如何使用 STM32 的 PVD 对电源的电压进行监控

用户在使用 STM32 时，可以利用其内部的 PVD 对 VDD 的电压进行监控，通过电源控制寄存器(PWR_CR)中的 PLS[2:0]位来设定监控的电压值。

PLS[2:0]位用于选择 PVD 监控电源的电压阀值：

000：2.2V
001：2.3V
010：2.4V
011：2.5V
100：2.6V
101：2.7V
110：2.8V
111：2.9V

在电源控制/状态寄存器(PWR_CSR)中的 PVDO 标志用来表明 VDD 是高于还是低于 PVD 设定的电压阀值。该事件连接到外部中断的第 16 线，如果该中断在外部中断寄存器中被使能的，该事件就会产生中断。当 VDD 下降到 PVD 阀值以下和（或）当 VDD 上升到 PVD 阀值之上时，根据外部中断第 16 线的上升/下降边沿触发设置，就会产生 PVD 中断。这一特性可用于发现电压出现异常时，执行紧急关闭任务。

下面是用于测试 PVD 的代码：

主程序的代码：

```
/* Includes ------------------------------------------------------------------*/
#include "stm32f10x_lib.h"

/* Private typedef -----------------------------------------------------------*/
typedef enum {FAILED = 0, PASSED = !FAILED} TestStatus;

/* Private define ------------------------------------------------------------*/

/* Private macro -------------------------------------------------------------*/

/* Private variables ---------------------------------------------------------*/
ErrorStatus HSEStartUpStatus;

/* Private function prototypes -----------------------------------------------*/
void RCC_Configuration(void);
void GPIO_Configuration(void);
void EXTI_Configuration(void);
void NVIC_Configuration(void);

/* Private functions ---------------------------------------------------------*/
```

```
/*******************************************************************************
* Function Name  : main
* Description    : Main program
* Input          : None
* Output         : None
* Return         : None
*******************************************************************************/
int main(void)
{

        RCC_Configuration();            /* System Clocks Configuration */
        GPIO_Configuration();           /* Configure the GPIO ports */
        NVIC_Configuration();           /* NVIC configuration */
        EXTI_Configuration();

        PWR_PVDLevelConfig(PWR_PVDLevel_2V8);
        PWR_PVDCmd(ENABLE);

        while(1) {}
}

/*******************************************************************
* Function Name  : RCC_Configuration
* Description    : Configures the different system clocks.
* Input          : None
* Output         : None
* Return         : None
*******************************************************************/
void RCC_Configuration(void)
{
        RCC_DeInit();  // RCC system reset(for debug purpose)
        RCC_HSEConfig(RCC_HSE_ON);  // Enable HSE
        HSEStartUpStatus = RCC_WaitForHSEStartUp();  // Wait till HSE is ready
        if(HSEStartUpStatus == SUCCESS) {
                RCC_HCLKConfig(RCC_SYSCLK_Div1);          // HCLK = SYSCLK
                RCC_PCLK2Config(RCC_HCLK_Div1); // PCLK2 = HCLK
                RCC_PCLK1Config(RCC_HCLK_Div1); // PCLK1 = HCLK/2
                FLASH_SetLatency(FLASH_Latency_2); // Flash 2 wait state
                FLASH_PrefetchBufferCmd(FLASH_PrefetchBuffer_Enable);       // Enable
Prefetch Buffer
                RCC_PLLConfig(RCC_PLLSource_HSE_Div1, RCC_PLLMul_9);// PLLCLK
= 8MHz * 9 = 72 MHz
                RCC_PLLCmd(ENABLE);                             // Enable PLL
                while(RCC_GetFlagStatus(RCC_FLAG_PLLRDY) == RESET) {}    // Wait till
PLL is ready
                RCC_SYSCLKConfig(RCC_SYSCLKSource_PLLCLK);    // Select PLL as
system clock source
                while(RCC_GetSYSCLKSource() != 0x08) {}    // Wait till PLL is used as
system clock source
        }

        RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);
        RCC_APB1PeriphClockCmd(RCC_APB1Periph_PWR, ENABLE);
}
```

```
/*******************************************************************
* Function Name  : GPIO_Configuration
* Description    : Configures the different GPIO ports.
* Input          : None
* Output         : None
* Return         : None
*******************************************************************/
void GPIO_Configuration(void)
{
        GPIO_InitTypeDef GPIO_InitStructure;

        GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5|GPIO_Pin_6|GPIO_Pin_7|GPIO_Pin_8;
        GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
        GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT_PP;
        GPIO_Init(GPIOB, &GPIO_InitStructure);          //GPIO B
}

/*******************************************************************
* Function Name  : EXTI_Configuration
* Description    : Configures .
* Input          : None
* Output         : None
* Return         : None
*******************************************************************/
void EXTI_Configuration(void)
{
        EXTI_InitTypeDef EXTI_InitStructure;

        EXTI_DeInit();
        EXTI_StructInit(&EXTI_InitStructure);
        EXTI_InitStructure.EXTI_Line = EXTI_Line16;
        EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;
        EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Rising_Falling;
        EXTI_InitStructure.EXTI_LineCmd = ENABLE;
        EXTI_Init(&EXTI_InitStructure);     // Configure EXTI Line16 to generate an interrupt
}
/*******************************************************************************
* Function Name  : NVIC_Configuration
* Description    : Configures Vector Table base location.
* Input          : None
* Output         : None
* Return         : None
*******************************************************************************/
void NVIC_Configuration(void)
{
        NVIC_InitTypeDef NVIC_InitStructure;

#ifdef VECT_TAB_RAM
        /* Set the Vector Table base location at 0x20000000 */
        NVIC_SetVectorTable(NVIC_VectTab_RAM, 0x0);
#else     /* VECT_TAB_FLASH  */
        /* Set the Vector Table base location at 0x08000000 */
        NVIC_SetVectorTable(NVIC_VectTab_FLASH, 0x0);
#endif
```

```
  NVIC_PriorityGroupConfig(NVIC_PriorityGroup_1);  // Configure one bit for preemption
priority

  NVIC_InitStructure.NVIC_IRQChannel = PVD_IRQChannel;
  NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 2;
  NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
  NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
  NVIC_Init(&NVIC_InitStructure);          // Enable the PVD Interrupt

}
```

中断程序：

```
/*******************************************************************************
* Function Name  : PVD_IRQHandler
* Description    : This function handles PVD interrupt request.
* Input          : None
* Output         : None
* Return         : None
*******************************************************************************/
void PVD_IRQHandler(void)
{
        if (PWR_GetFlagStatus(PWR_FLAG_PVDO))
                GPIO_WriteBit(GPIOB, 1 << 5, Bit_SET);
        else
                GPIO_WriteBit(GPIOB, 1 << 5, Bit_RESET);
}
```

注：在 void EXTI_Configuration(void)中，对于 EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Rising_Falling; 中的初始化值，根据你的需要进行修改，具体细节如下：

EXTI_Trigger_Rising --- 表示电压从高电压下降到低于设定的电压阀值产生中断；

EXTI_Trigger_Falling --- 表示电压从低电压上升到高于设定的电压阀值产生中断；

EXTI_Trigger_Rising_Falling --- 表示电压从高电压下降到低于设定的电压阀值、或从低电压上升到高于设定的电压阀值产生中断。