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：在部分 PDK 包中，此处需修正
#if defined(TRIM_IRC)
    ICS_Trim(ICS_TRIM_VALUE);//TRIM 值校准，校准内部 IRC 时钟
#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
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 RESET 脚
#endif
#if defined(DISABLE_SWD)
        sSIMConfig.sBits.bDisableSWD = 1;//禁用 SWD 调试，注：慎用，考虑后面的调试方式
#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，这也就是为什么有时候没有初始化 LED，但是板载的绿灯还是会亮，在 NV32Config.h 文件中注释掉 #define OUTPUT_BUSCLK 即可 */
#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，具体查看 NV32 管脚分配图
#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

```
#define CPU_NV32     /*NV32 的宏定义*/
#define TEST

#define TRIM_IRC                        /*!<定义 TRIM 值校准内部 IRC*/
#define SPI0_PINREMAP                   /*!<映射 SPI0 可选管脚 */
#define PRINT_SYS_LOG                   /*!<定义是否打印系统信息 */

/* 定义输出系统时钟 */
// 定义是否输出总线时钟，输出管脚为 PH2，通过宏定义的方式，决定是否开启
// #define OUTPUT_BUSCLK
#define ICS_TRIM_VALUE    0x29          /*!< trim IRC to 37.5KHz and FLL output=48MHz */
/*!
   定义所选择的时钟模式，若选择外部时钟还需选择外接频率，以及波特率等设置，若没有选择，则
   默认选择 FEI 内部时钟模式
*/
// #define USE_FEI                      //选择内部时钟
// #define USE_FEE_OSC
   #define USE_FEE                     //选择外部时钟
// #define USE_FBELP
// #define USE_FBE_OSC
   /*! 定义外部时钟晶振的频率. */
// #define EXT_CLK_FREQ_KHZ      32          /* in KHz */
// #define EXT_CLK_FREQ_KHZ      4000        /* in KHz */
// #define EXT_CLK_FREQ_KHZ      4000        /* in KHz */
// #define EXT_CLK_FREQ_KHZ      1000        /* in KHz */
   #define EXT_CLK_FREQ_KHZ      12000       /* in KHz */   //板载的晶振为 12M，这里选择 12M
 /*! define UART port # to be used */
   #define TERM_PORT     UART1             //NV32 的开发板默认 USB 串口为 UART1

 /* 定义总线时钟 */
 #if defined(USE_FEI)    //选择内部时钟，默认为 48MHZ，若要进行其他选择，见内部时钟管理说明
       #define   BUS_CLK_HZ        48000000L
 #elif   (EXT_CLK_FREQ_KHZ == 20000)
         #define   BUS_CLK_HZ          50000000L
 #elif   (EXT_CLK_FREQ_KHZ == 12000)
// 12M 的晶振进行 512 分频，在进行 1280 的倍频，得到当前 30M 系统时钟，详细说明见内部时钟模块
       #define   BUS_CLK_HZ          30000000L
  #elif   (EXT_CLK_FREQ_KHZ == 8000)
       #define   BUS_CLK_HZ        24000000L
  #elif   (EXT_CLK_FREQ_KHZ == 4000)
         #define   BUS_CLK_HZ          40000000L
```

```
#elif   (EXT_CLK_FREQ_KHZ == 32)
        #define   BUS_CLK_HZ          16777216L
#else
        #define   BUS_CLK_HZ          60000000L
#endif

/*! 定义 UART 的波特率 */
#define UART_PRINT_BITRATE                  115200
```

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	ETMO_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，具体查看 NV32 管脚分配图
#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：

*在禁用 RESET 时，要考虑复位方式，禁用 RESET 管脚时 MCU 可以通过上电复位解决。

*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，按住复位开关，重新插上跳帽，在此过程中，按键一直按住，点击烧录按钮，此时松开跳帽对 MCU 进行复位。

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）

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 12M，即为 12000KHZ，则同样在 NV32Config.h 定义为 #define EXT_CLK_FREQ_KHZ 12000
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 12000L:
            /* 开发上为 12MHz 的晶振 */
    ICS->C1 = (ICS->C1 & ~(ICS_C1_RDIV_MASK)) | ICS_C1_RDIV(4);
    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 32k～39k		
		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;            /* high range */
#endif
```

所以超过 4MHZ 的话，选择右边的分频位，例程中的为外接 12MHZ 晶振

*首先分频位选择 4，即 512 分频,ICS->C1 = (ICS->C1 & ~(ICS_C1_RDIV_MASK)) | ICS_C1_RDIV(4)

再倍频 1280 得到总线时钟:BUS_CLOCK=12000000/512*1280=30MHZ

得到总线频率为 30MHZ，在 NV32Config.h 中定义

```
 #elif  (EXT_CLK_FREQ_KHZ == 12000)
        #define   BUS_CLK_HZ    30000000L
```

*当选择分频位为 3 时，即 256 分频 ，ICS->C1 = (ICS->C1 & ~(ICS_C1_RDIV_MASK)) | ICS_C1_RDIV(3)

再倍频 1280 得到总线时钟：BUS_CLOCK=12000000/256*1280=60MHZ

同样，修改 BUS_CLK_HZ 值为 6000000L

```
#elif  (EXT_CLK_FREQ_KHZ == 12000)
    #define   BUS_CLK_HZ    60000000L
```

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 37.5K。1280 倍频后达到 48MHZ，最高需求可达 60MHZ（不建议）。

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 BDIV	内部时钟源分频参数 000 对选中的时钟源做 1 分频 （48MHZ 前提下，为 48MHZ） 001 对选中的时钟源做 2 分频 （48MHZ 前提下，为 24MHZ），以下同理 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

```
#if defined(CPU_NV32)
  if(((ICS->C2 & ICS_C2_BDIV_MASK)>>5) == 1)//判断有无进行分频
    {
        ICS->C2 = (ICS->C2 & ~(ICS_C2_BDIV_MASK)) | ICS_C2_BDIV(0);
    }
        #else
        ICS->C2 = (ICS->C2 & ~(ICS_C2_BDIV_MASK)) | ICS_C2_BDIV(0);
         #endif
```

若选择分频，则对应上表，修改红色标记处的分频值

例：选择使用 FEI 内部时钟达到系统时钟为 24MHZ，则在红色标记处修改 BDIV 为 1

即 ICS->C2 = (ICS->C2 & ~(ICS_C2_BDIV_MASK)) | ICS_C2_BDIV(1);

3.在 NV32_Config.h 文件中定义

```
#if defined(USE_FEI)
#define   BUS_CLK_HZ          24000000L
```

例程中为 48MHZ，可按照上文相应的配置进行修改

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 时看门狗开启，否则为禁用。

```
void SystemInit( void )
{
#if !defined(ENABLE_WDOG)
  /* Disable the watchdog ETMer */
    WDOG_Disable();
#else
  /* Disable the watchdog ETMer but enable update */
    WDOG_DisableWDOGEnableUpdate();
#endif

}
```