

Application Note

N32G43x&N32L43x&N32L40x Series HSI Frequency Adjustment Application Notes

Introduction

This document introduces the HSI frequency adjustment method, which is convenient for users to adjust the HSI frequency according to their actual needs in normal use.

This document is applicable to N32G43x&N32L43x&N32L40x series products of National Technology.

All Rights Reserved

Contents

Contents	I
1 Overview	1
2 Operation method	2
2.1 Parameter definition	2
2.2 Method of use	4
2.2.1 API functions	4
2.2.2 System clock setting	4
2.3 Application example	6
3 Version history	7
4 NOTICE	8

1 Overview

MCU provides a variety of clocks for users to choose, including the internal RC oscillator HSI (16MHz). The HSI RC oscillator can provide the system clock without any external devices.

If the user's application is based on different voltages or ambient temperatures, this will affect the accuracy of the RC oscillator. The HSI frequency can be roughly adjusted by the `afec_hsi_opt[24:21]` bit in the `TRIMR1` register with a step of about 700kHz, and the HSI frequency can be finely adjusted by `afec_hsi_trim[20:16]` with a step of about 40kHz.

2 Operation method

2.1 Parameter definition

The following parameters are predefined:

```

/** 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) \
    ((NUM) == AFEC_HSI_OPT_NUM0) || ((NUM) == AFEC_HSI_OPT_NUM1) \
    || ((NUM) == AFEC_HSI_OPT_NUM2) || ((NUM) == AFEC_HSI_OPT_NUM3) \
    || ((NUM) == AFEC_HSI_OPT_NUM4) || ((NUM) == AFEC_HSI_OPT_NUM5) \
    || ((NUM) == AFEC_HSI_OPT_NUM6) || ((NUM) == AFEC_HSI_OPT_NUM7) \
    || ((NUM) == AFEC_HSI_OPT_NUM8) || ((NUM) == AFEC_HSI_OPT_NUM9) \
    || ((NUM) == AFEC_HSI_OPT_NUM10) || ((NUM) == AFEC_HSI_OPT_NUM11) \
    || ((NUM) == AFEC_HSI_OPT_NUM12) || ((NUM) == AFEC_HSI_OPT_NUM13) \
    || ((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 Method of use

2.2.1 API functions

Call the following API function, and the MCU resets the internal high-speed clock calibration value to calibrate the frequency of the internal HSI RC oscillator.

```
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 System clock setting

Refer to the following function to set the system clock to 16MHz and use HSI as the clock source.

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

    RCC_EnableHsi(ENABLE);

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

    if (HSISStartUpStatus == 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 Application example

Refer to the application note example RCC_HSIClockTrim, which demonstrates how to adjust the HSI frequency, and the frequency change of the waveform can be viewed through an oscilloscope.

3 Version history

Version	Date	Note
V1.0	2021-06-10	Create the document

4 NOTICE

This document is the exclusive property of Nations Technologies Inc. (Hereinafter referred to as NATIONS). This document, and the product of NATIONS described herein (Hereinafter referred to as the Product) are owned by NATIONS under the laws and treaties of the People's Republic of China and other applicable jurisdictions worldwide.

NATIONS does not grant any license under its patents, copyrights, trademarks, or other intellectual property rights. Names and brands of third party may be mentioned or referred thereto (if any) for identification purposes only.

NATIONS reserves the right to make changes, corrections, enhancements, modifications, and improvements to this document at any time without notice. Please contact NATIONS and obtain the latest version of this document before placing orders.

Although NATIONS has attempted to provide accurate and reliable information, NATIONS assumes no responsibility for the accuracy and reliability of this document.

It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. In no event shall NATIONS be liable for any direct, indirect, incidental, special, exemplary, or consequential damages arising in any way out of the use of this document or the Product.

NATIONS Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at user's risk. User shall indemnify NATIONS and hold NATIONS harmless from and against all claims, costs, damages, and other liabilities, arising from or related to any customer's Insecure Usage.

Any express or implied warranty with regard to this document or the Product, including, but not limited to, the warranties of merchantability, fitness for a particular purpose and non-infringement are disclaimed to the fullest extent permitted by law.

Unless otherwise explicitly permitted by NATIONS, anyone may not use, duplicate, modify, transcribe or otherwise distribute this document for any purposes, in whole or in part.