NV32F100x PMC 模块编程示例

第一章 库函数简介

1.1 库函数列表

void PMC_Init(PMC_Type *pPMC, PMC_ConfigType *pPMC_Config)

通过结构体 PMC_ConfigType 初始化 PMC

void PMC_DeInit(PMC_Type *pPMC)

PMC 模块恢复默认状态函数

void PMC_SetMode(PMC_Type *pPMC,uint8_t u8PmcMode)

MCU 工作模式选择函数

PMC_GetLVWFlag(PMC);

获取低压报警标志位

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 PMC_EnableLVDInStopMode(PMC_Type *pPMC)

低压检测在停止模式期间使能

void PMC_DisableLVDInStopMode(PMC_Type *pPMC)

低压检测在停止模式期间禁用

void PMC_EnableLVDRst(PMC_Type *pPMC)

发生使能低压检测事件时强制进行 MCU 复位

void PMC_DisableLVDRst(PMC_Type *pPMC)

LVD 事件不产生硬件复位

void PMC_EnableLVD(PMC_Type *pPMC)

LVD 逻辑使能

void PMC_DisableLVD(PMC_Type *pPMC)

LVD 逻辑禁用

void PMC_SetLVDTripVolt(PMC_Type *pPMC, uint8_t Trippoint)

选择低压检测电压

void PMC_SetLVWTripVolt(PMC_Type *pPMC, uint8_t Trippoint)

选择低压报警电压

void PMC_EnableLVWInterrupt(PMC_Type *pPMC)

LVWF=1 时请求硬件中断

void PMC_DisableLVWInterrupt(PMC_Type *pPMC)

禁用硬件中断(使用轮询)

uint8_t PMC_GetLVWFlag(PMC_Type *pPMC)

读取低压警报标志

void PMC_ClrLVWFlag(PMC_Type *pPMC）

清除低压报警标志位

1.2 PMC 模块的特性

*包含一个防御低电压影响的系统，以便在供电电压变动时保护存储器中的内容并控制 MCU 系统的状态

*包含一个上电复位(POR)电路

*具有可供用户选择跳闸电压（高电压(VLVDH)或低电压(VLVDL)）的 LVD 电路

1.3 PMC 使用说明

*通过对 PMC_SPMSC2[LVDV]和 PMC_SPMSC2[LVWV]配置，设置低压检测和报警电压

*对 PMC_SPMSC1 寄存器配置，设置是否使用低压检测功能

*当 ADC 模块选择带隙通道时，使能带隙缓冲区

关于低压检测与报警的具体配置，请参阅参考手册

第二章 PMC 模块初始化

2.1 系统电源管理状态和控制寄存器 1（PMC_SPMSC1）

字段	描述
7 LVWF	低压警报标志 指示低压警报状态。 注: 如果 Vsupply 转换低于跳变点或在复位位后 Vsupply 已低于 VLVW，那么 LVWF 将置位。上电复位后，LVWF 可能为 1；因此，为了使用 LVW 中断功能，在使能 LVWIE 前，LVWF 必须首先通过写 LVWACK 清零。 0 不存在低压警报。 1 存在或曾经存在低压警报。
6 LVWACK	低压警报应答 若 LVWF=1，则已发生低压条件。为了应答该低压警报，向 LVWACK 写入 1，如果低压警报不再发生，那么就会自动清零 LVWF。
5 LVWIE	低压警报中断使能 使能 LVWF 的硬件中断请求。 0 禁用硬件中断(使用轮询)。 1 LVWF=1 时请求硬件中断。
4 LVDRE	低压检测复位使能 该一次性写入位可使能 LVD 事件以产生硬件复位（假设 LVDE = 1）

	注: 复位后，该字段仅可可写入一次，其他写操作会被忽略。 若 LVDRE = 0，则使用用 LVW 监控状态，因为没有标志变为有效。 0 LVD 事件不产生硬件复位 1 发生使能低压检测事件时强制进行 MCU 复位
3 LVDSE	低压检测停止使能 假设 LVDE = 1,则该读/写字段可确定低压检测功能在 MCU 处于停止模式时是否工作。 0 低压检测在停止模式期间禁用。 1 低压检测在停止模式期间使能。
2 LVDE	低压检测使能 该一次性写入位使能低压检测逻辑并认证该寄存器中其他字段的操作。 注: 复位后，该字段仅可可写入一次，其他写操作会被忽略。 0 LVD 逻辑禁用。 1 LVD 逻辑使能。
1 保留	该字段为保留字段。 此只读字段为保留字段且值始终为 0。
0 BGBE	带隙缓冲区使能 使能某个内部缓冲区以便带隙基准电压源可供 ADC 模块在其某个内部通道或选作 ACMP 基准的带隙上使用。 0 带隙缓冲区禁用。 1 带隙缓冲区使能。

2.2 系统电源管理状态和控制寄存器 2（PMC_SPMSC2）

字段	描述
7 保留	该字段为保留字段。 此只读字段为保留字段且值始终为 0。
6 LVDV	低压检测电压选择 该一次性写入选择低压检测(LVD)跳变点设置。 0 选择低电平跳变点(VLVD=VLVDL)。 1 选择高电平跳变点(VLVD=VLVDH)。
5-4 LVWV	低压警报电压选择 选择低压警报(LVW)跳变点电压。更多详情请参见数据手册。 00 选择低电平跳变点(VLVW=VLVW1)。 01 选择中间电平 1 跳变点(VLVW=VLVW2)。 10 选择中间电平 2 跳变点(VLVW=VLVW3)。 11 选择高电平跳变点(VLVW=VLVW4)。
3-0 保留	该字段为保留字段。 读取始终为 0。

函数名	PMC_Init
函数原形	PMC_Init(PMC_Type *pPMC, PMC_ConfigType *pPMC_Config)
功能描述	配置结构体 pConfig 来初始化 PMC
输入参数	PMC 的配置结构体 pPMC_Config，PMC 模块 pPMC
输出参数	无

返回值	无
先决条件	无
函数使用实例	先设置 PMC 配置结构体和模块类型， PMC_Init(PMC, &PMC_Config);

```
/*****************************************************************************//*!
* @初始化 PMC 模块
*
* @输入     pPMC_Config         指向 PMC 配置结构体.
*
* @输入     pPMC                 指向 PMC 模块.
*
* @无返回
*****************************************************************************/
void PMC_Init(PMC_Type *pPMC, PMC_ConfigType *pPMC_Config)
{
    pPMC->SPMSC1 = pPMC_Config->sCtrlstatus.byte;          //配置 SPMSC1 寄存器
    pPMC->SPMSC2 = pPMC_Config->sDetectVoltSelect.byte;   //配置 SPMSC2 寄存器
}
```

第三章 MCU 工作模式选择

函数名	PMC_SetMode
函数原形	PMC_SetMode(PMC_Type *pPMC,uint8_t u8PmcMode)
功能描述	设置 MCU 工作模式
输入参数	MCU 模式选择参数
输出参数	无
返回值	无
先决条件	无
函数使用实例	先设置 MCU 工作模式参数， PMC_SetMode(PMC,PmcModeStop4)

```
/*****************************************************************************//*!
*
* @ 设置 MCU 运行模式
*
* @   输入   u8PmcMode          运行模式的选择
*
* @   输入   pPMC               指向 PMC 模块
*
*   无返回
*
*****************************************************************************/
void PMC_SetMode(PMC_Type *pPMC,uint8_t u8PmcMode)
```

```
    switch(u8PmcMode & 0x3)
     {
          case PmcModeRun:       //运行模式
               break;
          case PmcModeWait:      //等待模式
               wait();
               break;
          case PmcModeStop4:    //停止模式，低压检测使能
               pPMC->SPMSC1 |= (PMC_SPMSC1_LVDE_MASK | PMC_SPMSC1_LVDSE_MASK);
               stop();
               break;
          case PmcModeStop3: //停止模式低压检测禁用
               pPMC->SPMSC1  &= ~(PMC_SPMSC1_LVDE_MASK | PMC_SPMSC1_LVDRE_MASK |
PMC_SPMSC1_LVDSE_MASK);
               stop();
               break;
          default:
               break;
     }
}
```

注：PMC 为用户提供多种功耗的选择，该器件支持运行、待机、和停止模式，具体请参看第一章电源管理模块。

第四章 样例程序

PMC_demo

```
/*****************************************************************************************
*
*PMC 样例程序
*
*例程通过 RTC 中断，将 MCU 从停止模式下唤醒,程序运行 MCU 进入停止模式，经过 4s MCU 被唤醒
*
*注：在 MCU 执行 STOP 命令之前使能 RTC 模块，有关将 MCU 从等待和停止模式唤醒的方式，请参看参
*考手册第一章电源管理模块。
*
*****************************************************************************************/
#include "common.h"
#include "rtc.h"
#include "pmc.h"
#include "uart.h"
#include "sysinit.h"
```

```
#include "sim.h"

int main (void);
void RTC_Task(void);
/*********************************************************************/
int main (void)
{
    uint8_t u8Ch;
    PMC_ConfigType    PMC_Config={0};
    RTC_ConfigType    RTC_Config={0};

    /*  系统初始化  */
      sysinit();
    //SIM_Init();
    printf("\nRunning the PMC_demo project.\n");

    LED0_Init();
    LED2_Init();
    /*PMC 配置结构体初始化*/
    PMC_Config.sCtrlstatus.bits.bBandgapEn = 1;    //使能带隙缓冲区
    PMC_Config.sCtrlstatus.bits.bLvdStopEn = 0;    //低压检测在停止模式禁用
    PMC_Config.sCtrlstatus.bits.bLvdRstEn = 0;     //LVD 时间不产生硬件复位
    PMC_Init(PMC, &PMC_Config);                    //PMC 模块初始化
    PMC_DisableLVWInterrupt(PMC);                  //禁用外部中断
    u8Ch = PMC_GetLVWFlag(PMC);                    //获取低压报警标志位

  /*    RTC 配置结构体初始化，使其每 4 秒产生一个中断*/

    RTC_Config.u16ModuloValue = 4;         // 装载值到模数寄存器中
    RTC_Config.bClockSource     = RTC_CLKSRC_1KHZ;   / /选取时钟源为 1KHZ
    RTC_Config.bClockPresaler = RTC_CLK_PRESCALER_1000; //分频系数为 1000
    RTC_SetCallback(RTC_Task);         //设置回调函数
    RTC_Config.bInterruptEn     = 1;     //使能 RTC 中断
    RTC_Init(&RTC_Config);             //RTC 初始化

    printf("\nEnter stop mode and will be woken up in about 4s by RTC Irq.\n");
//      SIM->SCGC    = 0x00000001;
    PMC_SetMode(PMC,PmcModeStop4); //mcu 进入停止模式
    //经过 4sMCU 被唤醒
    printf("\nWake up now.\n");

}

/*********************************************************************************//*!
```

```
*
* RTC 任务子函数，闪烁 led 灯
*
*******************************************************************************/
void RTC_Task(void)
{
    /* toggle LED1 */
    LED0_Toggle();
}
```