独立看门狗实验iwdg

看门狗的原理：单片机系统在外界的干扰下会出现程序跑飞的现象导致出现死循环，看门狗电路就是为了避免这种情况的发生。看门狗的作用就是在一定时间内（通过定时计数器实现）没有接收喂狗信号（表示 MCU 已经挂了），便实现处理器的自动复位重启（发送复位信号）。

STM32F429 内部自带了 2 个看门狗：独立看门狗（IWDG）和窗口看门狗（WWDG）。这一 章 我们只介绍独立看门狗。在本章中，我们将通过按键PB2来喂狗，然后通过指示灯PF14来显示复位情况。通过用户手册，我们可以看到按键按下则PB2输入高电平，否则输入低电平，使用时要设置为输入状态。

STM32F4 的独立看门狗由**内部专门的 32Khz 低速时钟（LSI）**驱动，即使主时钟发生故障， 它也仍然有效。这里需要注意独立看门狗的时钟是一个内部RC时钟，所以 并不是准确的32Khz， 而是在 15~47Khz 之间的一个可变化的时钟，只是我们在估算的时候，以 32Khz 的频率来计算。所以在看门狗复位之前，我们要提前喂狗操作。

看门狗使用方法：

在键值寄存器(IWDG_KR)中写入 0xCCCC，开始启用独立看门狗；此时计数器开始从其复位值 0xFFF 递减计数。当计数器计数到末尾 0x000 时，会产生一个复位信号(IWDG_RESET)。无论何时，只要键寄存器 IWDG_KR 中被写入 0xAAAA， IWDG_RLR 中的值就会被重新加载到计数器中从而避免产生看门狗复位 。

用HAL库配置独立看门狗：

1.取消寄存器写保护，设置看门狗预分频系数和重装载值。

首先我们必须取消 IWDG_PR （预分频值）和 IWDG_RLR （重装载值）寄存器的写保护，这样才可以设置寄存器 IWDG_PR 和 IWDG_RLR 的值。取消写保护和设置预分频系数以及重装载值在 HAL 库中是通 过函数 HAL_IWDG_Init 实现的。该函数声明为：

```
HAL_StatusTypeDef HAL_IWDG_Init(IWDG_HandleTypeDef *hiwdg);
```

> *设置好看门狗的分频系数 prer 和重装载值rlr就可以知道看门狗的喂狗时间 （也就是**看门狗溢出时间**），该时间的计算方式为：$Tout = 4 \times 2^{prer} \times rlr/32(ms)$，prer 为看门狗时钟预分频值（IWDG_PR 值），范围为 0~7；rlr 为看门狗的重装载值（IWDG_RLR 的值）。*

比如我们设定 prer 值为 4（4 代表的是 64 分频，HAL 库中可以使用宏定义标识符 IWDG_PRESCALER_64），rlr 值为 500，那么就可以得到 Tout=64×500/32=1000ms，这样，看 门狗的溢出时间就是 1s，只要你在一秒钟之内，有一次写入 0XAAAA 到 IWDG_KR，就不会 导致看门狗复位（当然写入多次也是可以的）。这里需要提醒大家的是，看门狗的时钟不是准确 的 32Khz，所以在喂狗的时候，最好不要太晚了，否则，有可能发生看门狗复位。

2.重载计数值喂狗（向 IWDG_KR 写入 0XAAAA）

在 HAL 中重载计数值的函数是 HAL_IWDG_Refresh，该函数声明为：`HAL_StatusTypeDef HAL_IWDG_Refresh(IWDG_HandleTypeDef *hiwdg);`

3.启动看门狗(向 IWDG_KR 写入 0XCCCC)

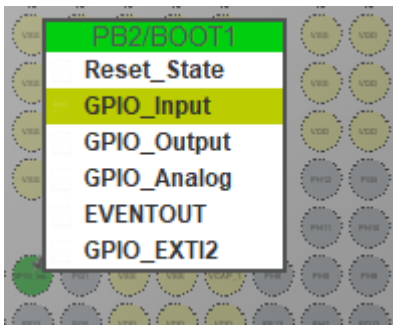
这里可以省略启动看门狗的步骤，因为你可以发现在初始化看门狗函数:`HAL_StatusTypeDef HAL_IWDG_Init(IWDG_HandleTypeDef *hiwdg)`中已经开启门门狗，如图所示。

```c
HAL_StatusTypeDef HAL_IWDG_Init(IWDG_HandleTypeDef *hiwdg)
{
  uint32_t tickstart;

  /* Check the IWDG handle allocation */
  if (hiwdg == NULL)
  {
    return HAL_ERROR;
  }

  /* Check the parameters */
  assert_param(IS_IWDG_ALL_INSTANCE(hiwdg->Instance));
  assert_param(IS_IWDG_PRESCALER(hiwdg->Init.Prescaler));
  assert_param(IS_IWDG_RELOAD(hiwdg->Init.Reload));

  /* Enable IWDG. LSI is turned on automatically */
  __HAL_IWDG_START(hiwdg);

  /* Enable write access to IWDG_PR and IWDG_RLR registers by writing
  0x5555 in KR */
  IWDG_ENABLE_WRITE_ACCESS(hiwdg);

  /* Write to IWDG registers the Prescaler & Reload values to work with */
  hiwdg->Instance->PR = hiwdg->Init.Prescaler;
  hiwdg->Instance->RLR = hiwdg->Init.Reload;

  /* Check pending flag, if previous update not done, return timeout */
  tickstart = HAL_GetTick();

  /* Wait for register to be updated */
  while (hiwdg->Instance->SR != 0x00u)
  {
```

下面开始用Cube生成IWDG模板。

1.配置时钟部分。

2.配置PB2管脚为下拉输入模式，如图。



3.配置PF14为推挽输出模式，默认下拉。

4.开启IWDG，设置pre和rlr如图所示。

则溢出时间为128*2000/32=8s

5.生成keil文件，在主函数中加入如图所示语句。

```
  /* USER CODE BEGIN 2 */
    HAL_GPIO_WritePin(GPIOF,GPIO_PIN_14,GPIO_PIN_SET);
    HAL_Delay(2000);
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
//      HAL_GPIO_WritePin(GPIOF,GPIO_PIN_14,GPIO_PIN_SET);
      if(HAL_GPIO_ReadPin(GPIOB,GPIO_PIN_2)==GPIO_PIN_SET)
      {
      HAL_IWDG_Refresh(&hiwdg);
      }
      HAL_GPIO_WritePin(GPIOF,GPIO_PIN_14,GPIO_PIN_RESET);
//      }


//    if(HAL_GPIO_ReadPin(GPIOB,GPIO_PIN_2)==GPIO_PIN_SET)
//      {
//        HAL_GPIO_WritePin(GPIOF,GPIO_PIN_14,GPIO_PIN_SET);
//      }
//      else
//      {
//        HAL_GPIO_WritePin(GPIOF,GPIO_PIN_14,GPIO_PIN_RESET);
//      }
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
```

则在8s中之内只要按下按键，则喂狗。