# NV32F100x 系统初始化配置说明

# 1. NV32 系统初始化流程及相应配置

## 1.1 NV32 系统初始化函数-Sysinit

```
/*****************************************************************************//*!
*
* @NV32 的系统初始化函数，配置 FLASH 等待周期，管脚复用，时钟选择等。
*
*****************************************************************************/
void sysinit (void)
{
    SIM_ConfigType    sSIMConfig = {{0},0};
    ICS_ConfigType    sICSConfig = {0};

     global_pass_count = 0;
     global_fail_count = 0;

EFMCR &=0xFFFF0001; //  设置 FLASH 等待周期，注：此处需修正
#if defined(TRIM_IRC)
    ICS_Trim(ICS_TRIM_VALUE);//粗略的 TRIM 值校准，若不使用此值则使用内部出厂 40M 主频
#endif
/*
定义一些功能管脚的设置，比如禁用复位脚 RESET,以及 SWD 调试，通过宏定义的方式
RST 脚与 NMI 脚在上电后只能配置一次，第二次即视为无效
注：要考虑到这些管脚复用的问题，在 Sysinit 函数中，必须一开始就要禁用
*/
#if defined(DISABLE_RST)
    sSIMConfig.sBits.bDisableRESET = 1;//禁用 RST 脚
#endif
#if defined(DISABLE_SWD)
    sSIMConfig.sBits.bDisableSWD = 1;//禁用 SWD 调试，注：慎用，禁用则上电后不可再通过 SWD 调试
#endif
#if defined(SPI0_PINREMAP)
    sSIMConfig.u32PinSel |= SIM_PINSEL_SPI0PS_MASK;
#endif

/*   输出总线时钟，定义为管脚 PH2 输出  */
#if defined(OUTPUT_BUSCLK)
sSIMConfig.sBits.bEnableCLKOUT = 1;
#endif
#if defined(DISABLE_NMI)
    sSIMConfig.sBits.bDisableNMI = 1;//禁用不可屏蔽中断的管脚 PB4
#endif
```

```
    /* 使能部分模块的时钟信号 */
      sSIMConfig.u32SCGC|=SIM_SCGC_SWD_MASK|SIM_SCGC_FLASH_MASK|
SIM_SCGC_UART0_MASK | SIM_SCGC_UART1_MASK | SIM_SCGC_UART2_MASK;
    /*初始化 SIM 模块*/
      SIM_Init(&sSIMConfig);

#if defined(XOSC_STOP_ENABLE)
sICSConfig.oscConfig.bStopEnable = 1;
#endif
#if defined(CRYST_HIGH_GAIN)
      sICSConfig.oscConfig.bGain = 1;
#endif
#if   (EXT_CLK_FREQ_KHZ >=4000)
      sICSConfig.oscConfig.bRange = 1;                //OSC_CR[RANGE]置位
#endif
      sICSConfig.oscConfig.bEnable = 1;               //使能 OSC
      sICSConfig.u32ClkFreq = EXT_CLK_FREQ_KHZ;
#if   defined(USE_FEE)                                //选择外部晶振时钟，常用的两种时钟模式为 FEE 和 FEI
      sICSConfig.u8ClkMode = ICS_CLK_MODE_FEE;
#elif defined(USE_FBE_OSC)
      sICSConfig.u8ClkMode = ICS_CLK_MODE_FBE_OSC;
#elif defined(USE_FEE_OSC)
      sICSConfig.u8ClkMode = ICS_CLK_MODE_FEE_OSC;
#elif defined(USE_FBILP)
      sICSConfig.u8ClkMode = ICS_CLK_MODE_FBILP;
#elif defined(USE_FBELP)
      sICSConfig.u8ClkMode = ICS_CLK_MODE_FBELP;
#endif
       /*初始化 ICS 模块 */
      ICS_Init(&sICSConfig);
      /* 初始化 UART 打印串口输出 */
      UART_InitPrint();

#if   defined(PRINT_SYS_LOG)
      print_sys_log();   //打印系统相关的信息
#endif
  }
```

## 1.2 头文件 NV32Config.h 的说明

```
/*********************************************************************************
*
* NOTE:系统所使用的一些宏定义，以及时钟模式的选择。
*
*********************************************************************************/
#ifndef _NVxx_CONFIG_H_
#define _NVxx_CONFIG_H_

#include <stdint.h>
#define CPU_NV32
#define TEST

/* 是否使用定义的 TRIM 值来校准内部 IRC，若注释则使用出厂校准的 TRIM 值出厂校准至 31.25K-1280 倍频->40M  */
//#define TRIM_IRC
#define ICS_TRIM_VALUE   0x50   //由是否定义 TRIM_IRC 决定，此值粗略为 40M
#define SPI0_PINREMAP     //SPI0 的 SCK、MOSI、MISO 和 PCS 映射到 PE0、PE1、PE2 和 PE3 上
#define ENABLE_WDOG     //禁用看门狗，但允许其更新配置
/*! 禁用 NMI 中断输入引脚-PB4,该引脚建议布板外接 4.7K-10K 的上拉，以及上电时该引脚不可为低(若不使用 NMI 功能) */
#define DISABLE_NMI
/*! 定义是否打印系统相关信息 */
#define PRINT_SYS_LOG
/*! 定义是否输出系统时钟，输出引脚为 PH2 */
//#define OUTPUT_BUSCLK

/*********************************************************************************
*
* 根据所选的时钟模式，及时钟分频系数确定最终的总线时钟 BUSCLK 和系统/内核时钟 ICSOUT
* 内部时钟 FEI 模式：内核/系统时钟=IRC*1280/BDIV ： 总线时钟=IRC*1280/BDIV/BUSDIV
* 外部时钟 FEE 模式：内核/系统时钟=晶振/RDIV*1280/BDIV :总线时钟=晶振/RDIV*1280/BDIV/BUSDIV
* 其他时钟模式频率的计算请参考：库函数说明及参考手册 ICS 部分
*
 *********************************************************************************/

/*! 定义时钟的时钟模式以及频率 */
//#define USE_FEE                    //使用外部时钟 FEE 模式，频率参考晶体振荡器
//#define USE_FEE_OSC                //使用外部时钟输入 OSC 模式，频率参考 EXTAL-PB7
 #define USE_FEI                    //使用系统内部时钟 40M 作为主频
//#define USE_FBELP                  //外部旁路低功耗模式，禁用倍频环 FLL，频率直接由晶振提供
//#define USE_FBE_OSC
```

```
/*! 定义 BDIV 分频系数 (BDIV) */
/*
 * BDIV_ENCODE:    0   1   2   3   4    5    6    7
 *
 * BDIV--分频系数:    1   2   4   8   16   32   64   128
 */
#define BDIV_ENCODE          0                    //BDIV 分频系数为 1

/*! 定义 BUSDIV 分频系数(BUSDIV) */
/*
 * BUSDIV_ENCODE        0     1
 *
 * BUSCLK=ICSOUT/x      1     2
 */
//此处建议不要修改，即系统/内核时钟=总线时钟
#define BUSDIV_ENCODE            0          //总线时钟=ICSOUT/1

/*! 若使用外部晶振，则需要定义外部晶振频率，如下使用板载 10M 晶振*/
//#define EXT_CLK_FREQ_KHZ 32              /* in KHz */
//#define EXT_CLK_FREQ_KHZ 4000            /* in KHz */
//#define EXT_CLK_FREQ_KHZ 8000            /* in KHz */
 #define EXT_CLK_FREQ_KHZ 10000            /* in KHz */
//#define EXT_CLK_FREQ_KHZ 16000           /* in KHz */
//#define EXT_CLK_FREQ_KHZ 20000           /* in KHz */

例：使用 FEI 模式
    内核时钟/系统时钟 = ICSOUT=31.25K*1280/1=40M，
    总线时钟 = BUSCLK = ICSOUT/1 = 40M

/* 定义总线时钟主频 */
//一、前提：若定义使用 FEI 内部时钟
#if  defined(USE_FEI)
    #define   BUS_CLK_HZ    40000000L     //#define BDIV_ENCODE      0 //BDIV 分频系数为 1
    //#define   BUS_CLK_HZ    20000000L     //#define BDIV_ENCODE      1 //BDIV 分频系数为 2
    //#define   BUS_CLK_HZ    10000000L     //#define BDIV_ENCODE      2 //BDIV 分频系数为 4
//二、前提：若定义使用外部时钟的模式，以下情况为使用通用 FEE 模式，关于其他模式的使用参见手册及库函数说明
#elif  (EXT_CLK_FREQ_KHZ == 10000)
    #define   BUS_CLK_HZ    25000000L     //#define BDIV_ENCODE      1 //BDIV 分频系数为 2
//NV32F100x 的主频需在 40M 以内
#elif  (EXT_CLK_FREQ_KHZ == 16000)
    #define   BUS_CLK_HZ    25000000L     //#define BDIV_ENCODE      1 //BDIV 分频系数为 2
//NV32F100x 的主频需在 40M 以内
#elif  (EXT_CLK_FREQ_KHZ == 8000)
```

```
    #define   BUS_CLK_HZ      40000000L       //#define BDIV_ENCODE       0 //BDIV 分频系数为 1
    //#define   BUS_CLK_HZ    20000000L       //#define BDIV_ENCODE       1 //BDIV 分频系数为 2
#elif   (EXT_CLK_FREQ_KHZ == 4000)
    #define   BUS_CLK_HZ      40000000L       //#define BDIV_ENCODE       0 //BDIV 分频系数为 1
    //#define   BUS_CLK_HZ    20000000L       //#define BDIV_ENCODE       1 //BDIV 分频系数为 2
#elif   (EXT_CLK_FREQ_KHZ == 32)
    #define   BUS_CLK_HZ      16777216L
//三、定义了其他模式
#else
    #define   BUS_CLK_HZ      40000000L
#endif

/*! 定义所使用 UART 口的波特率 */
#define UART_PRINT_BITRATE                  115200          //UART 波特率
/*! 定义所使用的 UART 口,开发板上默认使用 UART1 口 */
#define TERM_PORT     UART1

/*! 板载 LED */
#define LED0_Init()                 GPIOB->PDDR |= (1<<25)     //初始化
#define LED0_Toggle()               GPIOB->PTOR    = (1<<25)   //翻转
#define LED0_On()                   GPIOB->PCOR    = (1<<25)   //置位
#define LED0_Off()                  GPIOB->PSOR    = (1<<25)   //清零

#define LED1_Init()                 GPIOB->PDDR |= (1<<26)
#define LED1_Toggle()               GPIOB->PTOR    = (1<<26)
#define LED1_On()                   GPIOB->PCOR    = (1<<26)
#define LED1_Off()                  GPIOB->PSOR    = (1<<26)

#define LED2_Init()                 GPIOB->PDDR |= (1<<7)
#define LED2_Toggle()               GPIOB->PTOR    = (1<<7)
#define LED2_On()                   GPIOB->PCOR    = (1<<7)
#define LED2_Off()                  GPIOB->PSOR    = (1<<7)

#endif /* NVxx_CONFIG_H_ */
```

## 1.3 初始化过程中的管脚复用

**以 PA5 脚为例，讲解一下在系统初始化过程中的管脚复用问题**

**首先通过看 NV32 的管脚分配图**

| 管脚编号 | | | 优先级 低 ---> 高 | | | | |
|---|---|---|---|---|---|---|---|
| LQFP64 | LQFP48 | LQFP32 | 管脚名称 | 功能 1 | 功能 2 | 功能 3 | 功能 4 |
| 63 | 47 | 31 | PA5 | IRQ | ETM0_CLK | - | $\overline{\text{RESET}}$ |

我们可以直观的看出，此管脚上默认优先级最高的就是复位功能，类于这种系统级的功能用来管脚复用的情况还有很多种，比如 NMI，SWD 功能所在引脚的管脚复用，都需要在系统初始化函数 Sysinit 中进行配置。

1.在 Sysinit.c 中的 sysinit 函数中初始化 SIM 模块的结构体：SIM_ConfigType sSIMConfig = {{0},0};

2.利用模块化编程的思想，若宏定义 DISABLE_RST 这个参数，则禁用 RESET 脚，即给对应的结构体变量赋值，对应的引脚参数参看 SIM 章节的 SIM_SOPT 系统选项寄存器的详细信息。

```
#if defined(DISABLE_RST)
        sSIMConfig.sBits.bDisableRESET = 1;//禁用 RESET 脚
#endif
```

再比如，要禁用 NMI 引脚功能，作为普通 IO 口，和禁用 RESET 管脚同样的方法，进行 DISABLE_NMI 宏定义即可，即在文件开头#define DISABLE_NMI

```
#if defined(DISABLE_NMI)
    sSIMConfig.sBits.bDisableNMI = 1;//禁用不可屏蔽中断的管脚 PB4
#endif
```

3.进行其他相关的配置以后，通过结构体传参进行 SIM 模块的初始化：SIM_Init(&sSIMConfig);具体的 SIM 模块的功能和函数见 NV32F100x 参考手册和 SIM 模块的相关说明

**特别提醒：**

*在上电复位默认 RESET 以及 NMI 功能开启，并且只允许写入一次有效。

*在禁用 SWD 调试方式时，要考虑再次下载调试。在开发板上烧录时，在烧写之前拔掉上电跳帽，按住复位按键，重新插上跳帽，在此过程中，复位按键一直按住，点击烧录按钮，然后松开复位对 MCU 进行烧录。

## 1.4 关于 NV32F100x 系列时钟配置简介

常用总线时钟配置的方式分为两种：FEE（使用外部晶振）和 FEI（使用内部 IRC-31.25KHz）

### 1.4.1 使用 FEE 外部晶振作为系统时钟

1. 在 NV32Config.h 文件中定义 #define USE_FEE（此时#define USE_FEI 应当被注释掉）
2. 定义外部晶振的频率，单位为 KHZ，如开发板上的晶振为 10M，即为 10000KHZ，则同样在 NV32Config.h 定义为 #define EXT_CLK_FREQ_KHZ 10000
3. 计算出所要得到的总线时钟，即先分频，而后倍频（1280 倍），外部时钟分频在 ICS.c 中查找到 void ICS_SetClkDivider(uint32_t u32ClkFreqKHz)；这个函数，根据对应所选的晶振，来配置相应的分频关系

```
case 10000L:
            /* 开发上为 10MHz 的晶振 */
    ICS->C1 = (ICS->C1 & ~(ICS_C1_RDIV_MASK)) | ICS_C1_RDIV(3);
    break;
```

参考如下表格进行分频

| 5-3 RDIV | 参考时钟分频系数，参考时钟的分频结果必须是 31.25k～39.0625k | | |
|---|---|---|---|
| | | 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 |

在 syinit 函数中我们定义了，超过 4MHZ,则将 OSC_CR[RANGE]置位，目的就是为了能将分频后的频率限制在红色字体以内

```
#if   (EXT_CLK_FREQ_KHZ >=4000)
    sICSConfig.oscConfig.bRange = 1;              /* 高范围模式 */
#endif
```

所以超过 4MHZ 的话，选择右边的分频位，例程中的为外接 10MHZ 晶振

*首先分频位选择 3，即 512 分频,ICS->C1 = (ICS->C1 & ~(ICS_C1_RDIV_MASK)) | ICS_C1_RDIV(3)

再倍频 1280,BDIV_ENCODE 设置为 1 即配置 BDIV 为 2

得到内核/系统时钟为:BUS_CLOCK=10000000/512*1280/2=25MHZ(NV32F100x 的内核/系统时钟不得超过 40M)

得到总线频率为 25MHZ，在 NV32Config.h 中定义

```
 #elif   (EXT_CLK_FREQ_KHZ == 10000)
       #define   BUS_CLK_HZ     25000000L
```

### 1.4.2 使用 FEI 内部时钟作为系统时钟

1 在 NV32Config.h 文件中定义 #define USE_FEI（此时#define USE_FEE 应当被注释掉）

2. 在芯片量产的时候，内部振荡器 IRC 校准至 31.25K。1280 倍频后达到 40MHZ(此处作修改，NV32F100x 的系统时钟不得超过 40M)

例程包中 BDIV_ENCODE 设置为 0 即配置 BDIV 为分频系数 1，系统时钟为 40MHz
例程包中 BDIV_ENCODE 设置为 1 即配置 BDIV 为分频系数 2，系统时钟为 20MHz
选择内部时钟源分频，在 ICS.c 中找到函数 void ICS_Init(ICS_ConfigType *pConfig)；
参见 ICS_C2 寄存器的 BDIV 位

| | |
|---|---|
| 7-5<br>BDIV | 内部时钟源分频参数<br>000 对选中的时钟源做 1 分频 （40MHZ 前提下，为 40MHZ）<br>001 对选中的时钟源做 2 分频 （40MHZ 前提下，为 20MHZ），以下同理<br>010 对选中的时钟源做 4 分频<br>011 对选中的时钟源做 8 分频<br>100 对选中的时钟源做 16 分频<br>101 对选中的时钟源做 32 分频<br>110 对选中的时钟源做 64 分频<br>111 对选中的时钟源做 128 分频 |

## 1.5 关于看门狗的启用与禁用

Start.c 文件中的 void SystemInit( void )；函数定义了看门狗的使能和禁用，当宏定义#define ENABLE_WDOG 时看门狗关闭，但允许其更新；否则看门狗关闭，并后续配置无效。

2018.4.11