NV32F100x ACMP 比较器模块编程示例

第一章 所有库函数简介

库函数列表

void ACMP_Init(ACMP_Type *pACMPx, ACMP_ConfigType *pConfig);
通过结构体 ACMP_ConfigType 来初始化 ACMP
void ACMP_DeInit(ACMP_Type *pACMPx);
恢复初始化值
void ACMP_ConfigDAC(ACMP_Type *pACMPx, ACMP_DACType *pDACConfig);
配置 ACMP 内部的 DAC
void ACMP_SetCallback(ACMP_Type *pACMPx, ACMP_CallbackPtr pfnCallback);
设置回调函数

寄存器操作的内联函数，调用内联函数和直接操作寄存器效率一样高
void ACMP_InputPinDisable(ACMP_Type *pACMPx, uint8_t u8InputPin)
比较器输入使能
void ACMP_DacOutputSet(ACMP_Type *pACMPx, uint8_t u8DacValue)
比较器内部 DAC 输出电压配置
Void ACMP_DacReferenceSelect(ACMP_Type *pACMPx, uint8_t u8RefSelect)
内部 DAC 参考电压配置
void ACMP_DacDisable(ACMP_Type *pACMPx)
禁止内部 DAC
void ACMP_DacEnable(ACMP_Type *pACMPx)
使能内部 DAC
void ACMP_NegativeInputSelect(ACMP_Type *pACMPx, uint8_t u8NegPinSel)
比较器负极输入配置
void ACMP_PositiveInputSelect(ACMP_Type *pACMPx, uint8_t u8PosPinSel)
比较器正极输入配置
void ACMP_ClrFlag(ACMP_Type *pACMPx)
清比较器中断标识位
uint8_t ACMP_GetFlag(ACMP_Type *pACMPx)
获取比较器中断标识位
void ACMP_DisableInterrupt(ACMP_Type *pACMPx)
禁止比较器中断
void ACMP_EnableInterrupt(ACMP_Type *pACMPx)
使能比较器中断
void ACMP_SelectHyst(ACMP_Type *pACMPx, uint8_t u8HystSelect)
设置比较器迟滞
void ACMP_DisablePinOut(ACMP_Type *pACMPx)
禁止比较器结果输出
void ACMP_EnablePinOut(ACMP_Type *pACMPx)
使能比较器结果输出

void ACMP_SelectIntMode(ACMP_Type *pACMPx, uint8_t u8EdgeSelect)
设置触发比较器中断模式
void ACMP_Disable(ACMP_Type *pACMPx)
禁止比较器
void ACMP_Enable(ACMP_Type *pACMPx)
使能比较器

ACMP 特性说明

*每个 ACMP 最多 4 个输入可以选择，3 个外部输入和内部 DAC 产生的电压。
*可以通过 IO 管脚输出，可选择翻转比较器输出。
*可在比较器输出的下降沿，上升沿，上升或者下降沿触发。
*可在停止模式下操作。

ACMP 使用说明

*ACMP 为 mcu 内部比较器，首先需要对其输入选择，中断模式进行初始化。
*设置好之后 ACMP 就开始工作了，条件满足就会引发中断。
*运行过程中最好不要做输入选择的配置。不使用 DAC 时，关闭 DAC 以降低功耗。

1.1 ACMP 初始化

函数名	ACMP_Init
函数原形	ACMP_Init(ACMP_Type *pACMPx, ACMP_ConfigType *pConfig)
功能描述	以配置结构体 pConfig 来初始化 ACMP
输入参数	配置结构体 ACMP_ConfigType，模块结构体 ACMP_Type
输出参数	无
返回值	无
先决条件	无
函数使用实例	先设置配置结构体，ACMP_Init(ACMP0, &ACMP_Config);

```
/*****************************************************************************//*!
*
* @初始化 ACMP
*
* @param     pACMPx          ACMP 配置结构体
* @param     pConfig          control parameters.
*
* @return none.
*
* @ Pass/ Fail criteria: none.
*
* @see     ACMP_DeInit.
*
*****************************************************************************/
```

```
void ACMP_Init(ACMP_Type *pACMPx, ACMP_ConfigType *pConfig)
{
    if(ACMP0 == pACMPx)
    {
        /* 如果要初始化的为 ACMP0,使能 acmp0 的时钟*/
        SIM->SCGC |= SIM_SCGC_ACMP0_MASK;

        /* 使能 ACMP0 的中断 */
        if(pConfig->sCtrlStatus.bits.bIntEn)
            NVIC_EnableIRQ(ACMP0_IRQn);
    }
    else
    {
        SIM->SCGC |= SIM_SCGC_ACMP1_MASK;
        if(pConfig->sCtrlStatus.bits.bIntEn)
            NVIC_EnableIRQ(ACMP1_IRQn);
    }
    /* neg and pos pin are not equal */
    pACMPx->C0 = pConfig->sPinSelect.byte; // ACMP 输入正负极配置
    ACMP_ConfigDAC(pACMPx, &pConfig->sDacSet ); //ACMP 内部 DAC 配置
    //pACMPx->C1 = pConfig->sDacSet.byte;
    pACMPx->C2 = pConfig->sPinEnable.byte;//外部管脚配置
    pACMPx->CS = pConfig->sCtrlStatus.byte;//
}
```

1.2 配置 ACMP 中的 DAC

函数名	ACMP_ConfigDAC
函数原形	ACMP_ConfigDAC(ACMP_Type *pACMPx, ACMP_DACType *pDACConfig)
功能描述	配置 ACMP 内部 DAC
输入参数	ACMP 模块结构体，以及 DAC 的配置结构体
输出参数	无
返回值	无
先决条件	无
函数使用实例	先设置 DAC 配置结构体，ACMP_ConfigDAC(ACMP0, &pDAC_Config)；

```
/*****************************************************************************//*!
*
* @brief   对内部 DAC 进行初始化
* @param    pACMPx          pointer to an ACMP register base.
* @param    pDACConfig      pointer to an ACMP DAC control structure.
*
* @return none.
```

```
*
* @ Pass/ Fail criteria: none.
*
*****************************************************************************/
void ACMP_ConfigDAC(ACMP_Type *pACMPx, ACMP_DACType *pDACConfig)
{
    pACMPx->C1 = pDACConfig->byte;
}
```

第二章 样例程序

2.1 ACMP_demo

说明：比较内部的 DAC 生成的一个电压和外部管脚的电压，外部管脚 PA0 电压低于 DAC 电压产生中断，并且通过 PA4 输出比较结果

```
/*****************************************************************************
*
* @内部比较器的 demo 程序，
*
*
*****************************************************************************/

#include "common.h"
#include "acmp.h"
#include "pmc.h"
#include "sysinit.h"

int main (void);
void FunForCallback(void);


/*****************************************************************************/
int main (void)
{
    ACMP_ConfigType ACMP_Config;
    PMC_ConfigType   PMC_Config={0};

    /* Perform processor initialization */
```

```
    sysinit();
    PMC_Config.sCtrlstatus.bits.bBandgapEn = 1; //PMC 模块使能 bandgap
    PMC_Config.sCtrlstatus.bits.bLvdStopEn = 0; //禁止 PMC  模块关闭低电压检测
    PMC_Config.sCtrlstatus.bits.bLvdRstEn = 0; //禁止 PMC 低电压复位
    PMC_Init(PMC, &PMC_Config);
    PMC_DisableLVWInterrupt(PMC);
  //u8Ch = PMC_GetLVWFlag(PMC);
    printf("\nRunning the ACMP_demo project.\n");


    /* 初始化 ACMP 配置结构体 */
    ACMP_Config.sCtrlStatus.bits.bIntEn = TRUE; /* 使能中断，默认情况下比较器下降沿触发中断 */
    ACMP_Config.sCtrlStatus.bits.bOutEn = 1;      /* 通过引脚 ACMP0_OUT(PA4) */
    ACMP_Config.sPinSelect.bits.bNegPin = 0x3;   /* 比较器负极输入为 DAC */
    ACMP_Config.sPinSelect.bits.bPosPin = 0;      /* 正极输入为外部端口 ACMP0_IN0(PA0) */
    ACMP_Config.sDacSet.bits.bEn          = TRUE; /* 使能内部 DAC */
    ACMP_Config.sDacSet.bits.bRef         = DAC_REF_VDDA;      /* reference:Vdda */
    ACMP_Config.sDacSet.bits.bVal         = 0x1F; /* DAC 的输出电压为一半 */
    ACMP_Config.sPinEnable.bits.bEn       = TRUE; /* 使能外部引脚 */

    ACMP_SetCallback(ACMP0,FunForCallback);       /* 注册回调函数 */

    ACMP_Init(ACMP0, &ACMP_Config);                 /* 初始化 ACMP */

    ACMP_Enable(ACMP0);
    /*这个时候来使能 ACMP0，ACMP0 就开始 work 了，后面尽量不要再对 ACMP0 的输入做修改*/

 // PMC_SetMode(PMC,PmcModeStop3);
    //while(1)
    //{
 // PMC_SetMode(PMC,PmcModeStop3);
    //}
}

/*****************************************************************************//*!
*
* @ 回调函数
*
* @param none.
*
* @return none.
*
* @ Pass/ Fail criteria: none.
*
```

```
*****************************************************************************/
void FunForCallback(void)
{
    if(ACMP_GetFlag(ACMP0))
    {
        ACMP_ClrFlag(ACMP0);
    }

    printf("\nCallback happens.");
}
```