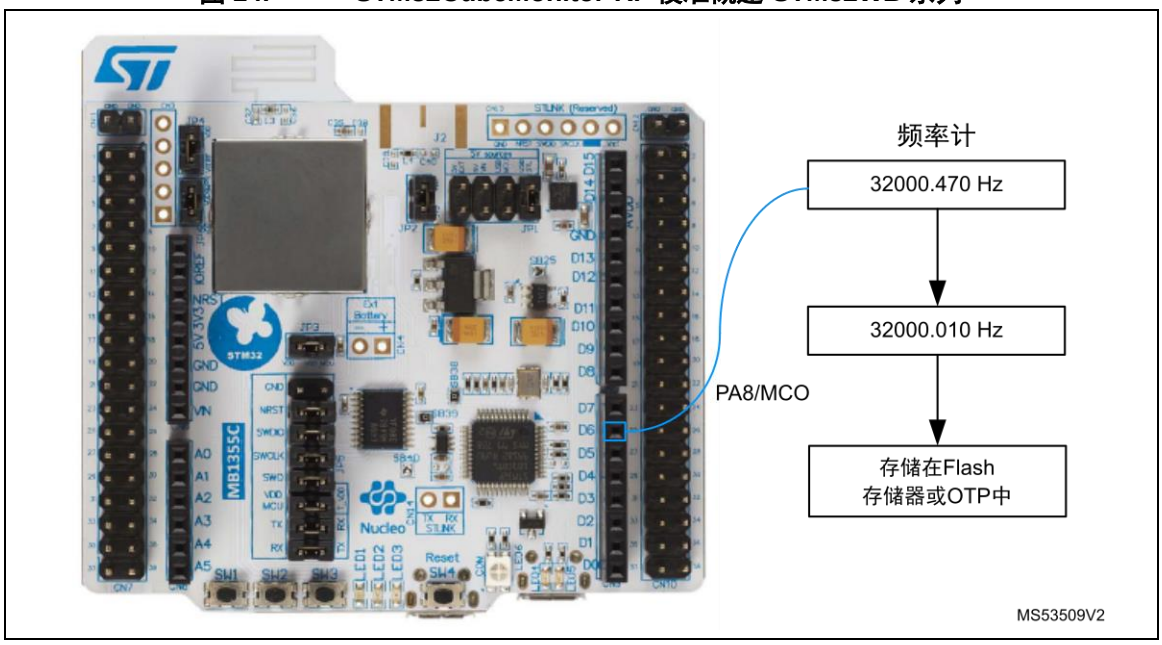
C 代码 retrieve_trimming_values 提供了检索保存在存储器中的参数的样例。

7.3 实现方法

7.3.1 硬件设置

图 14 展示了 STM32WB 系列 Nucleo-68 板的 STM32CubeMonitor-RF HSE 校准流程。

图 14. STM32CubeMonitor-RF 校准概述-STM32WB 系列



必须将精密频率计（优于 0.1 ppm）连接到引脚 PA8/MCO，并设置成检测 32 MHz 3.3 V 方波峰对峰信号。

注意： *对于这种测量而言，标准示波器不够精确。*

7.3.2 软件和脚本设置

在 Cortex® M4 内核中刷写透传固件

BLE 透传模式固件在 STM32Cube_FW_WB 包中提供：

- STM32Cube_FW_WB_Vx.y.z\Projects\P-NUCLEO-WB55.Nucleo\Applications\BLE\BLE_TransparentMode

该项目可以在用户喜爱的 IDE 中打开，而且必须在 Cortex® M4 内核上运行。

在 Cortex® M0+内核中刷写 BLE 协议栈

BLE 协议栈在 STM32Cube_FW_WB 包中提供：

- STM32Cube_FW_WB_Vx.y.z\Projects\STM32WB_Copro_Wireless_Binaries\stm32wb5x_BLE_Stack_fw.bin

版本说明文件中提供了将该协议栈刷写到 Cortex® M0+ 内核的说明：STM32Cube_FW_WB_Vx.y.z\Projects\STM32WB_Copro_Wireless_Binaries\Release_Notes.html

注意： *有关透传模式、BLE 协议栈和 STM32CubeMonitor-RF 关系的更多信息，请参阅 UM2288“STM32CubeMonitor-RF 无线性能测量软件工具”（可从 www.st.com 下载）。*

将产品连接到 STM32CubeMonitor-RF

如果使用 Nucleo 板，只需通过 USB 将其连接到电脑。如果没有使用，则用户可以通过 MCU 的 USART 接口与透传模式固件通信，该接口在引脚 PB6/RX 和 PB7/TX 上提供。

注意： *PB6 和 PB7 由 Nucleo 板上的 ST-Link VCP（虚拟 COM 端口）使用。*

7.3.3 脚本

注意： *下列脚本在 STM32Cube 扩展包（X-CUBE-CLKTRIM_vx.y\Projects\P-NUCLEO-WB55.Nucleo\RCC_HSE_MonitorRFCalib）中提供。*

mco_output_config.txt

无需修改，将其在 STM32CubeMonitor-RF 中运行，即可在 PA8 上输出 HSE。

hse_tuning.txt

该脚本包括要测试的 HSETUNE 值。该值是以下脚本行（第 11 行）中的最后一个参数，在该样例中被设置为 0x18。

Send(VS_HCI_C1_WRITE_REGISTER;0x04;0x0000FF00;0x5800009C;0x00001800)

该行更新了 RCC_HSECR 寄存器的 HSETUNE 位。

如要测试 HSETUNE 调谐参数，应根据需要更改该值，并在 STM32CubeMonitor-RF 中运行脚本。

然后，使用频率计（在 PA8 上测量 HSE）评价固定的 HSETUNE 值的质量：HSE 频率越接近 32 MHz，HSETUNE 质量越好。

注意： *BLE_TransparentMode，系统时钟为 HSE。在该脚本中，HSE 处于关闭状态，因此在将 HSE 设置为关闭之前，将系统时钟切换到另一个时钟。*

erase_flash_page.txt

该脚本可用于部分清理 Flash 存储器，以将新的调谐参数保存在已经满的页面中。

注意： *对于该流程，脚本的运行不是强制性的。*

要擦除的页面编号是下面脚本行（第 20 行）中的最后一个参数，对于该样例，其被设置为 0xA0（因为交付的 trim_param_flash.txt 脚本将参数保存在该页面上）。

Send(VS_HCI_C1_WRITE_REGISTER;0x04;0x000007F8;0x58004014;0x00000500)

该行更新了 FLASH_CR 寄存器的 PNB 位。

注意： *若要计算页码（0x0500 → 0xA0），应考虑掩码 0x7F8。*

trim_param_flash.txt

该脚本将 HSETUNE 值保存在给定地址的 Flash 存储器中。

必须修改使用 HSE_tuning.txt 找到的 HSETUNE 值以获得最佳 HSE 频率，以及用于完成 64 位数据的其他信息和必须保存这些数据的地址。

下面的脚本行（第 19 行到第 25 行）包括要修改的 HSETUNE 值以及可以修改的其他信息。

例如，保存的 HSETUNE 值为 0x1B。

#写入低位 BD 地址：公司 Id = 0xE105 + 板 Id = 0x7777（例如）

Send(VS_HCI_C1_WRITE_REGISTER;0x04;0xFFFFFFFF;0x080A0000;0xE1057777)

Wait (100)

#写入高位 BD 地址：Id = 0x0 + HSE_Tun = 0x1B（例如） + 高位 BD = 0x0080

Send(VS_HCI_C1_WRITE_REGISTER;0x04;0xFFFFFFFF;0x080A0004;0x001B0080)

Wait (100)

这些行在存储器中保存一个 64 位数据，等于 0x 001B 0080 E105 7777（64 位是 Flash 存储器字行的大小）。

除 HSETUNE 值外，其他 58 位可以根据需要进行选择。例如，它们可以是蓝牙器件地址。

此外，0x080A0000 对应于保存 64 位数据的地址（0x080A0000 处的 32 个低位和 0x080A0004 处的 32 个高位）。

trim_param_otp.txt

trim_param_otp.txt 等效于 trim_param_flash.txt：用户必须修改脚本的相同元素才能获得所请求的行为。

然而，保存地址并不相同，因为 OPT 和 Flash 存储器在 STM32WB 微控制器中在不同地址映射。在交付的脚本中，地址接近 OTP 存储器的开头：0x1FFF7008 和 0x1FFF700C。

7.3.4 C 代码

retrieve_trimming_values.c

当调谐参数保存在 OTP 存储器中时，在 RF 应用中给出该 C 代码作为实现 HSE 时钟初始化流程的样例。

它收集：

- 两个常数，CFG_OTP_BASE_ADDRESS 和 CFG_OTP_END_ADDRESS，定位 OTP 存储器的开头和结尾。
- OTP_ID0_t 结构表示按 trim_param_otp.txt 保存在存储器中的 64 位数据。
- OTP_Read 函数用于查找保存在 OTP 存储器中的最新 64 位数据，ID 为 0（本应用笔记选择 0 作为 OTP 存储器中 HSETUNE 数据的索引）。
- Config_HSE 函数利用 OTP_Read 在 OTP 存储器中查找最新 HSETUNE 值，从而在 RCC_HSECR 寄存器中设置 HSETUNE 位并获得符合要求的 HSE 频率。

注意：　通过更改 Flash 存储器地址的两个常数值，可以将相同的代码用于 Flash 存储器。

8 针对 STM32WB 系列的 HSE 启动优化

HSE 启动时间会影响应用的时序和功耗。

利用 STM32WB 系列产品 HSECR 寄存器中的 HSEGMC 和 HSES 位，可以在启动可靠性和功耗之间找到最佳平衡：

- HSEGMC 值越高，启动时间越短，降低 HSEGMC 值可减少电流功耗
- HSES 不会修改功耗（峰值电流可以忽略不计，因为非常短），但在设置后会将启动时间缩短十几微秒。

要进行该调整，应执行以下流程。

第一个条件是晶体 ESR（等效串联电阻）必须低于产品数据手册中定义的最大值。然后，使用无源组件的参数，计算所实施的振荡器的最小跨导 gmcrit。

假设 CL1=CL2，晶体在其焊盘上看到晶体制造商给出的 CL 值（参见*图 1*），g_{mcrit}=4*ESR*(2*π*F)^2*(C0+CL）^{2}，其中：

- C0 为晶体分流电容
- CL 为晶体标称负载电容
- F 为晶体标称振荡频率

在整个流程中，HSEGMC 值必须保持在 gmcrit 以上（更多详细信息请参阅 AN2867）。

1. 将 HSES 置 1，获得更高比较阈值（3/4）。该更高的阈值缩短了 HSEON=1 和 HSERDY=1（启动时间）之间的时间。
2. 将 HSEGMC 设置为其最大值（111），并通过设置 HSEON=1 来检查振荡器是否正确启动，并观察 HSERDY 在启动时间后是否也达到 1。如果成功，则该配置对应于应用可以实现的最短启动时间。
3. 如果在设置 HSES 并且 HSEGMC 达到其最大值时未触发 HSERDY，则降低 HSEGMC，直到被触发为止。请注意，HSEGMC 连续 Gm 值与其二进制值不成线性，例如：b011 为 1.13 mA/V，b100 为 0.61 mA/V（详细信息请参阅参考手册）。事实上，HSEGMC 最高有效位区分了两个取值范围。此外，在降低 HSEGMC 后，其值不得设置为低于第一步中计算的 gmcrit 值。
4. 如果在没有成功启动的情况下测试了所有 HSEGMC 值，则 HSES 必须复位，并再次测试每个 HSEGMC 值，从其最高值开始，并且不低于 gmcrit。
5. 当配置触发 HSERDY 时，流程结束：启动阶段成功。

如果所有可能的配置都失败，则选择另一个晶体。

完成该流程后，找到（HSEGMC，HSES）组合，其对应于振荡器可以实现的最短启动时间。如果启动时间对应用来说不是问题，则进一步减少 HSEGMC，以降低功耗。请记住保持在 gmcrit 之上，并验证 HSERDY 是否在设置 HSEON 后被升高，这表明启动成功。

提出以下软件方法用于测量获得的启动持续时间：

1. 设置（HSEGMC，HSES）组合进行测试。
2. 配置 DWT 外设的定时器或时钟周期计数器功能，以便在 µs 范围内进行测量。
3. 在设置 HSEON 之前启动该测量。
4. 设置 HSEON 并等待 HSERDY 升高，然后停止测量。
5. 计数器值给出了启动阶段持续时间的近似值。

注意： *配置 HSECR 的正确顺序是首先调整 HSEGMC 和 HSES，检查是否成功启动。第二步是使用 HSETUNE 调谐 HSE 频率。*

9 结论

RF 应用需要非常准确的时钟，以确保最佳性能。RF 时钟来自外部晶体，通过在晶体引脚上设置正确的负载电容，对频率进行精细调谐。

STM32 无线 MCU 引入了非常高效的架构，其采用内部电容设置，无需在 PCB 上设置额外组件，减轻了对晶体性能的限制。

对于 STM32WB 系列产品，另一项设置使用户能够根据应用，定义振荡器启动持续时间和功耗之间的最佳平衡。

10 版本历史

表 6. 文档版本历史

日期	版本	变更
2017 年 9 月 27 日	1	初始版本。
2017 年 11 月 14 日	2	更新了： – *第 3.2.2 节：软件实现* – *第 3.2.3 节：脚本*
2019 年 2 月 21 日	3	文档分类由限于 ST 公司变为公开发布。 更新了*第 1 节：HSE 振荡器*。 对整个文档进行少量文字修订。
2020 年 1 月 30 日	4	介绍了 STM32WL 系列，因此更新了文档标题、*“前言”*和*第 9 节：结论*。 更新了*第 1 节：HSE 振荡器*，*第 1.2 节：STM32 无线 MCU 架构*，*第 1.3 节：HSE 配置参数 - STM32WB 系列*，*第 1.6 节：晶体参考文献*，*第 3 节：STM32WB 系列手动频率微调流程样例*，负载电容设置和项目配置。 增加了*第 1.4 节：HSE 配置参数-STM32WL 系列*，*第 2 节：微调方法比较*，*第 5 节：STM32WB 系列自动频率微调流程样例*和*第 7 节：STM32WB 系列 STM32CubeMonitor-RF 频率微调流程样例*。 更新了*表 3：晶体规格*。 对整个文档进行少量文字修订。
2020 年 2 月 4 日	5	更新了*“前言”*。
2020 年 4 月 27 日	6	更新了*第 1 节：HSE 振荡器*，*第 1.5 节：板实现*，*第 1.6 节：晶体参考文献*和*第 3 节：STM32WB 系列的手动频率微调流程样例*。 增加了*第 1.7 节：生产中的调谐*。 更新了*表 1：射频协议的载波精度要求*。 对整个文档进行少量文字修订。
2020 年 5 月 26 日	7	增加了注释：*第 2 节：微调方法比较*。

表 6. 文档修订历史（接上页）

日期	版本	变更
2020 年 6 月 29 日	8	更新了*“前言”*，*第 1.3 节：HSE 配置参数 - STM32WB 系列*，*第 1.7 节：生产中的调谐*，*第 3 节：STM32WB 系列的手动频率微调流程样例*，*第 3.2.1 节：硬件设置*，*第 3.2.3 节：脚本*，*第 5 节：STM32WB 系列自动频率微调流程样例*，*第 5.2.1 节：硬件设置、固件配置*，*第 5.2.3 节：脚本*，*第 7 节：STM32WB 系列的 STM32CubeMonitor-RF 频率微调流程样例*，*第 7.3.2 节：软件和脚本设置*以及*第 7.3.3 节：脚本*。 在*表 4：微调方法*和*表 5：微调方法比较*中添加了脚注。 更新了*图 5：以 PH3 引脚驱动的 BOOT0 值从 SRAM 启动的 OB 配置*，*图 6：以选项位 nBOOT0 驱动的 BOOT0 值从 SRAM 启动的 OB 配置*，*图 11：自动校准概述 -STM32WB 系列*和*图 14：STM32CubeMonitor-RF 校准概述-STM32WB 系列*。
2020 年 11 月 5 日	9	介绍了 STM32WL 系列，因此更新了*“前言”*并增加了*第 4 节：STM32WL 系列的手动频率微调流程样例*，*第 6 节：STM32WL 系列自动频率微调流程样例及其小节*。
2021 年 1 月 26 日	10	更新了文档标题、*“前言”*，*第 1 节：HSE 振荡器*，和*第 9 节：结论*。 增加了*第 8 节：STM32WB 系列 HSE 启动优化*。对整个文档进行少量文字修订。
2022 年 11 月 3 日	11	更新了*第 1.1 节：晶体振荡器*，*电流控制：HSEGMC[2:0]*和*第 8 节：STM32WB 系列 HSE 启动优化*。更新了*表 2：STM32WB 系列振荡器引脚数*。对整个文档进行少量文字修订。
2023 年 1 月 17 日	12	增加了*第 7.3.3 节：脚本*中的注释。 对整个文档进行少量文字修订。

表 7. 中文文档版本历史

日期	版本	变更
2023 年 10 月 13 日	1	中文初始版本。

重要通知 - 仔细阅读

意法半导体公司及其子公司（“意法半导体”）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意法半导体产品的最新信息。ST 产品的销售依照订单确认时的相关 ST 销售条款。

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

意法半导体不对任何知识产权进行任何明示或默示的授权或许可。

转售的意法半导体产品如有不同于此处提供的信息的规定，将导致意法半导体针对该产品授予的任何保证失效。

ST 及 ST 标识是意法半导体公司的商标。若需意法半导体商标的更多信息，请参考 www.st.com/trademarks。其他所有产品或服务名称是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。

© 2023 STMicroelectronics - 保留所有权利