

---

## N32G43x&N32L43x&N32L40x系列HSI频率调节应用笔记

---

### 简介

本文档介绍了 HSI 频率调节方法，便于用户在正常使用中根据自己实际需求调节 HSI 的频率。

本文档适用于国民技术的 N32G43x&N32L43x&N32L40x 系列产品。

# 目录

目录 .....	I
1. 概述 .....	1
2. 操作方法 .....	2
2.1 参数定义 .....	2
2.2 使用方法 .....	4
2.2.1 API 函数 .....	4
2.2.2 系统时钟设置 .....	5
2.3 应用示例 .....	7
3. 历史版本 .....	8
4. 声 明 .....	9

## 1. 概述

MCU 提供多种时钟供用户选择，其中包括内部 RC 震荡器 HSI（16MHz）。HSI RC 振荡器能够在不需要任何外部器件的条件下提供系统时钟。

如果用户的应用基于不同的电压或环境温度，这将会影响 RC 振荡器的精度。可以通过 TRIMR1 寄存器里的 afec\_hsi\_opt[24:21]位粗调 HSI 频率，步长约为 700kHz，afec\_hsi\_trim[20:16]细调 HSI 频率，步长约为 40kHz。

## 2. 操作方法

### 2.1 参数定义

预先定义以下几个参数：

```
/** hsi_opt_dec **/  
  
#define AFEC_HSI_OPT_NUM0 ((uint32_t)0x0<<21)  
#define AFEC_HSI_OPT_NUM1 ((uint32_t)0x1<<21)  
#define AFEC_HSI_OPT_NUM2 ((uint32_t)0x2<<21)  
#define AFEC_HSI_OPT_NUM3 ((uint32_t)0x3<<21)  
#define AFEC_HSI_OPT_NUM4 ((uint32_t)0x4<<21)  
#define AFEC_HSI_OPT_NUM5 ((uint32_t)0x5<<21)  
#define AFEC_HSI_OPT_NUM6 ((uint32_t)0x6<<21)  
#define AFEC_HSI_OPT_NUM7 ((uint32_t)0x7<<21)  
#define AFEC_HSI_OPT_NUM8 ((uint32_t)0x8<<21) //default  
#define AFEC_HSI_OPT_NUM9 ((uint32_t)0x9<<21)  
#define AFEC_HSI_OPT_NUM10 ((uint32_t)0xA<<21)  
#define AFEC_HSI_OPT_NUM11 ((uint32_t)0xB<<21)  
#define AFEC_HSI_OPT_NUM12 ((uint32_t)0xC<<21)  
#define AFEC_HSI_OPT_NUM13 ((uint32_t)0xD<<21)  
#define AFEC_HSI_OPT_NUM14 ((uint32_t)0xE<<21)  
#define AFEC_HSI_OPT_NUM15 ((uint32_t)0xF<<21)  
  
#define IS_AFEC_HSI_OPT(NUM) \\\n    (((NUM) == AFEC_HSI_OPT_NUM0) || ((NUM) == AFEC_HSI_OPT_NUM1) \\\n    || ((NUM) == AFEC_HSI_OPT_NUM2) || ((NUM) == AFEC_HSI_OPT_NUM3) \\\n    || ((NUM) == AFEC_HSI_OPT_NUM4) || ((NUM) == AFEC_HSI_OPT_NUM5) \\\n    || ((NUM) == AFEC_HSI_OPT_NUM6) || ((NUM) == AFEC_HSI_OPT_NUM7) \\\n    || ((NUM) == AFEC_HSI_OPT_NUM8) || ((NUM) == AFEC_HSI_OPT_NUM9) \\\n    || ((NUM) == AFEC_HSI_OPT_NUM10) || ((NUM) == AFEC_HSI_OPT_NUM11) \\\n    || ((NUM) == AFEC_HSI_OPT_NUM12) || ((NUM) == AFEC_HSI_OPT_NUM13) \\\n    || ((NUM) == AFEC_HSI_OPT_NUM14) || ((NUM) == AFEC_HSI_OPT_NUM15))
```

```
/** hsi_trim_dec **/  
  
#define AFEC_HSI_TRIM_NUM0 ((uint32_t)0x0<<16)  
#define AFEC_HSI_TRIM_NUM1 ((uint32_t)0x1<<16)  
#define AFEC_HSI_TRIM_NUM2 ((uint32_t)0x2<<16)  
#define AFEC_HSI_TRIM_NUM3 ((uint32_t)0x3<<16)  
#define AFEC_HSI_TRIM_NUM4 ((uint32_t)0x4<<16)  
#define AFEC_HSI_TRIM_NUM5 ((uint32_t)0x5<<16) //default  
#define AFEC_HSI_TRIM_NUM6 ((uint32_t)0x6<<16)  
#define AFEC_HSI_TRIM_NUM7 ((uint32_t)0x7<<16)  
#define AFEC_HSI_TRIM_NUM8 ((uint32_t)0x8<<16)  
#define AFEC_HSI_TRIM_NUM9 ((uint32_t)0x9<<16)  
#define AFEC_HSI_TRIM_NUM10 ((uint32_t)0xA<<16)  
#define AFEC_HSI_TRIM_NUM11 ((uint32_t)0xB<<16)  
#define AFEC_HSI_TRIM_NUM12 ((uint32_t)0xC<<16)  
#define AFEC_HSI_TRIM_NUM13 ((uint32_t)0xD<<16)  
#define AFEC_HSI_TRIM_NUM14 ((uint32_t)0xE<<16)  
#define AFEC_HSI_TRIM_NUM15 ((uint32_t)0xF<<16)  
#define AFEC_HSI_TRIM_NUM16 ((uint32_t)0x10<<16)  
#define AFEC_HSI_TRIM_NUM17 ((uint32_t)0x11<<16)  
#define AFEC_HSI_TRIM_NUM18 ((uint32_t)0x12<<16)  
#define AFEC_HSI_TRIM_NUM19 ((uint32_t)0x13<<16)  
#define AFEC_HSI_TRIM_NUM20 ((uint32_t)0x14<<16)  
#define AFEC_HSI_TRIM_NUM21 ((uint32_t)0x15<<16)  
#define AFEC_HSI_TRIM_NUM22 ((uint32_t)0x16<<16)  
#define AFEC_HSI_TRIM_NUM23 ((uint32_t)0x17<<16)  
#define AFEC_HSI_TRIM_NUM24 ((uint32_t)0x18<<16)  
#define AFEC_HSI_TRIM_NUM25 ((uint32_t)0x19<<16)  
#define AFEC_HSI_TRIM_NUM26 ((uint32_t)0x1A<<16)  
#define AFEC_HSI_TRIM_NUM27 ((uint32_t)0x1B<<16)  
#define AFEC_HSI_TRIM_NUM28 ((uint32_t)0x1C<<16)  
#define AFEC_HSI_TRIM_NUM29 ((uint32_t)0x1D<<16)
```

```
#define AFEC_HSI_TRIM_NUM30 ((uint32_t)0x1E<<16)
#define AFEC_HSI_TRIM_NUM31 ((uint32_t)0x1F<<16)

#define IS_AFEC_HSI_TRIM(NUM) \
    (((NUM) == AFEC_HSI_TRIM_NUM0) || ((NUM) == AFEC_HSI_TRIM_NUM1) \
    || ((NUM) == AFEC_HSI_TRIM_NUM2) || ((NUM) == AFEC_HSI_TRIM_NUM3) \
    || ((NUM) == AFEC_HSI_TRIM_NUM4) || ((NUM) == AFEC_HSI_TRIM_NUM5) \
    || ((NUM) == AFEC_HSI_TRIM_NUM6) || ((NUM) == AFEC_HSI_TRIM_NUM7) \
    || ((NUM) == AFEC_HSI_TRIM_NUM8) || ((NUM) == AFEC_HSI_TRIM_NUM9) \
    || ((NUM) == AFEC_HSI_TRIM_NUM10) || ((NUM) == AFEC_HSI_TRIM_NUM11) \
    || ((NUM) == AFEC_HSI_TRIM_NUM12) || ((NUM) == AFEC_HSI_TRIM_NUM13) \
    || ((NUM) == AFEC_HSI_TRIM_NUM14) || ((NUM) == AFEC_HSI_TRIM_NUM15) \
    || ((NUM) == AFEC_HSI_TRIM_NUM16) || ((NUM) == AFEC_HSI_TRIM_NUM17) \
    || ((NUM) == AFEC_HSI_TRIM_NUM18) || ((NUM) == AFEC_HSI_TRIM_NUM19) \
    || ((NUM) == AFEC_HSI_TRIM_NUM20) || ((NUM) == AFEC_HSI_TRIM_NUM21) \
    || ((NUM) == AFEC_HSI_TRIM_NUM22) || ((NUM) == AFEC_HSI_TRIM_NUM23) \
    || ((NUM) == AFEC_HSI_TRIM_NUM24) || ((NUM) == AFEC_HSI_TRIM_NUM25) \
    || ((NUM) == AFEC_HSI_TRIM_NUM26) || ((NUM) == AFEC_HSI_TRIM_NUM27) \
    || ((NUM) == AFEC_HSI_TRIM_NUM28) || ((NUM) == AFEC_HSI_TRIM_NUM29) \
    || ((NUM) == AFEC_HSI_TRIM_NUM30) || ((NUM) == AFEC_HSI_TRIM_NUM31))
```

## 2.2 使用方法

### 2.2.1 API 函数

调用下面的 API 函数，MCU 重新设置内部高速时钟校准值，用以校准内部 HSI RC 振荡器的频率。

```
void AFEC_ConfigHSITrim(uint32_t HSI_OPT, uint32_t HSI_TRIM)
{
    uint32_t tmpregister = 0;

    /* Check the parameters */
    assert_param(IS_AFEC_HSI_OPT(HSI_OPT));
    assert_param(IS_AFEC_HSI_TRIM(HSI_TRIM));

    tmpregister = AFEC->TRIMR1;
```

```
/* Clear OPT and TRIM[24:16] bits */
tmpregister &= ((uint32_t)0xFE00FFFF);

/* Set OPT[24:21] bits according to AFEC_HSI_OPT value */
tmpregister |= HSI_OPT;

/* Set OPT[20:16] bits according to AFEC_HSI_TRIM value */
tmpregister |= HSI_TRIM;

/* Store the new value */
AFEC->TRIMR1 = tmpregister;
}
```

## 2.2.2 系统时钟设置

参照下面的函数，将系统时钟设置为 16MHz，采用 HSI 作为时钟源。

```
void SetSysClockToHSI(void)
{
    uint32_t msi_ready_flag = RESET;

    RCC_EnableHsi(ENABLE);

    /* Wait till HSI is ready */
    HSIStartUpStatus = RCC_WaitHsiStable();

    if (HSIStartUpStatus == SUCCESS)
    {
        /* Enable Prefetch Buffer */
        FLASH_PrefetchBufSet(FLASH_PrefetchBuf_EN);

        if(((__(IO uint8_t*)(UCID_BASE + 0x2))) == 0x01)
        || ((__(IO uint8_t*)(UCID_BASE + 0x2))) == 0x11)
        || ((__(IO uint8_t*)(UCID_BASE + 0x2))) == 0xFF)
        {
            /* Cheak if MSI is Ready */
            if(RESET == RCC_GetFlagStatus(RCC_CTRLSTS_FLAG_MSIRD))
            {
                /* Enable MSI and Config Clock */
                RCC_ConfigMsi(RCC_MSI_ENABLE, RCC_MSI_RANGE_4M);

                /* Waits for MSI start-up */
                while(SUCCESS != RCC_WaitMsiStable());

                msi_ready_flag = SET;
            }
        }
    }
}
```

```
/* Select MSI as system clock source */
RCC_ConfigSysclk(RCC_SYSCLK_SRC_MSI);

/* Disable PLL */
RCC_EnablePll(DISABLE);

RCC_ConfigPll(RCC_PLL_HSI_PRE_DIV2, RCC_PLL_MUL_2, RCC_PLLDIVCLK_DISABLE);

/* Enable PLL */
RCC_EnablePll(ENABLE);

/* Wait till PLL is ready */
while (RCC_GetFlagStatus(RCC_CTRL_FLAG_PLLRDF) == RESET);

/* Select PLL as system clock source */
RCC_ConfigSysclk(RCC_SYSCLK_SRC_PLLCLK);

/* Wait till PLL is used as system clock source */
while (RCC_GetSysclkSrc() != 0x0C);

if(msi_ready_flag == SET)
{
    /* MSI oscillator OFF */
    RCC_ConfigMsi(RCC_MSI_DISABLE, RCC_MSI_RANGE_4M);
}
else
{
    /* Select HSI as system clock source */
    RCC_ConfigSysclk(RCC_SYSCLK_SRC_HSI);

    /* Wait till HSI is used as system clock source */
    while (RCC_GetSysclkSrc() != 0x04)
    {
    }
}

/* Flash 0 wait state */
FLASH_SetLatency(FLASH_LATENCY_0);

/* HCLK = SYSCLK */
RCC_ConfigHclk(RCC_SYSCLK_DIV1);

/* PCLK2 = HCLK */
```

```
RCC_ConfigPclk2(RCC_HCLK_DIV1);

/* PCLK1 = HCLK */
RCC_ConfigPclk1(RCC_HCLK_DIV1);
}
else
{
    /* If HSI fails to start-up, the application will have wrong clock configuration.
       User can add here some code to deal with this error */

    /* Go to infinite loop */
    while (1)
    {
    }
}
}
```

## 2.3 应用示例

参照应用笔记例程 RCC\_HSIClockTrim，演示了如何调节 HSI 频率，可通过示波器查看波形的频率变化。

### 3. 历史版本

版本	日期	备注
V1.0	2021-06-10	创建文档

## 4. 声 明

国民技术股份有限公司（下称“国民技术”）对此文档拥有专属产权。依据中华人民共和国的法律、条约以及世界其他法域相适用的管辖，此文档及其中描述的国民技术产品（下称“产品”）为公司所有。

国民技术在此并未授予专利权、著作权、商标权或其他任何知识产权许可。所提到或引用的第三方名称或品牌（如有）仅用作区别之目的。

国民技术保留随时变更、订正、增强、修改和改良此文档的权利，恕不另行通知。请使用人在下单购买前联系国民技术获取此文档的最新版本。

国民技术竭力提供准确可信的资讯，但即便如此，并不推定国民技术对此文档准确性和可靠性承担责任。

使用此文档信息以及生成产品时，使用者应当进行合理的设计、编程并测试其功能性和安全性，国民技术不对任何因使用此文档或本产品而产生的任何直接、间接、意外、特殊、惩罚性或衍生性损害结果承担责任。

国民技术对于产品在系统或设备中的应用效果没有任何故意或保证，如有任何应用在其发生操作不当或故障情况下，有可能致使人员伤亡、人身伤害或严重财产损失，则此类应用被视为“不安全使用”。

不安全使用包括但不限于：外科手术设备、原子能控制仪器、飞机或宇宙飞船仪器、所有类型的安全装置以及其他旨在支持或维持生命的应用。

所有不安全使用的风险应由使用人承担，同时使用人应使国民技术免于因为这类不安全使用而导致被诉、支付费用、发生损害或承担责任时的赔偿。

对于此文档和产品的任何明示、默示之保证，包括但不限于适销性、特定用途适用性和不侵权的保证责任，国民技术可在法律允许范围内进行免责。

未经明确许可，任何人不得以任何理由对此文档的全部或部分进行使用、复制、修改、抄录和传播。