NV32F100x 时钟模块编程示例

第一章 库函数说明

1.1 库函数列表

void OSC_Init(OSC_ConfigType *pConfig)

通过结构体 OSC_ConfigType 初始化 OSC 函数

void OSC_DeInit(void)

OSC 恢复默认状态函数

void ICS_SetClkDivider(uint32_t u32ClkFreqKHz)

外部参考时钟分频函函数

void FEI_to_FEE(ICS_ConfigType *pConfig)

ICS 的工作模式由当前 FEI 模式切换到 FEE 模式，OSC 模块的输出选择振荡器时钟源。

void FEI_to_FBI(ICS_ConfigType *pConfig)

ICS 的工作模式由当前 FEI 模式切换到 FBI 模式。

void FEI_to_FBE(ICS_ConfigType *pConfig)

ICS 的工作模式由当前 FEI 模式切换到 FBE 模式，OSC 模块的输出选择振荡器时钟源

void FEI_to_FBE_OSC(ICS_ConfigType *pConfig)

ICS 的工作模式由 FEI 模式切换到 FBE 模式，OSC 模块的输出选择输入 EXTAL 引脚的外部时钟源

void FEI_to_FEE_OSC(ICS_ConfigType *pConfig)

ICS 的工作模式由 FEI 模式切换到 FEE 模式，OSC 模块的输出选择输入 EXTAL 引脚的外部时钟源

void FEE_to_FEI(ICS_ConfigType *pConfig)

ICS 的工作模式由当前 FEE 模式切换到 FEI 模式

void FEE_to_FBI(ICS_ConfigType *pConfig)

ICS 的工作模式由当前 FEE 模式切换到 FBI 模式

void FEE_to_FBE(ICS_ConfigType *pConfig)

ICS 的工作模式由当前 FEE 模式切换到 FBE 模式

void FBI_to_FBE(ICS_ConfigType *pConfig)

ICS 的工作模式由当前 FBI 模式切换到 FBE 模式

void FBI_to_FEE(ICS_ConfigType *pConfig)

ICS 的工作模式由当前 FBI 模式切换到 FEE 模式

void FBI_to_FBILP(ICS_ConfigType *pConfig)

ICS 的工作模式由当前 FBI 模式切换到 FBILP 模式

void FBI_to_FEI(ICS_ConfigType *pConfig)

ICS 的工作模式由当前 FBI 模式切换到 FEI 模式

void FBE_to_FBI(ICS_ConfigType *pConfig)

ICS 的工作模式由当前 FBE 模式切换到 FBI 模式

void FBE_to_FEE(ICS_ConfigType *pConfig)

ICS 的工作模式由当前 FBE 模式切换到 FEE 模式

void FBE_to_FEI(ICS_ConfigType *pConfig)

ICS 的工作模式由当前 FBE 模式切换到 FEI 模式

void FBELP_to_FBE(ICS_ConfigType *pConfig)

ICS 的工作模式由当前 FBELP 模式切换到 FBE 模式

void FBE_to_FBELP(ICS_ConfigType *pConfig)

ICS 的工作模式由当前 FBE 模式切换到 FBELP 模式

void FBILP_to_FBI(ICS_ConfigType *pConfig)

ICS 的工作模式由当前 FBELP 模式切换到 FBI 模式

void ICS_Trim(uint16_t u16TrimValue)

内部时钟调整函数（IRC）

void ICS_Init(ICS_ConfigType *pConfig)

通过结构体 ICS_ConfigType 初始化 ICS 函数

void ICS_DeInit(void)

ICS 模块恢复默认状态函数

寄存器操作的内联函数，调用内联函数和直接操作寄存器的效率一样

void ICS_EnableInt(void)

Loss of lock 中断使能

void ICS_DisableInt(void)

禁用 Loss of lock 中断

void ICS_EnableClockMonitor(void)

使能时钟模拟

void ICS_DisableClockMonitor(void)

禁用时钟模拟

void ICS_SetBusDivider(uint8_t u8BusDivide)

对选中的输出时钟源进行分频

void OSC_Enable(void)

使能 OSC 模块

void OSC_Disable(void)

禁用 OSC 模块

void OSC_SetLowRange(void)

选择低频范围

void OSC_SetHighRange(void)

选择高频范围

void OSC_SetHighGain(void)

选择高增益模式

void OSC_SetLowGain(void)

选择低功耗模式

void OSC_SelectCrystal(void)

OSC 输出模块选择振荡器时钟源

void OSC_SelectClock(void)

OSC 输出时钟选择来自 EXTAL 引脚的外部时钟源

void OSC_ActiveInStop(void)

OSC 在停止模式下保持使能

void OSC_InactiveInStop(void)

OSC 时钟停止模式下禁用

1.2 振荡器 OSC 模块特性

*支持 32kHz 晶振（低范围模式）

*支持 4－48 MHz 晶振和谐振器（高范围模式）

*自动增益控制(AGC)，可使用低功耗模式优化两个频率范围内的功耗（低增益模式)

*两种频率范围内均具有高增益选项：32 kHz，4－48 MHz

*电压和频率滤波功能，确保时钟频率和稳定性

*支持通过 ICS 使能

1.3 内部时钟 ICS 模块特性

*内置 FLL，倍频为 1280

*内部时钟和外部时钟源都可以作为 FLL 的参考输入

*针对外部时钟提供了基准分频器

*外部时钟需要先调整到 31.25－39.0625 kHz 之间

*内部时钟源有 9 位有效修调位

*内部和外部时钟源都可以作为 MCU 的时钟参考源

*无论选择哪个时钟作为时钟源，都可以对其进行分频

- ❖ 针对时钟分频器提供 3 位选择

- ❖ 可用分频比为：1、2、4、8、16、32、64、128（OSC_CR[RANGE]=0)；32、64、128、256、512、1024（OSC_CR[RANGE]=1)

*FLL 内部启用模式在退出复位后被自动选中

*可选数字控制振荡器(DCO)优化的频率范围

*FLL 锁定检测器和外部时钟监控器

- ❖ 具有中断功能的 FLL 锁定检测器
- ❖ 具有复位功能的外部参考时钟监视器

*数字控制振荡器针对 40-48MHz 频率范围进行了优化

1.4 OSC 模块使用说明

*选择输出时钟的来自振荡器时钟还是 EXTAL 引脚的外部时钟源

*选择外部时钟的频率范围

*设置 OSC 的工作模式，低功耗和高增益

OSC 模块的具体设置请参阅参考手册

1.5 ICS 模块使用说明

*选择 ICS 的工作模式（具体包括 ISC 输出时钟源的选择、FLL 输入时钟源的选择和低功耗选择）

*当选择外部时钟，设置 RDIV 对外部时钟分频

*设置 BDIV 对选中 ICS 输出时钟源进行分频

ICS 模块具体时钟配置请参阅参考手册

第二章 时钟模式配置

2.1 OSC 模块初始化

2.1.1 OSC 控制寄存器（OSC_CR）

位	描述
7 OSCEN	OSC 使能 使能 OSC 模块。OSC 模块也可通过 ICS 模块使能。 0 OSC 模块禁用。 1 OSC 模块使能。
6 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。

5 OSCSTEN	停止模式下的 OSC 使能 控制 OSC 时钟在 MCU 进入停止模式且 OSCEN 置位时是否保持使能。如果 ICS 请求 OSC 使能，则 OSCSTEN 无效。 0 OSC 时钟在停止模式下禁用。 1 OSC 时钟在停止模式下保持使能。
4 OSCOS	OSC 输出选择 选择 OSC 模块的输出时钟。 0 选择来自 EXTAL 引脚的外部时钟源。 1 选择振荡器时钟源。
3 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
2 RANGE	频率范围选择 选择 OSC 模块的频率范围。 0 32 kHz 的低频范围。 1 4–48MHz 的高频范围。
1 HGO	高增益振荡器选择 控制 OSC 工作模式。 0 低功耗模式 1 高增益模式
0 OSCINIT	OSC 初始化 该字段在振荡器初始化周期完成后置位。 0 振荡器初始化未完成。 1 振荡器初始化已完成。

函数名	OSC_Init
函数原形	OSC_Init(OSC_ConfigType *pConfig)
功能描述	配置结构体 pConfig 来初始化 OSC
输入参数	OSC 的配置结构体 OSC_ConfigType
输出参数	无
返回值	无
先决条件	无
函数使用实例	先设置 OSC 配置结构体，OSC_Init(&pConfig->oscConfig)

```
/*****************************************************************************//*!
    *
    * @ 概要  使用给定的参数初始化 XOSC
    *
    * @ 参数【输入】    pConfig     指向 osc 配置结构体
    *
    * @ 无返回
    *
    *****************************************************************************/
void OSC_Init(OSC_ConfigType *pConfig)
```

```
{
    uint8    cr = 0;
    if(pConfig->bGain)    /*高增益振荡器选择*/
    {
        /* 选择高增益模式 */
        cr |= OSC_CR_HGO_MASK ;          /* 变阻器必须增加到 200K */
    }
    if(pConfig->bRange)        /*频率范围的选择*/
    {
        cr |= OSC_CR_RANGE_MASK; /*选择高频范围*/
    }
    if(pConfig->bStopEnable)    /*停止模式下的 OSC 使能*/
    {
        cr |= OSC_CR_OSCSTEN_MASK;   /*OSC 在停止模式下保持使能*/
    }
    if(pConfig->bIsCryst)       /*OSC 输出选择*/
    {
        cr |= OSC_CR_OSCOS_MASK;      /*选择振荡器时钟*/
    }
    if(pConfig->bEnable)       /*OSC 使能*/
    {
        cr |= OSC_CR_OSCEN_MASK;
    }
    OSC->CR = cr;           /*数值写入控制寄存器*/

    if(pConfig->bWaitInit)   /*等待初始化完成*/
    {
        while(!(OSC->CR & OSC_CR_OSCINIT_MASK));
    }
}
```

2.2 外部时钟源分频

当选择外部时钟作为 FLL 的输入时钟源时，通过对 ICS_C1 寄存器 RDIV 进行配置，实现外部时钟分频，ICS_C1 寄存器，详细内容请查看参考手册

5-3 RDIV	参考时钟分频系数，参考时钟的分频结果必须是 31.25 – 39.0625 kHz		
		OSC_CR[RANGE]=0	OSC_CR[RANGE]=1
	000	1	32
	001	2	64
	010	4	128
	011	8	256

	100	16	512
	101	32	1024
	110	64	2048
	111	128	保留

对外部参考时钟进行分频，使得分频结果在 31.25 – 39.0625 kHz 内，在函数中已定义一些外部时钟频率值，如要使用其他外部时钟频率，可在此函数中添加，外部时钟频率宏观定义在 nv32.config.h 文件中

函数名	ICS_SetClkDivide
函数原形	ICS_SetClkDivider(uint32_t u32ClkFreqKHz)
功能描述	对输入 FLL 锁频环的外部时钟进行分频
输入参数	外部参考时钟频率：u32ClkFreqKHz
输出参数	无
返回值	无
先决条件	ICS 初始化中选择外部时钟作为 FLL 的输入时钟源
函数使用实例	先设置外部时钟频率 ， ICS_SetClkDivider(pConfig->u32ClkFreq);

```
/*****************************************************************************************//*!
 *
 * @ 概要 对外部参考时钟进行分频，使得分频结果在 31.25 – 39.0625 kHz 内
 *
 * @ 参数【输入】 u32ClkFreqKHz      参考时钟频率.
 *
 * @  无返回
 *
*****************************************************************************************/
void ICS_SetClkDivider(uint32_t u32ClkFreqKHz)
{
    switch(u32ClkFreqKHz)
    {
        case 8000L:
        case 10000L:
            /* 8MHz or 10MHz */
            ICS->C1 = (ICS->C1 & ~(ICS_C1_RDIV_MASK)) | ICS_C1_RDIV(3);/* 现在分频结果是 8000/256 = 31.25K */
             /* 分频结果 10000/256 = 39.0625K */
            Break;
            break;
        case 4000L:
            /* 4MHz */
            ICS->C1 = (ICS->C1 & ~(ICS_C1_RDIV_MASK)) | ICS_C1_RDIV(2);/* 分频结果 4000/128 = 31.25K */
            break;
        case 16000L:
```

```
            /* 16MHz */
            ICS->C1 = (ICS->C1 & ~(ICS_C1_RDIV_MASK)) | ICS_C1_RDIV(4);/*  分频结果  16000/512 = 31.25K */
            break;
        case 20000L:
            /* 20MHz */
            ICS->C1 = (ICS->C1 & ~(ICS_C1_RDIV_MASK)) | ICS_C1_RDIV(4);  /* 分频结果 20000/512 = 39.0625K */
            break;
        case 32L:
            /* 32KHz */
            ICS->C1   &= ~(ICS_C1_RDIV_MASK);
            break;
        default:
            break;
    }
}
```

2.3 时钟模式选择

ISC 的时钟模式通过 ICS_CI 寄存器 CLKS、IRFS 位和 ICS_C2 寄存器 LP 位进行选择

ICS_C1 寄存器，详细内容请查看参考手册

位	描述
7-6 CLKS	输出时钟源选择 00 FLL 的输出 01 内部时钟源 10 外部时钟源 11 保留，默认值为 00。
2 IREFS	FLL 参考时钟源选择 0 选择外部参考时钟源 1 选择内部参考时钟源
1 IRCLKEN	内部 IRC 时钟源使能 0 内部 IRC 不工作 1 内部 IRC 工作
0 IREFSTEN	内部参考时钟源 STOP 模式下是否有效 0 在 STOP 模式下内部时钟源不工作 1 如果 IRCLKEN 有效，或者在 FEI,FBI,FBILP 进入 STOP 模式，在 STOP 模式下内部时钟源保持使能

ICS_C2 寄存器

位	描述

7-5 BDIV	内部时钟源分频参数 000 对选中的时钟源做 1 分频 001 对选中的时钟源做 2 分频（复位初始值） 010 对选中的时钟源做 4 分频 011 对选中的时钟源做 8 分频 100 对选中的时钟源做 16 分频 101 对选中的时钟源做 32 分频 110 对选中的时钟源做 64 分频 111 对选中的时钟源做 128 分频
4 LP	低功耗选择 0 在 bypass 模式 FLL 不会被禁止 1 除非 debug 模式，FLL 将会禁止在 bypass 模式。
3-0 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。

2.3.1FLL 内部启用（FEI）

FLL 内部启用(FEI)模式是默认工作模式，该模式下 ICSOUT 时钟从 FLL 时钟获得，FLL 时钟由内部参考时钟控制。FLL 环路将频率锁定到内部参考时钟频率的 1280 倍。内部参考时钟源已启用。寄存器配置如下；

（1）00b 写入 ICS_C1[CLKS]：选择 FLL 的输出作为 ICSOUT 输出参考时钟。

（2）1b 写入 ICS_C1[IREFS]：选择内部时钟作为 FLL 参考时钟。

FLL 输出时钟频率=内部参考时钟频率*1280

ICS 输出时钟频率=FLL 输出时钟频率/BDIV

片上 RC 振荡器，范围为 31.25 – 39.0625 kHz（MCU 出厂校准值为 37.5KHz），作为 FLL 输入的基准

表 3-1 基于内部参考的可行的的 ICS 总线频率

基准		ICSOUTCLK
FEI(高范围）	BDIV=0	40MHz~50MHz[1]
	BDIV=1	20MHz~25MHz
	BDIV=2	10MHz~12.5MHz
	BDIV=4	5MHz~6.25MHz
	BDIV=8	2.5MHz~3.125MHz
	BDIV=16	1.25MHz~1.5625MHz
	BDIV=32	625kHz~781.25kHz
	BDIV=64	312.5kHz~390.625kHz
	BDIV=128	156.25kHz~195.3125kHz

注； FEI 模式下默认 BDIV=1，可在 ICS 初始化函数中更改 BDIV 分频系数

2.3.2 FLL 外部启用（FEE）

在“FLL 外部启用”模式下，ICSOUT 时钟从 FLL 时钟获得，FLL 时钟由外部参考时钟源控制。FLL 环路将 FLL 频率锁定到外部基准频率（由 ICS_C1[RDIV]和 OSC_CR[RANGE]选择）的 1280 倍。外部参考时钟源已启用。寄存器配置如下；

（1）00b 写入 ICS_C1[CLKS]：选择 FLL 的输出作为 ICSOUT 的输出参考时钟

（2）0b 写入 ICS_C1[IREFS]：选择外部时钟作为 FLL 的参考时钟

（3）对 ICS_C1[RDIV]和 OSC_CR[RANGE]进行写操作可将外部参考时钟分频到 31.25kHz 至 39.0625kHz 范围内。外部参考时钟的分频结果必须在 31.25 – 39.0625 kHz 之间，这样 FLL 才可以锁频。

FLL 的输出时钟频率=外部参考时钟/RDIV*1280

ICS 输出时钟频率=FLL 的输出时钟频率/BDIV

该模式下 ICS 输出时钟的计算如下；

将 ICS 的时钟模式配置为 FEE 模式，选择外部时钟时钟频率为 16MHz;由于需要将外部时钟频率分频到 26K~45K 之间，则向 ICS_CI[RDIV]寄存器写入 100B，对外部时钟进行 512 倍分频。

输入 FLL 的外部基准频率=16M/512=31.25K

FLL 输出频率=31.25K*1280=40M

通过对 ICS_C2[BDIV]寄存器写入数值，实现对选中的 ICS 输出时钟源进行分频，假如写入 ICS_C2[BDIV]寄存器的数值为 001B，则对时钟源进行 2 分频。

ICS 输出时钟频率=40M/2=20M

函数名	FEI_to_FEE
函数原形	FEI_to_FEE(ICS_ConfigType *pConfig)
功能描述	使 ICS 的工作模式由当前的 FEI 模式切换到 FEE 模式
输入参数	ICS 的配置结构体 ICS_ConfigType
输出参数	无
返回值	无
先决条件	当前的工作模式位 FEI 模式
函数使用实例	先设置 ICS 配置结构体 ， FEI_to_FEE(pConfig)

```
/*****************************************************************************//*!
    *
    * @ ICS 的工作模式由当前的 FEI 模式切换为 FEE 模式，对选中的输出时钟源做 2 分频
    *
    * @ OSC 模块的输出时钟选择振荡器时钟源
    *
    * @ 输入 pConfig 指向 ICS 配置结构体
    *
    * @ 无返回
    *
    *****************************************************************************/
```

```
void FEI_to_FEE(ICS_ConfigType *pConfig)
{
    /* 使能 OSC */
    OSC_Init(&pConfig->oscConfig); /*OSC 模块初始化*/ OSC 模块的输出时钟选择振荡器时钟源
    /* 对外部参考时钟进行分频，可将外部时钟分频到 31.25k~39.0625k 之间*/
    ICS_SetClkDivider(pConfig->u32ClkFreq);
    /*将 FLL 的参考时钟选择为外部时钟*/
    ICS->C1 =   ICS->C1 & ~ICS_C1_IREFS_MASK;
    /*等待 FLL 参考时钟变为外部时钟*/
   #if defined(IAR)
    asm(
    "nop \n"
    "nop \n"
     );
  #elif defined(__MWERKS__)
     asm{
     nop
     nop
     };
  #endif
 while(ICS->S & ICS_S_IREFST_MASK);
/* 等待 FLL 时钟成为 ICS 输出时钟源*/
 while(!(ICS->S & ICS_S_LOCK_MASK));
/* 现在 FLL 输出时钟变时钟频率等于 FLL 参考时钟分频结果乘以 FLL 的倍频系数 FLL 的倍频系数请参考参考手册 */
#if defined(CPU_NV32)
 /*对选中的 ICS 输出时钟源做 2 分频*/
    if(((ICS->C2 & ICS_C2_BDIV_MASK)>>5) != 1)
    {
        ICS->C2 = (ICS->C2 & ~(ICS_C2_BDIV_MASK)) | ICS_C2_BDIV(1);
    }
#else
    ICS->C2 = (ICS->C2 & ~(ICS_C2_BDIV_MASK)) | ICS_C2_BDIV(0);
#endif
    /*  完成对选中的 ICS 输出时钟源做 2 分频，系统/总线时钟时频率为设置的目标频率*/
    /*lols 清 0*/
    ICS->S |= ICS_S_LOLS_MASK;
}
```

注：void FEI_to_FEE(ICS_ConfigType *pConfig)和 void FEI_to_FEE_OSC(ICS_ConfigType *pConfig)的区别为 OSC 初始化时，OCS 的输出选择不同。

2.3.3 FLL 内部旁路（FBI）

在“FLL 内部旁路”模式下，ICSOUT 时钟从内部参考时钟获得。FLL 时钟由内部参考时钟控制，FLL 循环将 FLL 频率锁定为内部参考频率的 1280 倍。内部参考时钟源已启用。寄存器的配置如下；

（1）01b 写入 ICS_C1[CLKS]：ICSOUT 的输出时钟选择内部时钟

（2）1b 写入 ICS_C1[IREFS]：FLL 参考时钟选择内部时钟

FLL 输出时钟频率=内部参考时钟*1280

ICS 输出时钟频率=内部参考时钟/BDIV

2.3.4 FLL 内部旁路低功耗（FBILP）

在“FLL 内部旁路低功耗”模式下，ICSOUT 时钟从内部参考时钟获得，FLL 禁用。内部参考时钟源已启用。寄存器配置如下；

（1）01b 写入 ICS_C1[CLKS]：ICS 输出选择内部时钟

（2）1b 写入 ICS_C1[IREFS]：FLL 选择内部参考时钟

（1）1b 写入 ICS_C2[LP]： FLL 禁用

ICS 输出时钟频率=内部参考时钟/BDIV

2.3.5 FLL 外部旁路（FBE）

在 FLL 外部旁通模式下，ICSOUT 时钟由外部参考时钟源分频而来。FLL 时钟由外部参考时钟控制，FLL 环路将 FLL 频率锁定到外部参考频率（由 ICS_C1[RDIV]和 OSC_CR[RANGE]选择）的 1280 倍。寄存器配置如下；

（1）10b 写入 ICS_C1[CLKS]：选择外部时钟源作为 ICS 输出时钟源

（2）0b 写入 ICS_C1[IREFS]：选择外部时钟源作为 FLL 参考时钟

（3）对 ICS_C1[RDIV]和 OSC_CR[RANGE]进行写操作可将外部参考时钟分 31.25kHz 至 39.0625kH 范围内。外部参考时钟的分频结果必须在 31.25 – 39.0625 kHz 之间。这样 FLL 才可以锁频。

FLL 的输出时钟频率=外部参考时钟/RDIV*1280

ISC 的输出时钟频率=外部参考时钟/BDIV

2.3.6 外部旁路低功耗（FBELP）

在“FLL 外部旁路低功耗”模式下，ICSOUT 时钟从外部参考时钟源获得，FLL 禁用。外部参考时钟源已启用。寄存器配置如下；

（1）10b 写入 ICS_C1[CLKS]：选择外部时钟源作为 ICS 的输出时钟源

（2）0b 写入 ICS_C1[IREFS]：选择外部时钟源作为 FLL 的参考时钟

（3）1b 写入 ICS_C2[LP]： FLL 禁用

ICS 输出时钟频率=外部参考时钟/BDIV

2.3.7 FLL 停止模式（STOP）

只要 MCU 进入 STOP 状态，就会进入停止模式。在该模式下，所有 ICS 时钟信号都保持静态，但例外情况如下：

出现以下所有条件时，ICSIRCLK 在停止模式下无效：

- 1b 写入 ICS_C1[IRCLKEN]。
- 1b 写入 ICS_C1[IREFSTEN]。

注意：DCO 频率从预停止值变为其复位值，FLL 需要在频率稳定前重新获取锁定。时序敏感的操作在执行前，必须等待 FLL 获取时间 tAcquire。

2.4 ICS 各工作模式的切换关系

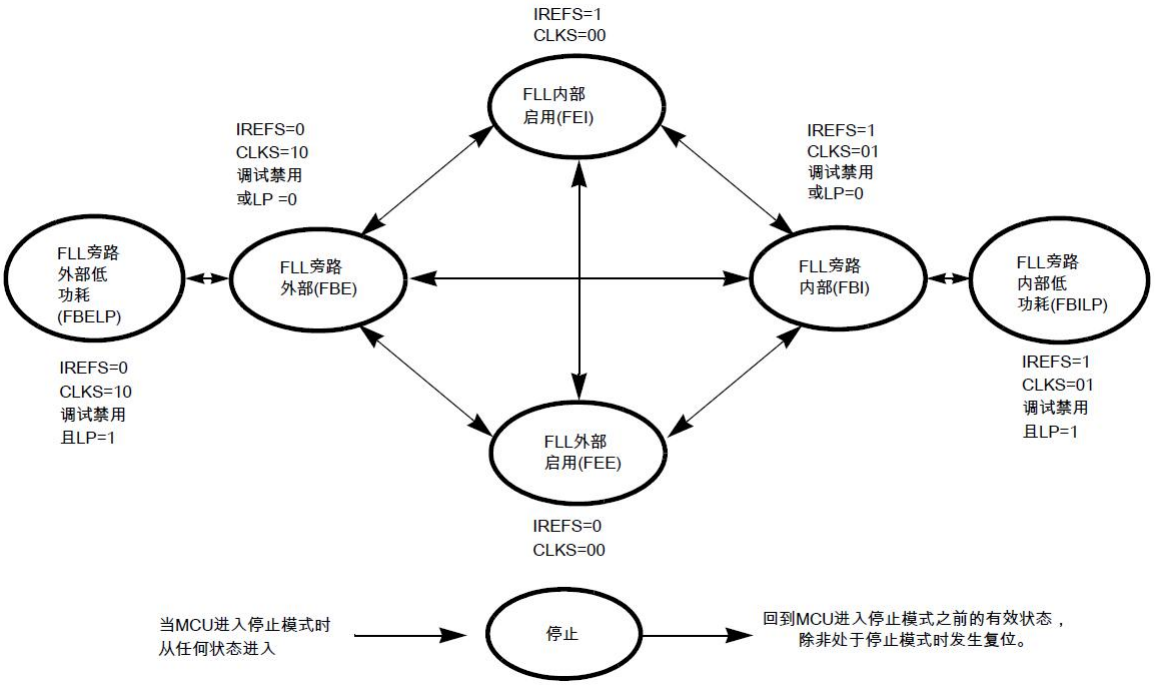


图 1-1 ICS 各工作方式切换

2.5 ICS 初始换函数

在 ICS 初始化函数包括选择 ICS 的时钟模式，和配置 OCS 模块的结构体,有关时钟模式选择的宏观定义在 nv32,config.h 文件中，ICS 初始化函数在系统初始化函数中被调用。

函数名	ICS_Init
函数原形	ICS_Init(ICS_ConfigType *pConfig)
功能描述	配置结构体 pConfig 来初始化 ICS
输入参数	ICS 的配置结构体 ICS_ConfigType
输出参数	无
返回值	无

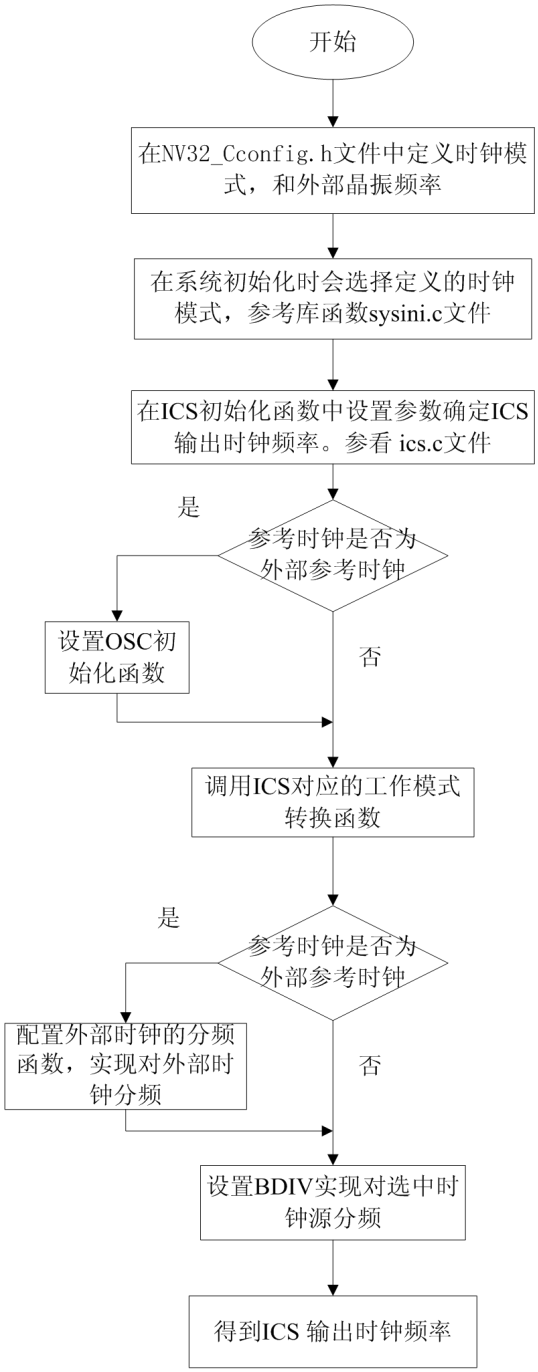
先决条件	无
函数使用实例	先设置 ICS 配置结构体 ， ICS_Init(&sICSConfig);

```
/*****************************************************************************//*!
*
* @ 概要 初始化 ICS 模块定义所需要的总线时钟频率.
*
* @ 参数【输入】 pConfig  指向配置结构体.
*
* @ 无返回
*
******** ***************************************************************************/
void ICS_Init(ICS_ConfigType *pConfig)
{
  if(pConfig->u8ClkMode == ICS_CLK_MODE_FEE)
  {
       /*初始化 OSC 模块结构体*/
        pConfig->oscConfig.bIsCryst = 1;          /*OSC 的输出选择选择振动器时钟源*/
        pConfig->oscConfig.bWaitInit = 1;         /* 等待振荡器初始化化完成 */
       FEI_to_FEE(pConfig);                       /*选择 FEE 模式，使用振荡器时钟源*/
  }
  else if (pConfig->u8ClkMode == ICS_CLK_MODE_FEE_OSC)
  {
        /*初始化 OSC 模块结构体*/
        pConfig->oscConfig.bIsCryst = 0;          /*OSC 输出时钟选择 EXTAL 引脚的外部时钟源*/
        FEI_to_FEE_OSC(pConfig);                  /*选择 FEE 工作模式*/
  }
  else if (pConfig->u8ClkMode == ICS_CLK_MODE_FBE_OSC)
  {

       /*初始化 OSC 模块结构体*/
        pConfig->oscConfig.bIsCryst = 0;          /* OSC 输出时钟选择 EXTAL 引脚的外部时钟源*/
        FEI_to_FBE_OSC(pConfig);                  /* 选择 FBE 工作模式*/
  }
```

```
else if(pConfig->u8ClkMode == ICS_CLK_MODE_FBELP )
        {

        /*初始化 OSC 模块结构体*/
        pConfig->oscConfig.bIsCryst = 1;          /*OSC 的输出时钟选择选择振动器时钟源*/
        pConfig->oscConfig.bWaitInit = 1;          /*等待振荡器初始化化完成 */
        FEI_to_FBE(pConfig);                       /*选择 PBE 模式*/
        FBE_to_FBELP(pConfig);                     /*选择 FBELP*/
       ICS->C2 = (ICS->C2 & ~(ICS_C2_BDIV_MASK)) | ICS_C2_BDIV(0);
       }
 else if(pConfig->u8ClkMode == ICS_CLK_MODE_FBILP )
        {
       /*初始化 OSC 模块结构体*/
        pConfig->oscConfig.bIsCryst = 0;          /* OSC 输出时钟选择 EXTAL 引脚的外部时钟源*/
        FEI_to_FBI(pConfig);                      /*选择 FBI 工作模式*/
        FBI_to_FBILP(pConfig);                    /*选择 FBILP 工作模式*/
        ICS->C2 = (ICS->C2 & ~(ICS_C2_BDIV_MASK)) | ICS_C2_BDIV(0);
        }
  else
  {
        /*ICS 默认工作模式 FEI 模式，FEI 模式下 BDIV 的分频系数在此设置*/
        #if defined(CPU_NV32)          /*对所选时钟源进行 2 分频*/
        if(((ICS->C2 & ICS_C2_BDIV_MASK)>>5) != 1)
        {
            ICS->C2 = (ICS->C2 & ~(ICS_C2_BDIV_MASK)) | ICS_C2_BDIV(1);
        }
        #else
            ICS->C2 = (ICS->C2 & ~(ICS_C2_BDIV_MASK)) | ICS_C2_BDIV(0);
        #endif
          ICS->C2 = (ICS->C2 & ~(ICS_C2_BDIV_MASK)) | ICS_C2_BDIV(0);
  }
     ICS->C2 = (ICS->C2 & ~(ICS_C2_BDIV_MASK)) | ICS_C2_BDIV(0);
}
```

2.6 时钟具体配置

在库函数中关于时钟模式的相关宏定义在库函数 NV32_Cconfig.h 文件中，可在该文件中定义时钟模式，外部晶振频率、总线时钟；时钟模式配置过程如下；



ICS 系统初始化时默认的模式为 FEI 模式，可根据对寄存器的配置实现 ICS 工作模式的转换。ICS 各模式之间的切换关系如图 1-1 所示。现以配置 ICS 的工作模式由 FEI 模式转换成 FEE 模式为例。其他模式间的准换与此类似。

配置 FEE 模式的具体操作具体操作如下；

1，在首先在 NV32_Cconfig.h 文件中选择定义外部是时钟（#define USE_FEE）和外部时钟的晶振频率（ #define EXT_CLK_FREQ_KHZ 10000）；

2. 系统初始化时（sysinit.c），系统会选择定义的工作模式，在 ICS 初始化函数中根据定义的模式选择 ICS 工作模式。参考 ICS 初始化函数

3. 当 ICS 选择 FEE 模式后，首先是对 OCS 模块的初始化设置。参考 OCS 初始化函数。

4. 经过系统的初始化，ClkFreqKHz= EXT_CLK_FREQ_KHZ， 分频函数 ICS_SetClkDivider()将对外部时钟晶振频率进行分频，对外部时钟的分频结果必须在 26K~40k 之间，因此分频系数的选 ICS_C1_RDIV(x)的选择要要根据自己定义的外部时钟晶振频率决定。参考 ICS_SetClkDivider()函数。

例如当选择的时钟晶振频率为 10Mhz 时，对其进行 256 倍时钟分频，分频结果；10000/256 = 39.0625K 外部时钟其他晶振频率的分频，可在分频函数中进行配置；

5. ICS 初始化最后是调用模式转换函数；由 FEI 模式转化成 FEE 模式，在模式转换函数中通过设定输出时钟源分频系数确定 ICS 输出的时钟频率。参考 FEI 模式转化成 FEE 模式转换函数

例；通过对 10MHz 的外部时钟分频得到分频结果为；10000/256=39.0625k;

内部时钟源分频系数为 2；则 ICS 输出时钟频率为； 39.0625*1280/2=25M

6.常用的外部晶振

外部晶振频率	目标频率	RDIV	FLL 输出频率	BDIV	ICS 工作模式
4M	40M	128	40M	0X00	FEE
4M	10M	128	40M	0X02	FEE
4M	5M	128	40M	0X03	FEE
4M	4M	128	40M	0X00	FBE
4M	4M	0	0	0X00	FBE_LP
8M	20M	256	40M	0X01	FEE
9.6M	48M	256	48M	0X00	FEE
9.6M	24M	256	48M	0X01	FEE
10M	50M	256	50M	0X00	FEE
12M	60M	256	60M	0X00	FEE*
48M	48M	0	0	0X00	FBE_LP

*不建议使用

注意：设置好 RDIV 和 BDIV 需要将 BUS_CLK_HZ 改成你算出来的值

例如:

```
#if defined(USE_FEI)
#define BUS_CLK_HZ 48000000L //ics trim 值为 0x25 RDIV=0,BDIV=0
```