如何用 bat 脚本代替按键调试 STM32WL HSE XO 的负载电容

关键字：HSE 校准， HSE 负载电容

1. bat 脚本方法简介

AN5042 中 Table 4 和 Table 5 已介绍了 STM32WL HSE XO 内部负载电容的调整方法的对比。

Table 4. Trimming methods

Method	Description
Manual	A precision frequency meter is used to measure the HSE frequency output on one of the STM32 pins. Then, the user tunes the HSE frequency with the buttons of a Nucleo board. A button is dedicated to the saving of the tuning parameters in the STM32 non-volatile memory.
Automatic	One STM32 timer is clocked with a precision external clock source provided by the user via one of the STM32 pins. This reference clock allows the user to measure the internal STM32 system HSE frequency. Then, the STM32 can compare the frequency measured with the one expected, to test and determine the best tuning parameters. Finally, the STM32 saves these parameters in its non-volatile memory.
STM32CubeMonitor-RF(1)	A precision frequency meter is used to measure the HSE frequency output on one of the STM32 pins. Then, the user tunes the HSE frequency with a script to run in STM32CubeMonitor-RF. The user has to change the tuning parameter values in the script to test them. When the correct values are found, another script saves them in the STM32 non-volatile memory.

1. Compatible with STM32WB Series only.

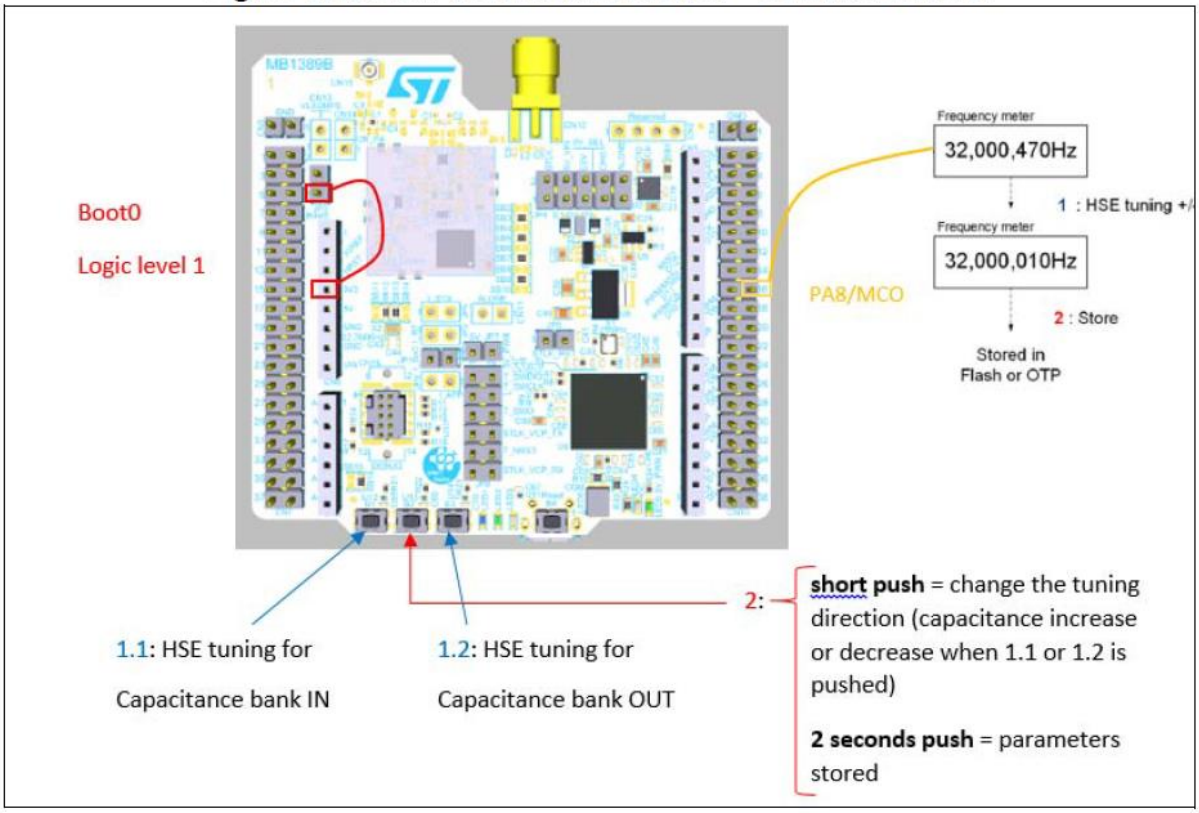
Table 5. Comparison of trimming methods

Method	Advantages	Disadvantages
Manual	Runs in SRAM (user program in Flash memory is not affected).	Needs the use of buttons and a frequency meter for each product.
Automatic	Runs in SRAM (user program in Flash memory is not affected). The user needs to set up a reference clock only once to trim as many devices as wanted.	Method based on a more complex principle.
STM32CubeMonitor-RF(1)	Convenient for users familiar with STM32CubeMonitor-RF, willing to achieve the maximum functionality.	BLE stack and transparent mode FW must be flashed in the device. Requires several actions from the user for each product (modifying and running the script, use of a frequency meter).

1. Compatible with STM32WB Series only.

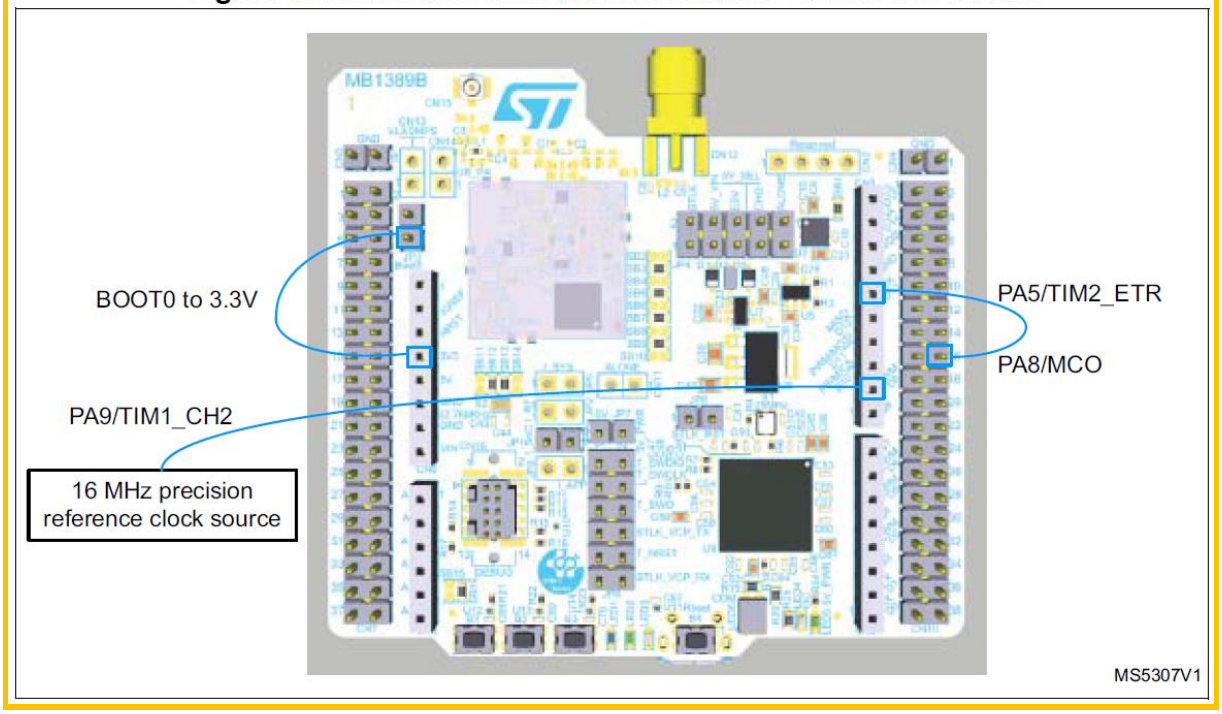
手动方法：需要一个精密频率计(能测量 32MHz)，STM32WL 1 个 GPIO PA8 (MCO 输出 32MHz 时钟)，STM32WL 3 个 GPIO 做按键。

Figure 8. Manual calibration overview - STM32WL Series



自动方法：需要一个精准的外部时钟源(精度要求小于 0.1 ppm)，STM32WL 1 个 GPIO PA9(外部时钟输入)， GPIO PA8 (MCO 输出 32MHz 时钟)， GPIO PA5 用作输入的 32MHz 时钟校准对比。

Figure 13. Automatic calibration overview - STM32WL Series

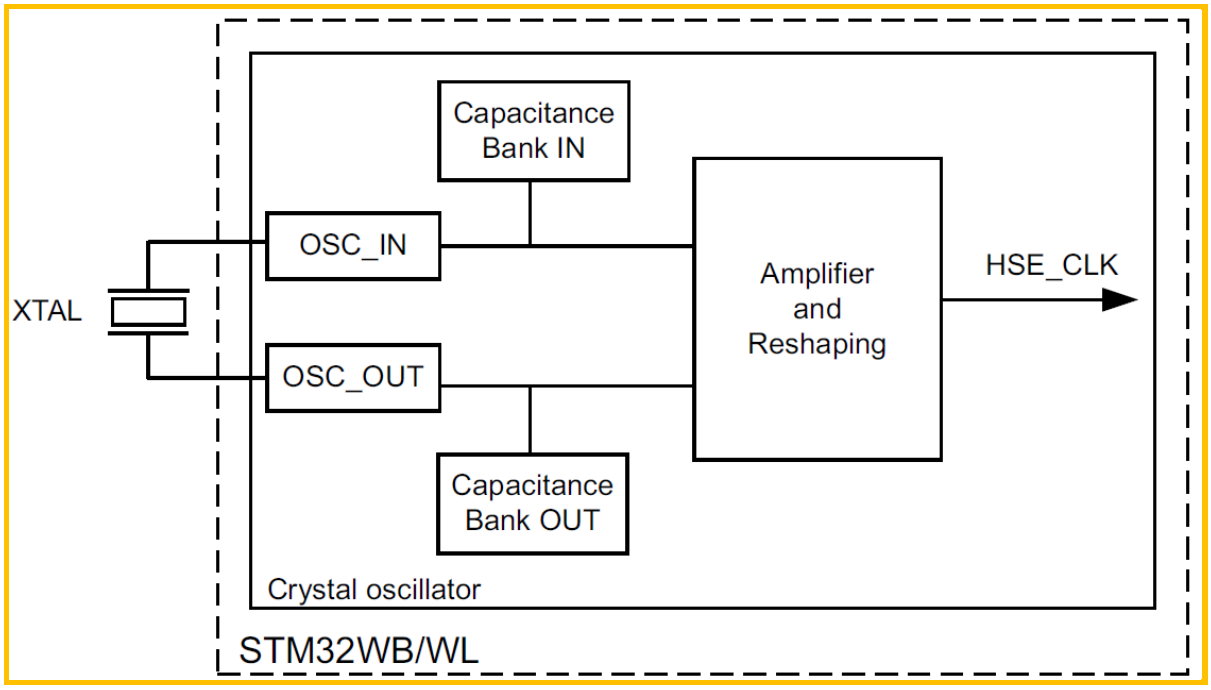


但是在实际客户产品中，可能没有预留足够多的按键，也没有预留 PA9 或 PA5, 或这些 GPIO 已用作其他功能，硬件不方便调整。还可能没法提供一个精度小于 0.1ppm 的外部时钟。

此时，可以使用 bat 的脚本来替代手动模式中的按键操作来实现 HSE XO 负载电容的调节。

原理：在 bat 脚本中通过 STM32CubeProgrammer 向 RAM 中写入操作命令(0x01)，STM32WL 轮训查询操作命令，并执行相应的操作命令来校准 HSE XO 的负载电容。

STM32WL HSE XO 内部架构如下图：



STM32WL Bank IN 和 OUT 的负载电容是可以独立调整，所以暂时定义操作命令如下表：

Offset (Bytes)	0	1	2	3	4	5	6	7
命令	IN +	IN -	OUT +	OUT -	save	check	RFU	RFU

2. bat 脚本手动方法实现

2.1. 代码修改

修改 X-CUBE-CLKTRIM_v3.0.0\Projects\NUCLEO-WL55JC\RCC_HSE_Calib_SingleCore\Src\main.c

添加 CMD_PTR 用于存放操作命令；声明 SaveHSETuneInRAM 可将校准值保存在 RAM 中；声明 SetHSETune 可设置校准值。

```
#define OTP_HSE_STR_IDX ((uint8_t)0x00) /*!< Index of the structure OTP_DATA_t  */
#define CMD_PTR ((uint32_t)0x20006FE0) /*!< SRAM Address to store cmd transmitted by STM32CubeProgrammer script */
#define ADDITIONAL_DATA_PTR ((uint32_t)0x20006FF0) /*!< SRAM Address to store address transmitted by STM32CubeProgrammer script */
```

```
static void SaveHSETune(uint8_t val_in, uint8_t val_out);
static void SaveHSETuneInRAM(uint8_t val_in, uint8_t val_out);
static void SetHSETune(uint8_t val_in, uint8_t val_out);
#else /* undef SET_CALIBRATION */
```

在 main 函数中，(可选)将设置默认的负载电容值并保存的 RAM 中，初始化操作命令，在 while 中命令轮训查询，并执行相应的命令，最后复位命令。

```
#ifndef SET_CALIBRATION
  /* In test mode, HSE calibration is retrieved from OTP area */

  if (FetchHSETune() == HAL_OK)
  {
    BSP_LED_On(LED2);
  }
  else
  {
    ErrorHandler();
  }
#else
  /* set default load capacitance value */
  SetHSETune(hsetune_in_val, hsetune_out_val);
  SaveHSETuneInRAM(hsetune_in_val, hsetune_out_val);
#endif /* SET_CALIBRATION */
```

```
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  uint8_t* cmd_ram = (uint8_t*)CMD_PTR;
  *(cmd_ram+0) = 0x00; /* in + */
  *(cmd_ram+1) = 0x00; /* in - */
  *(cmd_ram+2) = 0x00; /* out + */
  *(cmd_ram+3) = 0x00; /* out - */
  *(cmd_ram+4) = 0x00; /* save */
  *(cmd_ram+5) = 0x00; /* check */

  while (1)
```

```
while (1)
{
  /* USER CODE BEGIN 3 */
  if(*(cmd_ram+0) == 0x01) /* in + */
  {
    direction = 0;
    HAL_GPIO_EXTI_Callback(BUTTON_SW1_PIN);
    memset(cmd_ram, 0x00,8);
  }
  else if (*(cmd_ram+1) == 0x01) /* in - */
  {
    direction = 1;
    HAL_GPIO_EXTI_Callback(BUTTON_SW1_PIN);
    memset(cmd_ram, 0x00,8);
  }
  else if(*(cmd_ram+2) == 0x01) /* out + */
  {
    direction = 0;
    HAL_GPIO_EXTI_Callback(BUTTON_SW3_PIN);
    memset(cmd_ram, 0x00,8);
  }
  else if(*(cmd_ram+3) == 0x01) /* out - */
  {
    direction = 1;
    HAL_GPIO_EXTI_Callback(BUTTON_SW3_PIN);
    memset(cmd_ram, 0x00,8);
  }
  else if(*(cmd_ram+4) == 0x01) /* save */
  {
    SaveHSETune(hsetune_in_val, hsetune_out_val);
    memset(cmd_ram, 0x00,8);
  }
  else if(*(cmd_ram+5) == 0x01) /* check */
  {
    HAL_SUBGHZ_ReadRegisters( &hUserSubghz, 0x0911, &hsetune_in_val, 1 );
    HAL_SUBGHZ_ReadRegisters( &hUserSubghz, 0x0912, &hsetune_out_val, 1 );
    SaveHSETuneInRAM(hsetune_in_val, hsetune_out_val);
    memset(cmd_ram, 0x00,8);
  }
  /* USER CODE END 3 */
```

在 HAL_GPIO_EXTI_Callback 中，每次校准完后都将校准值保存到 RAM 中，(可选)删除 HAL_Delay

```
    default:
      BSP_LED_Off(LED1);
      BSP_LED_Off(LED2);
      BSP_LED_Off(LED3);
      break;
  } « end switch GPIO_Pin »

  SaveHSETuneInRAM(hsetune_in_val, hsetune_out_val);

  //HAL_Delay(1000);
  BSP_LED_Off(LED1);
```

在 SaveHSETune 后面，添加 SaveHSETuneInRAM 和 SetHSETune 函数

```
} « end SaveHSETune »

static void SaveHSETuneInRAM(uint8_t val_in, uint8_t val_out)
{
  OTP_DATA_t otp_data;

  /* Fill OTP_BT_t structure */
  memcpy(otp_data.additional_data, (void*)ADDITIONAL_DATA_PTR, 5);
  otp_data.hse_tuning_in = val_in & 0x3F;
  otp_data.hse_tuning_out = val_out & 0x3F;
  otp_data.index = OTP_HSE_STR_IDX;

  memcpy((uint8_t*)ADDITIONAL_DATA_PTR,(uint8_t*)&otp_data,sizeof(otp_data));
}

static void SetHSETune(uint8_t val_in, uint8_t val_out)
{
    /* set HSE tune load capacitance value */
    SetHSECalibrationIn(val_in);
    SetHSECalibrationOut(val_out);
}

#else /* undef SET_CALIBRATION */
```

IAR 编译工程生成 RCC_HSE_Calib_SingleCore\EWARM\RCC_HSE_Calib_SingleCore\Exe\RCC_HSE_Calib_SingleCore.bin

2.2. bat 脚本实现

在 X-CUBE-CLKTRIM_v3.0.0\Projects\NUCLEO-WL55JC\RCC_HSE_Calib_SingleCore\EWARM\下创建如下脚本，并修改 RCC_HSE_Calib_SingleCore_OTP.bat

cmd_boot_check.bat 查询 STM32WL 的启动模式，是从 flash 启动还是 RAM 启动？
cmd_boot_from_sram.bat 修改 Option Bytes 配置 STM32WL 从 RAM 启动。
cmd_boot_from_flash.bat 修改 Option Bytes 配置 STM32WL 从 Flash 启动。

cmd_hsetune_check.bat 检查当前的校准值
cmd_hsetune_in_plus.bat IN 校准值加 1
cmd_hsetune_in_minus.bat IN 校准值减 1
cmd_hsetune_out_plus.bat OUT 校准值加 1
cmd_hsetune_out_minus.bat OUT 校准值减 1
cmd_hsetune_save.bat 保存校准到 RAM 中

2.2.1. cmd_boot_check.bat

cmd_boot_check.bat

```
@echo off
REM This script is used to launch HSE calibration application on STM32WL nucleo board

REM If required, modify the next line to the current STM32-PROGRAMMER-CLI executable path
SET CLI="C:\Program Files\STMicroelectronics\STM32Cube\STM32CubeProgrammer\bin\STM32_Programmer_CLI.exe"

ECHO "STM32WL -- Check Option Bytes to check boot mode "

ECHO "SYSCFG memory remap register (SYSCFG_MEMRMP)"
ECHO "Bits 2:0 MEM_MODE[2:0]: memory mapping selection"
ECHO "000: Main Flash memory mapped at CPU1 0x00000000"
ECHO "001: System Flash memory mapped at CPU1 0x00000000"
ECHO "011: SRAM1 mapped at CPU1 0x00000000"
ECHO "110: QUADSPI memory mapped at CPU1 0x00000000"

REM get Option Bytes and memory remap
%CLI% -c port=swd mode=UR -ob displ -r32 0x40010000 4
timeout 1
pause
```

2.2.2. cmd_boot_from_sram.bat

cmd_boot_from_sram.bat

```
@echo off
REM This script is used to launch HSE calibration application on STM32WL nucleo board

REM If required, modify the next line to the current STM32-PROGRAMMER-CLI executable path
SET CLI="C:\Program Files\STMicroelectronics\STM32Cube\STM32CubeProgrammer\bin\STM32_Programmer_CLI.exe"

ECHO "STM32WL -- Modify Option Bytes to set boot from SRAM "

ECHO "SYSCFG memory remap register (SYSCFG_MEMRMP)"
ECHO "Bits 2:0 MEM_MODE[2:0]: memory mapping selection"
ECHO "000: Main Flash memory mapped at CPU 0x00000000"
ECHO "001: System Flash memory mapped at CPU 0x00000000"
ECHO "011: SRAM1 mapped at CPU 0x00000000"

REM get Option Bytes and memory remap
%CLI% -c port=swd mode=UR -ob displ -r32 0x40010000 4
timeout 1

REM Modify Option Bytes to set boot from SRAM
%CLI% -c port=swd mode=UR -ob nSWBOOT0=0x00 nBOOT0=0x00 nBOOT1=0x00
timeout 1

REM Check Option Bytes and memory remap
%CLI% -c port=swd mode=UR -ob displ -r32 0x40010000 4
pause
```

2.2.3. cmd_boot_from_flash.bat

cmd_boot_from_flash.bat

```
@echo off
REM This script is used to launch HSE calibration application on STM32WL nucleo board

REM If required, modify the next line to the current STM32-PROGRAMMER-CLI executable path
SET CLI="C:\Program Files\STMicroelectronics\STM32Cube\STM32CubeProgrammer\bin\STM32_Programmer_CLI.exe"

ECHO "STM32WL -- Modify Option Bytes to set boot from flash "

ECHO "SYSCFG memory remap register (SYSCFG_MEMRMP)"
ECHO "Bits 2:0 MEM_MODE[2:0]: memory mapping selection"
ECHO "000: Main Flash memory mapped at CPU 0x00000000"
ECHO "001: System Flash memory mapped at CPU 0x00000000"
ECHO "011: SRAM1 mapped at CPU 0x00000000"

REM get Option Bytes and memory remap
%CLI% -c port=swd mode=UR -ob displ -r32 0x40010000 4
timeout 1

REM Modify Option Bytes to set boot from flash
%CLI% -c port=swd mode=UR -ob nSWBOOT0=0x01 nBOOT0=0x01 nBOOT1=0x01
timeout 1

REM Check Option Bytes and memory remap
%CLI% -c port=swd mode=UR -ob displ -r32 0x40010000 4
pause
```

2.2.4. cmd_hsetune_check.bat

cmd_hsetune_check.bat

```
@echo off
REM This script is used to launch HSE calibration application on STM32WL nucleo board

REM If required, modify the next line to the current STM32-PROGRAMMER-CLI executable path
SET CLI="C:\Program Files\STMicroelectronics\STM32Cube\STM32CubeProgrammer\bin\STM32_Programmer_CLI.exe"

SET CMD_ADDR=0x20006FE0

ECHO "STM32WL -- HSE Triming -- cmd check"

REM Check cmd and hsetune
%CLI% -c port=swd mode=HOTPLUG shared -r32 %CMD_ADDR% 32
timeout 1

REM Set cmd save
%CLI%  -c port=swd mode=HOTPLUG shared -halt -w64 %CMD_ADDR% 0x3322110077660144 -run
timeout 2

REM Check cmd and hsetune
%CLI% -c port=swd mode=HOTPLUG shared -r32 %CMD_ADDR% 32
pause
```

2.2.5. cmd_hsetune_in_plus.bat

cmd_hsetune_in_plus.bat

```
@echo off
REM This script is used to launch HSE calibration application on STM32WL nucleo board

REM If required, modify the next line to the current STM32-PROGRAMMER-CLI executable path
SET CLI="C:\Program Files\STMicroelectronics\STM32Cube\STM32CubeProgrammer\bin\STM32_Programmer_CLI.exe"

SET CMD_ADDR=0x20006FE0

ECHO "STM32WL -- HSE Triming -- cmd in + "

REM Check cmd and hsetune
%CLI% -c port=swd mode=HOTPLUG shared -r32 %CMD_ADDR% 32
timeout 1

REM Set cmd in +
%CLI%  -c port=swd mode=HOTPLUG shared -halt -w64 %CMD_ADDR% 0x3322110177665544 -run
timeout 2

REM Check cmd and hsetune
%CLI% -c port=swd mode=HOTPLUG shared -r32 %CMD_ADDR% 32
pause
```

2.2.6. cmd_hsetune_in_minus.bat

cmd_hsetune_in_minus.bat

```
@echo off
REM This script is used to launch HSE calibration application on STM32WL nucleo board

REM If required, modify the next line to the current STM32-PROGRAMMER-CLI executable path
SET CLI="C:\Program Files\STMicroelectronics\STM32Cube\STM32CubeProgrammer\bin\STM32_Programmer_CLI.exe"

SET CMD_ADDR=0x20006FE0

ECHO "STM32WL -- HSE Triming -- cmd in - "

REM Check cmd and hsetune
%CLI% -c port=swd mode=HOTPLUG shared -r32 %CMD_ADDR% 32
timeout 1

REM Set cmd in -
%CLI% -c port=swd mode=HOTPLUG shared -halt -w64 %CMD_ADDR% 0x3322010077665544 -run
timeout 2

REM Check cmd and hsetune
%CLI% -c port=swd mode=HOTPLUG shared -r32 %CMD_ADDR% 32
pause
```

2.2.7. cmd_hsetune_out_plus.bat

cmd_hsetune_out_plus.bat

```
@echo off
REM This script is used to launch HSE calibration application on STM32WL nucleo board

REM If required, modify the next line to the current STM32-PROGRAMMER-CLI executable path
SET CLI="C:\Program Files\STMicroelectronics\STM32Cube\STM32CubeProgrammer\bin\STM32_Programmer_CLI.exe"

SET CMD_ADDR=0x20006FE0

ECHO "STM32WL -- HSE Triming -- cmd out + "

REM Check cmd and hsetune
%CLI% -c port=swd mode=HOTPLUG shared -r32 %CMD_ADDR% 32
timeout 1

REM Set cmd out +
%CLI% -c port=swd mode=HOTPLUG shared -halt -w64 %CMD_ADDR% 0x3301110077665544 -run
timeout 2

REM Check cmd and hsetune
%CLI% -c port=swd mode=HOTPLUG shared -r32 %CMD_ADDR% 32
pause
```

2.2.8. cmd_hsetune_out_minus.bat

cmd_hsetune_out_minus.bat

```
@echo off
REM This script is used to launch HSE calibration application on STM32WL nucleo board

REM If required, modify the next line to the current STM32-PROGRAMMER-CLI executable path
SET CLI="C:\Program Files\STMicroelectronics\STM32Cube\STM32CubeProgrammer\bin\STM32_Programmer_CLI.exe"

SET CMD_ADDR=0x20006FE0

ECHO "STM32WL -- HSE Triming -- cmd out - "

REM Check cmd and hsetune
%CLI% -c port=swd mode=HOTPLUG shared -r32 %CMD_ADDR% 32
timeout 1

REM Set cmd out -
%CLI% -c port=swd mode=HOTPLUG shared -halt -w64 %CMD_ADDR% 0x0122110077665544 -run
timeout 2

REM Check cmd and hsetune
%CLI% -c port=swd mode=HOTPLUG shared -r32 %CMD_ADDR% 32
pause
```

2.2.9. cmd_hsetune_save.bat

cmd_hsetune_save.bat

```
@echo off
REM This script is used to launch HSE calibration application on STM32WL nucleo board

REM If required, modify the next line to the current STM32-PROGRAMMER-CLI executable path
SET CLI="C:\Program Files\STMicroelectronics\STM32Cube\STM32CubeProgrammer\bin\STM32_Programmer_CLI.exe"

SET CMD_ADDR=0x20006FE0

ECHO "STM32WL -- HSE Triming -- cmd save "

REM Check cmd and hsetune
%CLI% -c port=swd mode=HOTPLUG shared -r32 %CMD_ADDR% 32
timeout 1

REM Set cmd save
%CLI% -c port=swd mode=HOTPLUG shared -halt -w64 %CMD_ADDR%  0x3322110077665501 -run
timeout 2

REM Check cmd and hsetune
%CLI% -c port=swd mode=HOTPLUG shared -r32 %CMD_ADDR% 32
pause
```

2.2.10. RCC_HSE_Calib_SingleCore_OTP.bat

RCC_HSE_Calib_SingleCore_OTP.bat

```
@echo off
REM This script is used to launch HSE calibration application on STM32WL nucleo board

REM If required, modify the next line to the current STM32-PROGRAMMER-CLI executable path
SET CLI="C:\Program Files\STMicroelectronics\STM32Cube\STM32CubeProgrammer\bin\STM32_Programmer_CLI.exe"
REM Application binary path (here for EWARM)
SET BINFILE="RCC_HSE_Calib_SingleCore\Exe\RCC_HSE_Calib_SingleCore.bin"
SET CMD_ADDR=0x20006FE0
ECHO "STM32WL -- Programing STM32WL Nucleo board for extra data and HSE Triming"

REM Next command stores the additional data at in SRAM @=0x20006FF0
REM additional data is 5 bytes. Most significant byte will be replaced by load capacitance value
%CLI% -c port=swd -w32 0x20006FF0  0x00000000
%CLI% -c port=swd -w32 0x20006FF4  0x00000000
%CLI% -c port=swd mode=HOTPLUG -r32 %CMD_ADDR%  32
REM Next command sends the binary in SRAM (@0x20000000) and launch the application
%CLI% -c port=swd -w %BINFILE% 0x20000000 -g 0x20000000
%CLI% -c port=swd mode=HOTPLUG -r32 %CMD_ADDR%  32
pause
```

3. bat 脚本使用

双击鼠标左键可运行 bat 脚本。

运行 cmd_boot_from_sram.bat 修改 Option Bytes 配置 STM32WL 从 RAM 启动。

运行 RCC_HSE_Calib_SingleCore_OTP.bat 下载 RCC_HSE_Calib_SingleCore.bin 到 RAM 中并运行测试程序。

运行 cmd_hsetune_check.bat 检查当前的校准值。

运行以下脚本调整 HSE XO IN 或 OUT 的负载电容值，并根据频率计查看 32MHz 时钟是否准确

cmd_hsetune_in_plus.bat IN 校准值加 1
cmd_hsetune_in_minus.bat IN 校准值减 1
cmd_hsetune_out_plus.bat OUT 校准值加 1
cmd_hsetune_out_minus.bat OUT 校准值减 1

调整完成后，运行 cmd_hsetune_save.bat 保存校准对到 RAM 中，并记录下来。

运行 cmd_boot_from_flash.bat 修改 Option Bytes 配置 STM32WL 从 Flash 启动。再下载应用以上校准值对的应用程序到 flash。

4. 校准值应用

4.1. HSE XO 支持配置

首先，确实是否使用 USE_BSP_DRIVER，

```
int32_t RBI_IsTCXO(void)
{
  /* USER CODE BEGIN RBI_IsTCXO_1 */

  /* USER CODE END RBI_IsTCXO_1 */
#if defined(USE_BSP_DRIVER)
  /* Important note: BSP code is board dependent
   * STM32WL_Nucleo code can be found
   *       either in STM32CubeWL package under Drivers/BSP/STM32WLxx_Nucleo/
   *       or at https://github.com/STMicroelectronics/STM32CubeWL/tree/main/Drivers/BSP/STM32WLxx_Nucleo/
   * 1/ For User boards, the BSP/STM32WLxx_Nucleo/ directory can be copied and replaced in the project. The copy must then be updated depending:
   *       on board RF switch configuration (pin control, number of port etc)
   *       on TCXO configuration
   *       on DC/DC configuration */
  return BSP_RADIO_IsTCXO();
#else
  /* 2/ Or implement RBI_IsTCXO here */
  int32_t retcode = IS_TCXO_SUPPORTED;
  /* USER CODE BEGIN RBI_IsTCXO_2 */
#warning user to provide its board code or to call his board driver functions
  /* USER CODE END RBI_IsTCXO_2 */
  return retcode;
#endif  /* USE_BSP_DRIVER  */
} « end RBI_IsTCXO »
```

LoRaWAN_AT_Slave\Core\Inc\ platform.h 中可确认。

```
#define USE_BSP_DRIVER
/* USER CODE BEGIN EC */
```

如果定义了 USE_BSP_DRIVER，则是使用 ST 的射频驱动。

可在 Drivers\BSP\STM32WLxx_Nucleo\ stm32wlxx_nucleo_radio.c 中的 BSP_RADIO_IsTCXO 中确认代码设置是不支持 TCXO 的，返回 RADIO_CONF_TCXO_NOT_SUPPORTED

```
/**
  * @brief  Get If TCXO is to be present on board
  * @note   never remove called by MW,
  * @retval
  *  RADIO_CONF_TCXO_NOT_SUPPORTED
  *  RADIO_CONF_TCXO_SUPPORTED
  */
int32_t BSP_RADIO_IsTCXO(void)
{
  return RADIO_CONF_TCXO_NOT_SUPPORTED;
}
```

如果没有定义 USE_BSP_DRIVER，则在 RBI_IsTCXO 中返回 0.

或 LoRaWAN_AT_Slave\LoRaWAN\Target\radio_board_if.h 定义 IS_TCXO_SUPPORTED 为 0

```
/* Indicates whether or not TCXO is supported by the board
 * 0: TCXO not supported
 * 1: TCXO supported
 */
#define IS_TCXO_SUPPORTED                   0U
```

4.2. HSE Tune 值应用

在 Middlewares\Third_Party\SubGHz_Phy\stm32_radio_driver\radio_driver.c 中 SUBGRF_Init 函数中通过 SUBGRF_WriteRegister 对 HSE XO IN 寄存器 REG_XTA_TRIM(0x0911)和 HSE XO OUT 寄存器 REG_XTB_TRIM(0x0912)分别写入以上步骤调好的值。

```
void SUBGRF_Init( DioIrqHandler dioIrq )
{
    if ( dioIrq != NULL)
    {
        RadioOnDioIrqCb = dioIrq;
    }

    RADIO_INIT();

    /* set default SMPS current drive to default*/
    Radio_SMPS_Set(SMPS_DRIVE_SETTING_DEFAULT);

    ImageCalibrated = false;

    SUBGRF_SetStandby( STDBY_RC );

    // Initialize TCXO control
    if (1U == RBI_IsTCXO() )
    {
        SUBGRF_SetTcxoMode( TCXO_CTRL_VOLTAGE, RF_WAKEUP_TIME << 6 );// 100 ms
        SUBGRF_WriteRegister( REG_XTA_TRIM, 0x00 );

        /*enable calibration for cut1.1 and later*/
        CalibrationParams_t calibParam;
        calibParam.Value = 0x7F;
        SUBGRF_Calibrate( calibParam );
    }
    else
    {
        SUBGRF_WriteRegister( REG_XTA_TRIM, XTAL_DEFAULT_CAP_VALUE );
        SUBGRF_WriteRegister( REG_XTB_TRIM, XTAL_DEFAULT_CAP_VALUE );
    }
    /* Init RF Switch */
    RBI_Init();

    OperatingMode = MODE_STDBY_RC;
} « end SUBGRF_Init »
```

如果两个数值相同，可在 LoRaWAN_AT_Slave\LoRaWAN\Target\radio_conf.h 中仅仅修改 XTAL_DEFAULT_CAP_VALUE。

注意：量产时，强烈建议从 OTP 或 Flash 中读取 HES XO 负载电容的调整值，具体方法参考 AN5042。

参考文献

【如有， 请注明；否则，请注明：无】

文件编号	文件标题	版本号	发布日期
AN5042	Precise HSE frequency and startup time tuning for STM32 wireless MCUs	10	January 2021

文档中所用到的工具及版本

IAR v8.50.9

STM32CubeProgrammer v2.9.0

X-CUBE-CLKTRIM v3.0

LAT 中的附件

RCC_HSE_Calib_SingleCore-bat-v0.3.zip

RCC_HSE_Calib_SingleCore-bat-v0.3.zip 是 X-CUBE-CLKTRIM_v3.0.0\Projects\NUCLEO-WL55JC\RCC_HSE_Calib_SingleCore 整个测试工程

版本历史

日期	版本	变更
2022 年 05 月 23 日	1.0	首版发布

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。若需 ST 商标的更多信息，请参考 *www.st.com/trademarks*。所有其他产品或服务名称均为其 各自所有者的财产。

本文档是 ST 中国本地团队的技术性文章，旨在交流与分享，并期望借此给予客户产品应用上足够的帮助或提醒。若文中内容存有局限或与 ST 官网资料不一致，请以实际应用验证结果和 ST 官网最新发布的内容为准。您拥有完全自主权是否采纳本文档（包括代码，电路图等）信息，我们也不承担因使用或采纳本文档内容而导致的任何风险。

本文档中的信息取代本文档所有早期版本中提供的信息。