STM32Cube高效开发教程（基础篇）

第21章 窗口看门狗

王维波
中国石油大学（华东）控制科学与工程学院

STM32Cube高效开发教程（基础篇）

作者：王维波，鄢志丹，王钊

人民邮电出版社

2021年9月出版

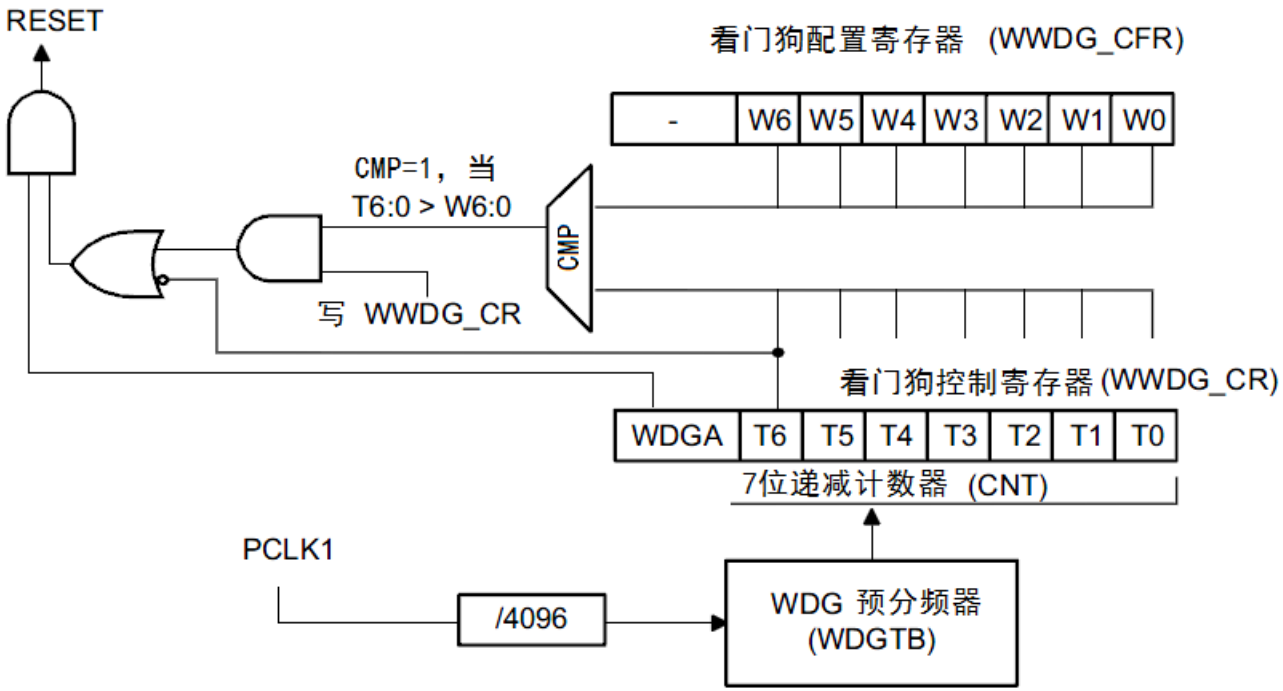
如果有读者需要本书课件的PPT版本用于备课，可以给作者发邮件免费获取，并可加入专门的教学和技术交流QQ群

邮箱：wangwb@upc. edu. cn

21.1窗口看门狗工作原理

21.2窗口看门狗HAL驱动程序

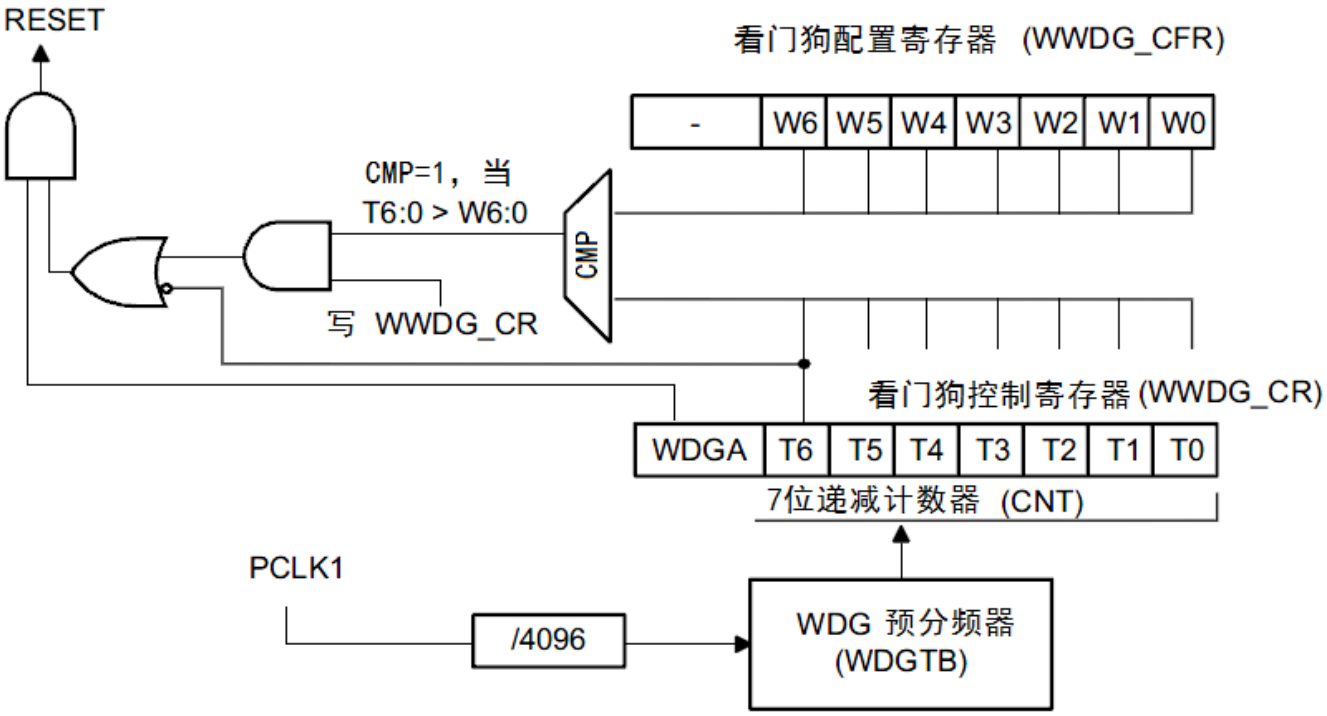
21.3窗口看门狗使用示例



1. 递减计数器

内部有一个7位递减计数器，7位递减计数器的时钟频率是

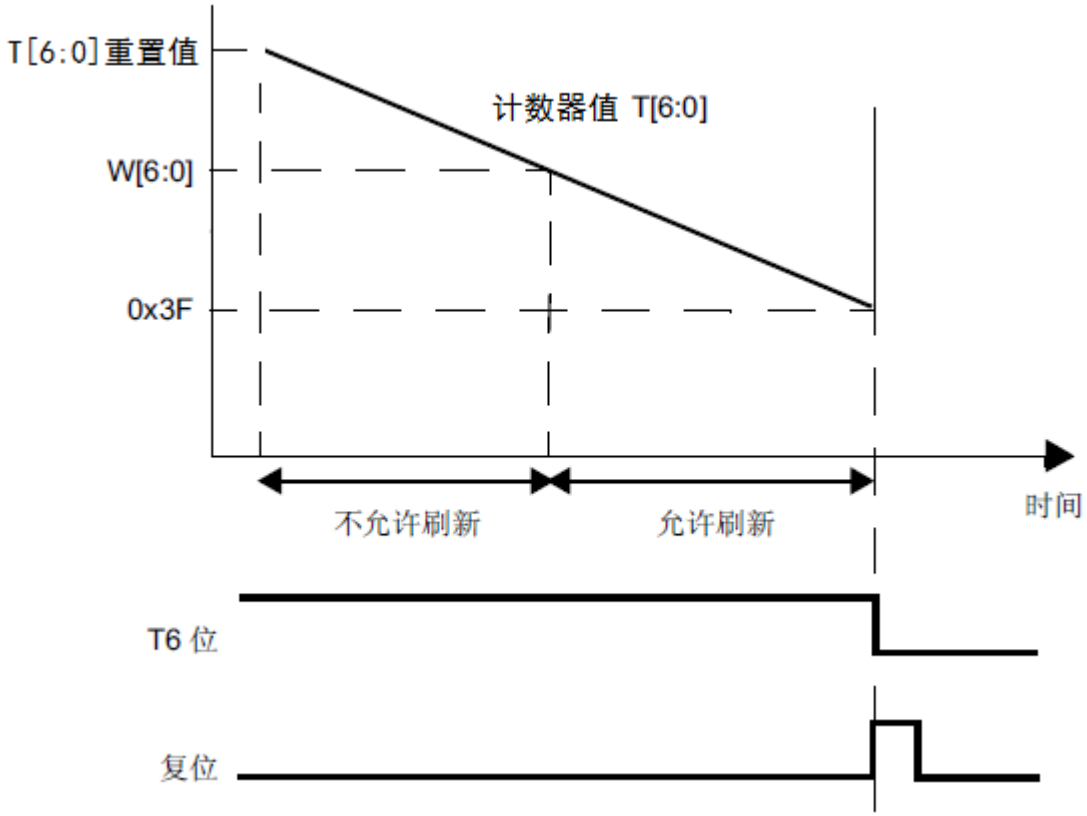
$$f_{CNT} = \frac{f_{PCLK1}}{4096 * DIV}$$



7位递减计数器在T6位由1变为0时就会使系统产生复位，也就是计数值由0x40变为0x3F时产生复位。要避免系统复位，就必须在计数值变为0x3F之前重置计数器，重置计数器的值必须大于0x3F。

2. 窗口值和比较器

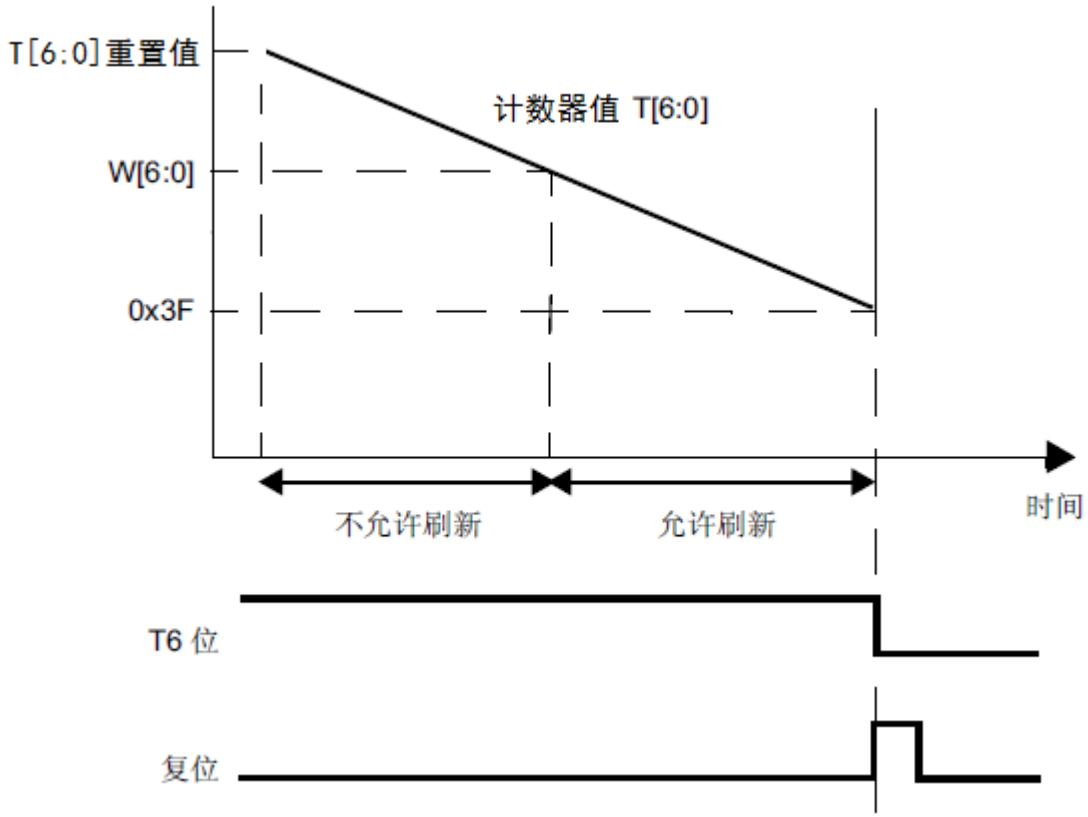
配置7位的窗口值W[6:0]，这个值与计数器的当前值T[6:0]进行比较。



- 当T[6:0]>W[6:0]时，比较器输出为1，这时不允许重置计数器的值，否则系统复位。
- 只有当T[6:0]≤W[6:0]时才可以重置计数器的值，如果在T[6:0]变化到0x3F之前没有重置计数器，就会产生系统复位信号。

3. 看门狗的启动

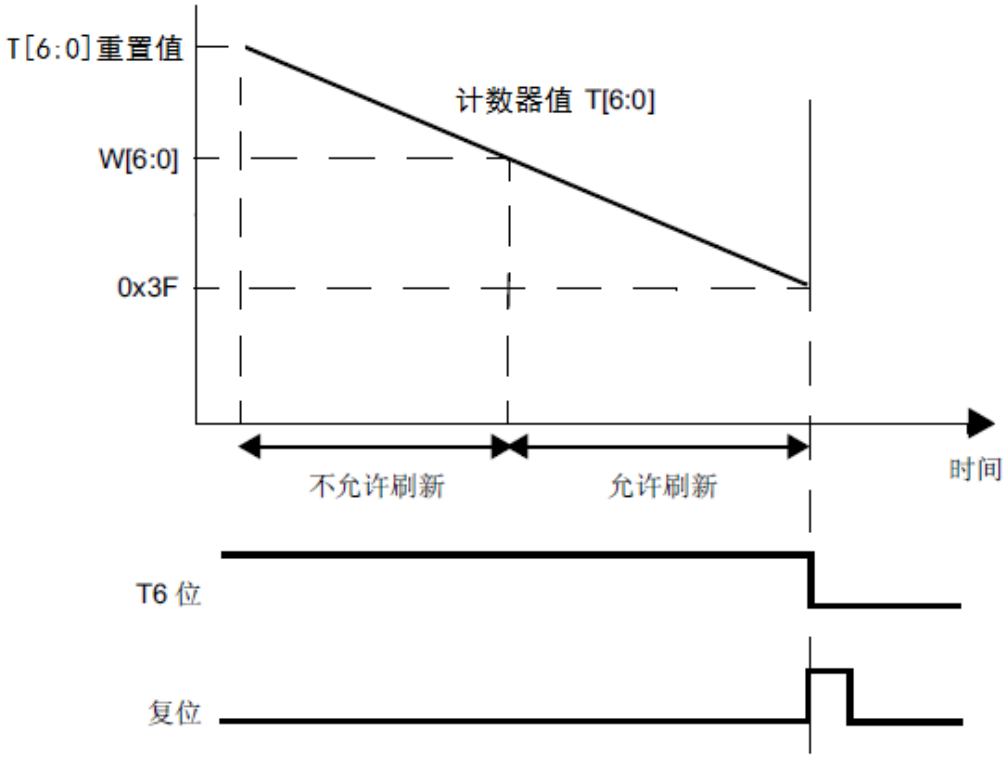
控制寄存器WWDG_CR中的位WDGA用于启动看门狗。启动窗口看门狗后就无法再停止，除非系统复位。



根据窗口看门狗的特点，可以使用软件使系统立刻复位，方法是将WDGA位置1（启动窗口看门狗），并将T6位清零（使看门狗立刻产生复位），也就是设置一个小于0x3F的重置值

4. 提前唤醒中断

有一个提前唤醒中断（Early Wakeup Interrupt，EWI）事件，在递减计数器的值变为0x40时就会触发此中断。



可在此中断服务程序里执行系统复位之前的一些关键操作，但是执行时间有限，只有一个计数器时钟周期。

21.1窗口看门狗工作原理

21.2窗口看门狗HAL驱动程序

21.3窗口看门狗使用示例

1. 窗口看门狗初始化

使用函数HAL_WWDG_Init()进行窗口看门狗初始化，该函数原型定义如下：

```
HAL_StatusTypeDef   HAL_WWDG_Init(WWDG_HandleTypeDef *hwwdg);
```

其中，hwwdg是WWDG_HandleTypeDef结构体类型指针，是窗口看门狗外设对象指针。CubeMX生成的WWDG外设初始化文件wwdg.c中会定义WWDG外设对象变量，定义如下：

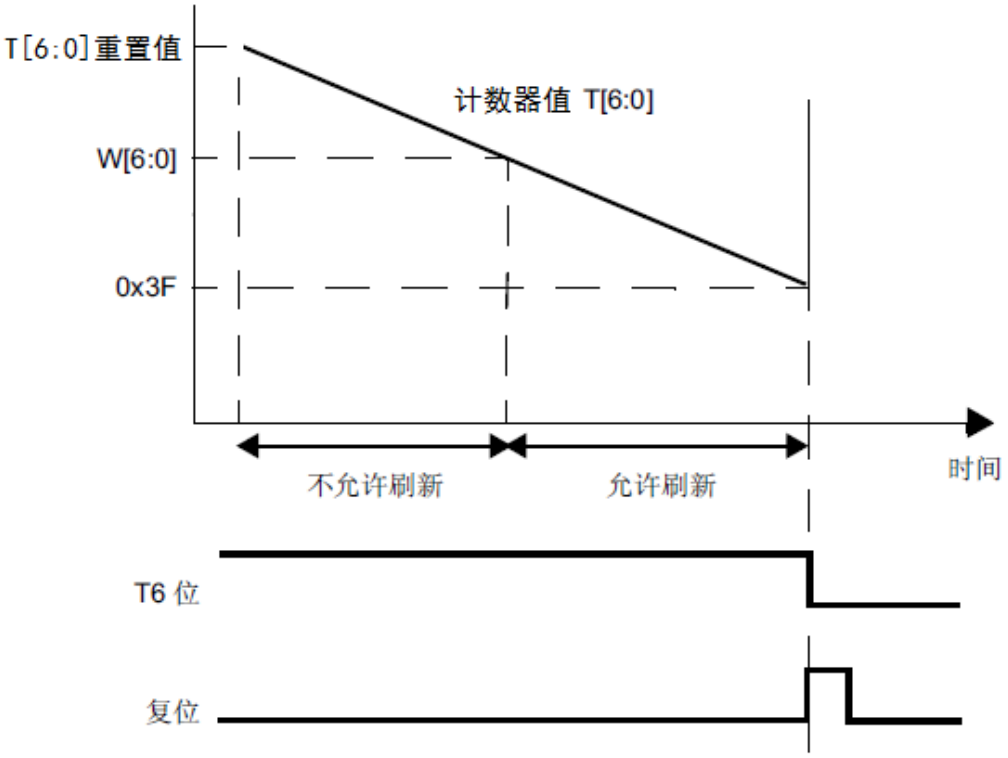
```
WWDG_HandleTypeDef  hwwdg;   //WWDG外设对象变量
```

2. 窗口看门狗刷新

```
HAL_StatusTypeDef   HAL_WWDG_Refresh(WWDG_HandleTypeDef *hwwdg);
```

其功能就是将计数器重置值加载到看门狗的递减计数器，以避免看门狗触发系统复位。

注意，只能在图中的允许刷新时间段才能刷新看门狗



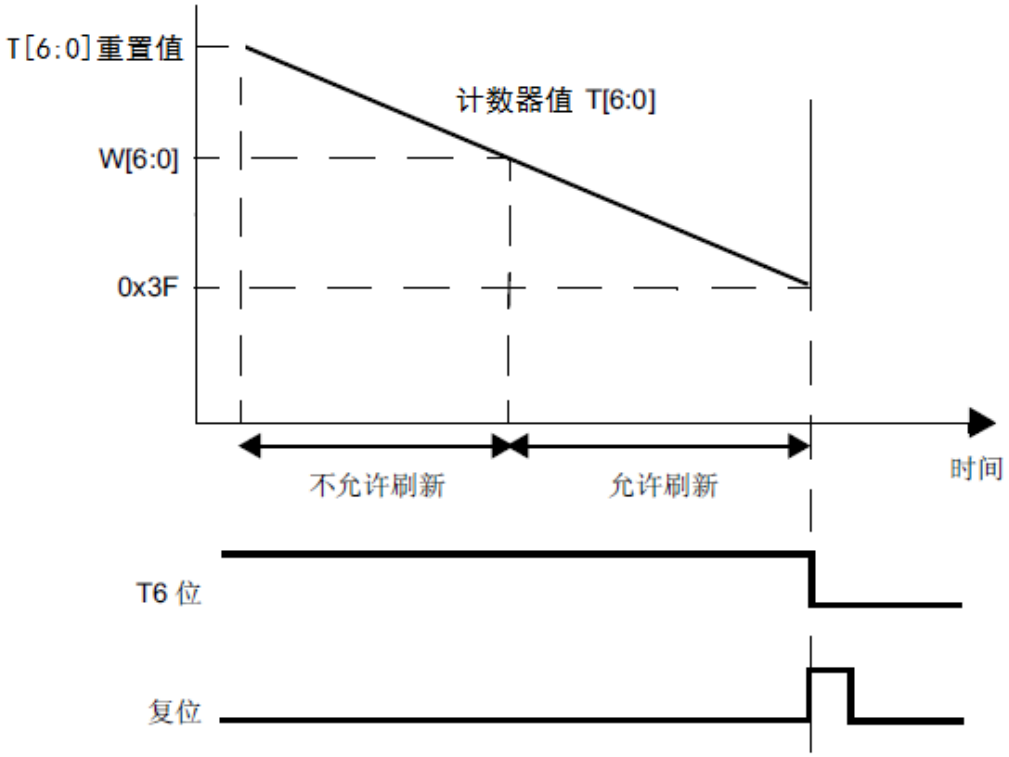
3. EWI中断及其处理

EWI中断回调函数

```
void  HAL_WWDG_EarlyWakeupCallback(WWDG_HandleTypeDef *hwwdg);
```

在递减计数器的值变为0x40时会触发此中断

可在此回调函数里执行系统复位之前的一些关键操作，但是执行时间有限，只有一个计数器时钟周期



21.1窗口看门狗工作原理

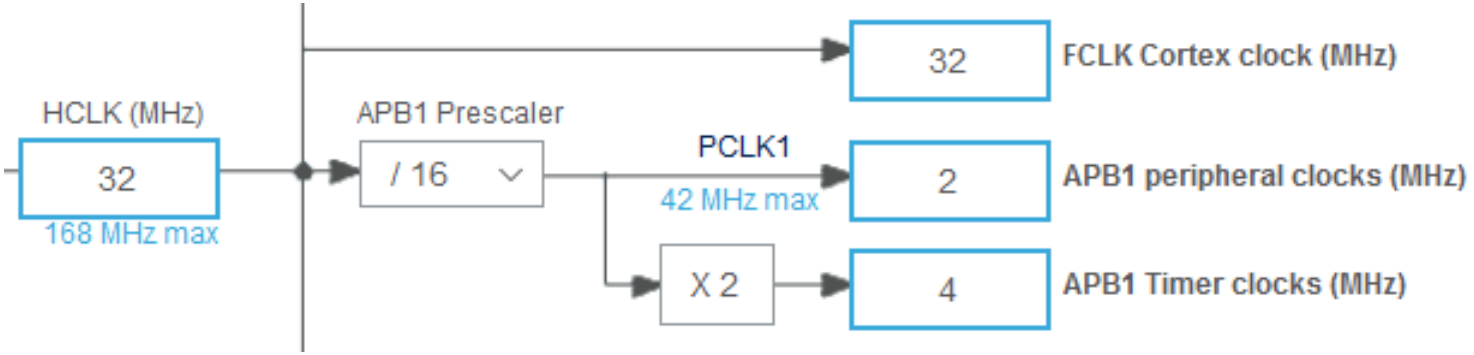
21.2窗口看门狗HAL驱动程序

21.3窗口看门狗使用示例

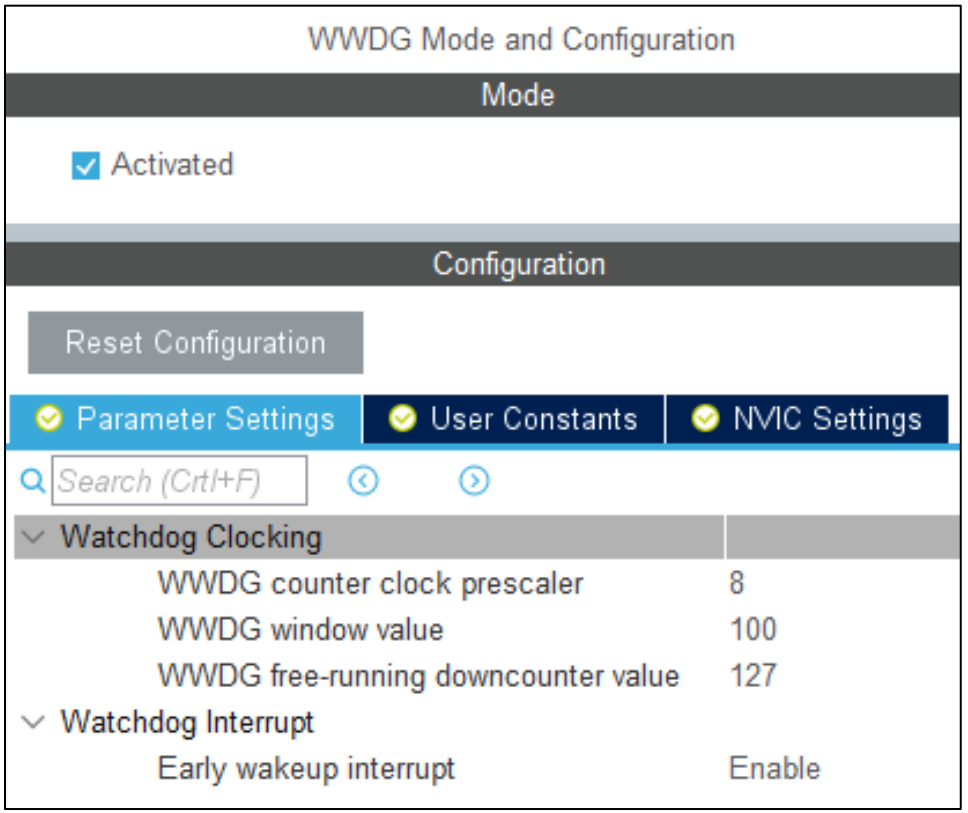
21.3.1 示例功能和CubeMX项目设置

本节示例项目Demo22_1WWDG，只需用到两个LED，不使用LCD，因为程序可能会频繁复位。

配置时钟树，设置HCLK为32 MHz，设置APB1 Prescaler为16，使PCLK1为2 MHz（如图）。因为窗口看门狗要用到PCLK1时钟，使PCLK1为2 MHz是为得到一个较低频率时钟信号用于看门狗的递减计数器，便于观察程序运行效果。



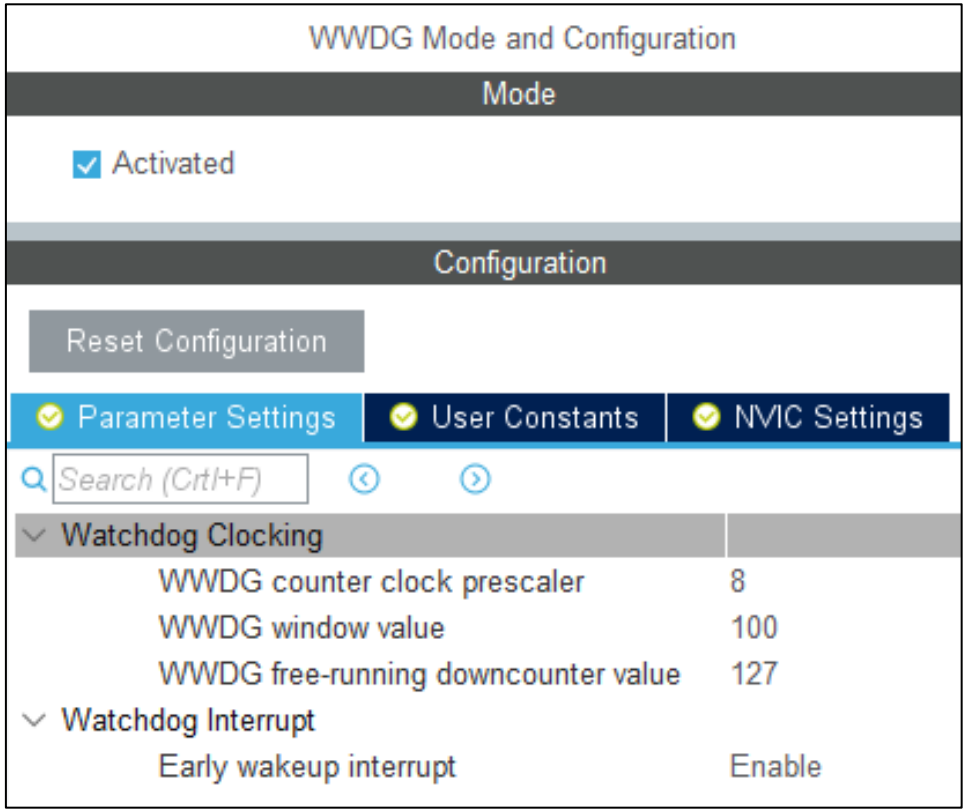
- WWDG counter clock prescaler，看门狗计数器预分频系数，1、2、4、8等
- WWDG window value，窗口值，也就是W[6:0]的值。这个值必须小于计数器的重置值，也必须大于0x3F（十进制值63）



- WWDG free-running downcounter value，递减计数器T[6:0]的重置值，最大值为127（也就是0x7F），必须大于W[6:0]的值

根据图中设置的参数以及PCLK1为2 MHz，可以计算出看门狗的超时时间为

$$timeout = \frac{4096 * DIV * \Delta}{f_{PCLK1}} = \frac{4096 * 8 * (127 - 63)}{2 * 10^6}\text{s} \approx 1049ms$$



计数器重置后不允许刷新的时间段长度是

$$Time_{\text{不允许刷新}} = \frac{4096 * 8 * (127 - 100)}{2 * 10^6}\text{s} \approx 442ms$$

21.3.2 不使用EWI中断

1.主程序

（1）看门狗在允许刷新时间段内及时刷新，可以观察到LED1一直慢速闪烁。

```
LED1_ON();
while (1)
{
    HAL_Delay(800);    //1. 在允许刷新时间段内，看门狗不会触发复位，LED1闪烁
    //      HAL_Delay(1200);//2. 超时，看门狗会触发系统复位,LED1总是亮
    HAL_WWDG_Refresh(&hwwdg);    //刷新看门狗，也就是重置计数器的值
    LED1_Toggle();
/* USER CODE END WHILE */
}
```

（2）看门狗超时自动复位

将程序中的延时改为1200ms，则运行时会看到LED1一直亮着。因为在延时1200ms的过程中，看门狗已经超时导致系统复位，while循环里使LED1输出翻转的代码不会被执行。

```
LED1_ON();
while (1)
{
    //    HAL_Delay(800);    //1. 在允许刷新时间段内，看门狗不会触发复位，LED1闪烁
    HAL_Delay(1200);  //2. 超时，看门狗会触发系统复位,LED1总是亮
    HAL_WWDG_Refresh(&hwwdg);    //刷新看门狗，也就是重置计数器的值
    LED1_Toggle();
/* USER CODE END WHILE */
}
```

（3）在不允许刷新时间段内刷新看门狗

在运行时会看到LED1快速闪烁。在执行LED1_Toggle()前后两个延时合计400ms，还在不允许刷新时间段内（小于442ms），这时调用HAL_WWDG_Refresh()刷新看门狗会使系统复位。

```
/* USER CODE BEGIN WHILE */
LED1_ON();
while (1)
{
    HAL_Delay(200);          //前延时200ms
    LED1_Toggle();
    HAL_Delay(200);          //后延时200ms
    HAL_WWDG_Refresh(&hwwdg);    //3. 在不允许刷新段内刷新看门狗
    HAL_Delay(500);          //这行代码不会被执行
/* USER CODE END WHILE */
}
```

2. WWDG初始化

CubeMX自动生成WWDG初始化函数MX_WWDG_Init()，文件wwdg.c中部分代码如下：

看完整源代码

```
/* 文件： wwdg.c   ----------------------------------------------------------*/
#include "wwdg.h"
WWDG_HandleTypeDef hwwdg;          //WWDG外设对象变量

/* WWDG 初始化函数 */
void MX_WWDG_Init(void)
{
      hwwdg.Instance = WWDG;           //寄存器基址
      hwwdg.Init.Prescaler = WWDG_PRESCALER_8;          //预分频系数
      hwwdg.Init.Window = 100;       //窗口值，W[6:0]的值
      hwwdg.Init.Counter = 127;       //计数器重置值
      hwwdg.Init.EWIMode = WWDG_EWI_ENABLE;          //开启EWI中断
      if (HAL_WWDG_Init(&hwwdg) != HAL_OK)
          Error_Handler();
}
```

21.3.3 使用EWI中断

```
/* USER CODE BEGIN WHILE */
LED1_ON();
LED2_OFF();
while (1)
{
    HAL_Delay(1200);    //超时
    LED1_Toggle();      //不会被执行
    HAL_WWDG_Refresh(&hwwdg);   //刷新看门狗，不会被执行
/* USER CODE END WHILE */
}
```

```
/*  EWI中断回调函数  */
void HAL_WWDG_EarlyWakeupCallback(WWDG_HandleTypeDef *hwwdg)
{
     LED2_ON();       //模拟系统复位前紧急处理
}
```

程序运行时的效果是LED1常亮，LED2闪烁，但是LED2亮的时间很短，灭的时间长。