第 1 章 IWDG—独立看门狗

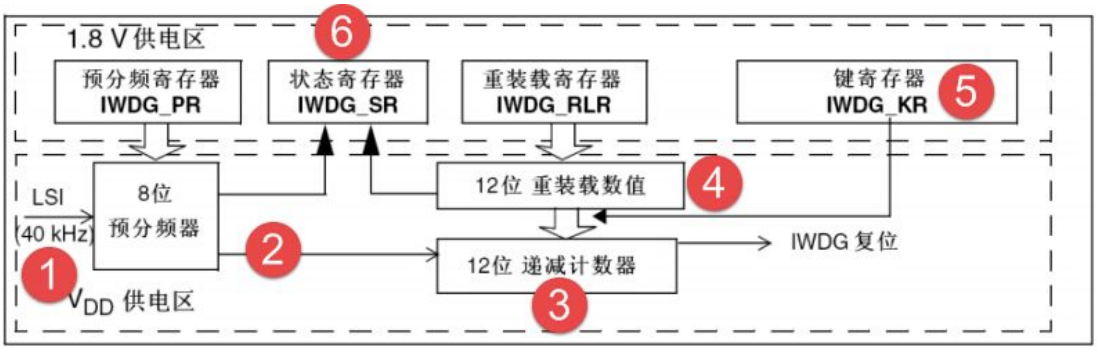
本章参考资料：《STM32 中文参考手册》IWDG 章节。

学习本章时，配合《STM32 中文参考手册》IWDG 章节一起阅读，效果会更佳，特别是涉及到寄存器说明的部分。

1.1 IWDG 简介

STM32 有两个看门狗，一个是独立看门狗另外一个窗口看门狗，独立看门狗号称宠物狗，窗口看门狗号称警犬，本章我们主要分析下独立看门狗的功能框图和它的应用。独立看门狗用通俗一点的话来解释就是一个 12 位的递减计数器，当计数器的值从某个值一直减到 0 的时候，系统就会产生一个复位信号，即 IWDG_RESET。如果在计数没减到 0 之前，刷新了计数器的值的话，那么就不会产生复位信号，这个动作就是我们经常说的喂狗。

1.2 IWDG 功能框图剖析



1. ①独立看门狗时钟

独立看门狗的时钟由独立的 RC 振荡器 LSI 提供，即使主时钟发生故障它仍然有效，非常独立。LSI 的频率一般在 30~60KHZ 之间，根据温度和工作场合会有一定的漂移，我们一般取 40KHZ，所以独立看门狗的定时时间并一定非常精确，只适用于对时间精度要求比较低的场合。

2. ②计数器时钟

递减计数器的时钟由 LSI 经过一个 8 位的预分频器得到，我们可以操作预分频器寄存器 IWDG_PR 来设置分频因子，分频因子可以是：[4,8,16,32,64,128,256,256]，计数器时钟 CK_CNT= 40/ 4*2^PRV，一个计数器时钟计数器就减一。

3. ③计数器

独立看门狗的计数器是一个 12 位的递减计数器，最大值为 0XFFF，当计数器减到 0 时，会产生一个复位信号:IWDG_RESET，让程序重新启动运行，如果在计数器减到 0 之前刷新了计数器的值的话，就不会产生复位信号，重新刷新计数器值的这个动作我们俗称喂狗。

4. ④重装载寄存器

重装载寄存器是一个 12 位的寄存器，里面装着要刷新到计数器的值，这个值的大小决定着独立看门狗的溢出时间。超时时间 Tout = (4*2^prv) / 40 * rlv (s) ，prv 是预分频器寄存器的值，rlv 是重装载寄存器的值。

5. ⑤键寄存器

键寄存器 IWDG_KR 可以说是独立看门狗的一个控制寄存器，主要有三种控制方式，往这个寄存器写入下面三个不同的值有不同的效果。

表格 1 键寄存器取值枚举

0XAAAA	把 RLR 的值重装载到 CNT
0X5555	PR 和 RLR 这两个寄存器可写
0XCCCC	启动 IWDG

通过写往键寄存器写 0XCCC 来启动看门狗是属于软件启动的方式，一旦独立看门狗启动，它就关不掉，只有复位才能关掉。

6. ⑥状态寄存器

状态寄存器 SR 只有位 0：PVU 和位 1：RVU 有效，这两位只能由硬件操作，软件操作不了。RVU：看门狗计数器重装载值更新，硬件置 1 表示重装载值的更新正在进行中，更新完毕之后由硬件清 0。PVU: 看门狗预分频值更新，硬件置’1’指示预分频值的更新正在

进行中，当更新完成后，由硬件清 0。所以只有当 RVU/PVU 等于 0 的时候才可以更新重装载寄存器/预分频寄存器。

1.3 怎么用 IWDG

独立看门狗一般用来检测和解决由程序引起的故障，比如一个程序正常运行的时间是 50ms，在运行完这个段程序之后紧接着进行喂狗，我们设置独立看门狗的定时溢出时间为 60ms，比我们需要监控的程序 50ms 多一点，如果超过 60ms 还没有喂狗，那就说明我们监控的程序出故障了，跑飞了，那么就会产生系统复位，让程序重新运行。

1.4 IWDG 超时实验

1.4.1 硬件设计

1、IWDG 一个

2、按键一个

3、LED 一个

IWDG 属于单片机内部资源，不需要外部电路，需要一个外部的按键和 LED，通过按键来喂狗，喂狗成功 LED 亮，喂狗失败，程序重启，LED 灭一次。

1.4.2 软件设计

我们编写两个 IWDG 驱动文件，bsp_iwdg.h 和 bsp_iwdg.c，用来存放 IWDG 的初始化配置函数。

1. 代码分析

这里只讲解核心的部分代码，有些变量的设置，头文件的包含等并没有涉及到，完整的代码请参考本章配套的工程。

IWDG 配置函数

代码 1 IWDG 配置函数

```
1 void IWDG_Config(uint8_t prv ,uint16_t rlv)
2 {
3     // 使能 预分频寄存器 PR 和重装载寄存器 RLR 可写
```

```
    IWDG_WriteAccessCmd( IWDG_WriteAccess_Enable );

    // 设置预分频器值
    IWDG_SetPrescaler( prv );

    // 设置重装载寄存器值
    IWDG_SetReload( rlv );

    // 把重装载寄存器的值放到计数器中
    IWDG_ReloadCounter();

    // 使能 IWDG
    IWDG_Enable();
}
```

IWDG 配置函数有两个形参，prv 用来设置预分频的值，取值可以是：

代码 2 形参 prv 取值

```
/*
 *     @arg IWDG_Prescaler_4:    IWDG prescaler set to 4
 *     @arg IWDG_Prescaler_8:    IWDG prescaler set to 8
 *     @arg IWDG_Prescaler_16:   IWDG prescaler set to 16
 *     @arg IWDG_Prescaler_32:   IWDG prescaler set to 32
 *     @arg IWDG_Prescaler_64:   IWDG prescaler set to 64
 *     @arg IWDG_Prescaler_128:  IWDG prescaler set to 128
 *     @arg IWDG_Prescaler_256:  IWDG prescaler set to 256
 */
```

这些宏在 stm32f10x_iwdg.h 中定义，宏展开是 8 位的 16 进制数，具体作用是配置配置预分频寄存器 IWDG_PR，获得各种分频系数。形参 rlv 用来设置重装载寄存器 IWDG_RLR 的值，取值范围为 0~0XFFF。溢出时间 Tout = prv/40 * rlv (s)，prv 可以是 [4,8,16,32,64,128,256]。如果我们需要设置 1s 的超时溢出，prv 可以取 IWDG_Prescaler_64，rlv 取 625，即调用:IWDG_Config(IWDG_Prescaler_64 ,625)。Tout=64/40*625=1s。

喂狗函数

代码 3 喂狗函数

```
void IWDG_Feed(void)
{
    // 把重装载寄存器的值放到计数器中，喂狗，防止 IWDG 复位
    // 当计数器的值减到 0 的时候会产生系统复位
    IWDG_ReloadCounter();
}
```

主函数

代码 4 主函数

```
int main(void)
{
    // 配置 LED GPIO，并关闭 LED
    LED_GPIO_Config();
    // 配置按键 GPIO
    Key_GPIO_Config();

    SOFT_Delay(0X008FFFFF);
```

```
    // IWDG 1s 超时溢出
    IWDG_Config(IWDG_Prescaler_64 ,625);
    LED1_ON;

    while (1)
    {
        if ( Key_Scan(macKEY1_GPIO_PORT,macKEY1_GPIO_PIN) == KEY_ON  )
        {
            // 喂狗，如果不喂狗，系统则会复位，LD1 则会灭一次，如果在 1s
            // 时间内准时喂狗的话，则 LED1 会常亮
            IWDG_Feed();
        }
    }
}
```

主函数中我们初始化好 LED 和按键相关的配置，设置 IWDG 1s 超时溢出之后，马上点亮 LED1，然后进入 while 死循环，通过按键来喂狗，如果喂狗成功，前面点亮的 LED 则不会灭，如果喂狗失败的话，系统重启，程序重新执行，在重新启动的时候，LED 会被关断，灭的时间通过 SPFT_Delay（）这个函数调节，软件延时越长，灯灭的时间就越长，喂狗失败的直观感受就越明显。

我们这里是通过按键来模拟一个程序，真正的项目中则不是这样使用。while 部分是我们在项目中具体需要写的代码，这部分的程序可以用独立看门狗来监控，如果我们知道这部分代码的执行时间，比如是 500ms，那么我们可以设置独立看门狗的溢出时间是 00ms，比 500ms 多一点，如果要被监控的程序没有跑飞正常执行的话，那么执行完毕之后就会执行喂狗的程序，如果程序跑飞了那程序就会超时，到达不了喂狗的程序，此时就会产生系统复位，但是也不排除程序跑飞了又跑回来了，刚好喂狗了，歪打正着。所以要想更精确的监控程序，可以使用窗口看门狗，窗口看门狗规定必须在规定的窗口时间内喂狗，早了不行，晚了也不行。

1.4.3 下载验证

把编译好的程序下载到开发板，当 LED 被点亮之后，在 1s 的时间内通过按键来不断的喂狗，如果喂狗失败，LED 会灭一次。如果一直喂狗成功，LED 一直被点亮，如果喂狗一直失败，LED 则以大于 1s 的频率闪烁。