第十三章 软件狗

本章主要介绍 STM32 中的两种看门狗类型，分别为独立看门狗和窗口看门狗。在本章中，我们将完成两个实验，一是验证不及时喂狗，系统将复位重启的实验，二是验证窗口看门狗的功能的实验。本章分为如下几个部分：

13.1 看门狗简介

13.2 硬件设计

13.3 软件设计

13.4 下载验证

13.1 看门狗简介

由于单片机的工作常常会受到来自外界电磁场的干扰，造成程序的跑飞，而陷入死循环，程序的正常运行被打断，由单片机控制的系统无法继续工作，会造成整个系统的陷入停滞状态，发生不可预料的后果，所以出于对单片机运行状态进行实时监测的考虑，便产生了一种专门用于监测单片机程序运行状态的模块或者芯片，俗称“看门狗”(watchdog)，分为独立看门狗和窗口看门狗。

表 13.1.1 对比表

对比点	独立看门狗	窗口看门狗
时钟源	LSI(40KHz 或 32KHz)	PCLK1 或 PCLK3
复位条件	递减计数到 0	计数值大于 W[6:0]值喂狗或减到 0x3F
中断	没有中断	计数值减到 0x40 可产生中断
递减计数器位数	12 位（最大计数范围：4096~0）	7 位（最大计数范围：127~63）
应用场合	防止程序跑飞，死循环，死机	检测程序时效，防止软件异常

- 独立看门狗(Independent watchdog)，IWDG

  本质：能够产生**系统复位信号**的计数器。

  特性：递减的计数器，时钟由独立的 RC 振荡器提供（可在待机和停止模式下运行），看门狗被激活后，当递减计数器计数到 0x000 时产生复位。

  喂狗：在计数器计到 0 之前，重装载计数器的值，防止复位。

  作用：主要用于检测外界电磁干扰，或**硬件**异常导致的程序跑飞问题。

  应用：在一些需要高稳定性的产品中，并且对时间精度要求较低的场合。

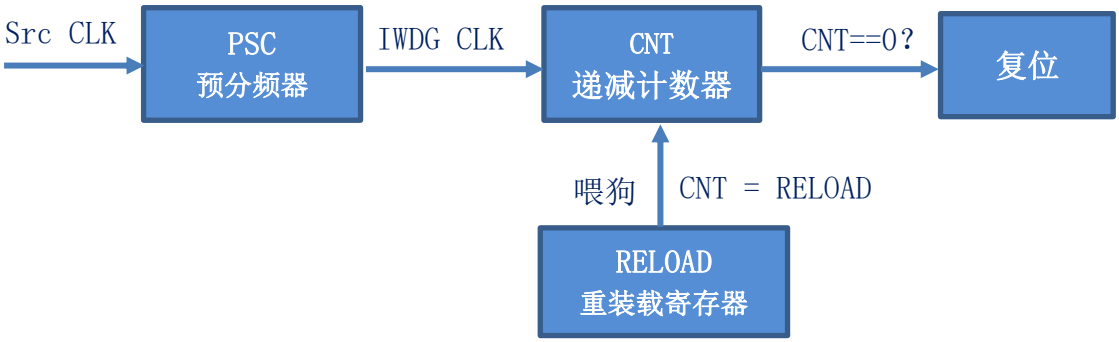

图 13.1.1 IWDG 工作原理

CPU 必须及时喂狗，否则系统复位重启！

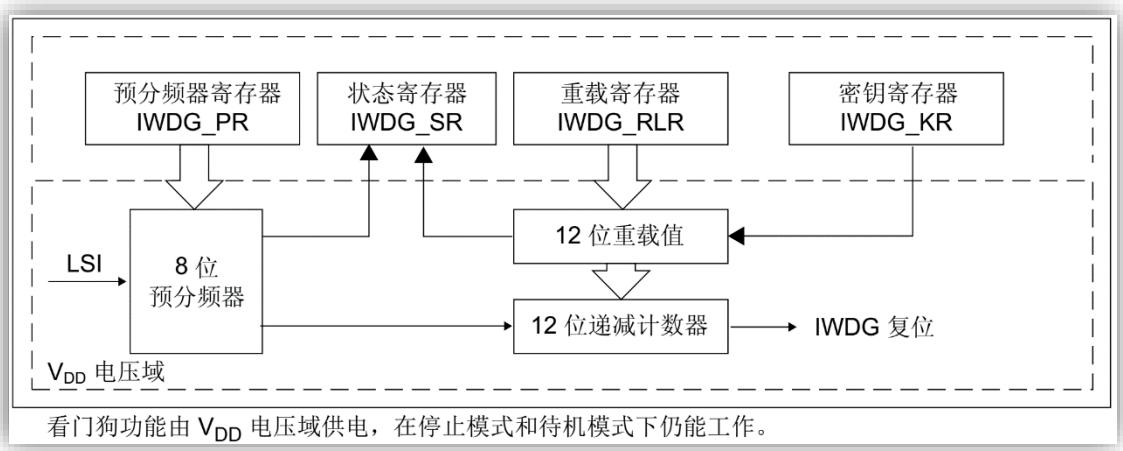

图 13.1.2 IWDG 框图

独立看门狗时钟： 独立看门狗是由专门的低速总线进行驱动，即 **LSI 总线**（LSI 时钟并不精准，F1 用**时钟频率 40KHz**），它可以在主时钟故障的情况下仍然可以工作。启用独立看门狗后，LSI 时钟会自动开启。

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。

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^prer) / 40 * rlr，prer 是预分频器寄存器的值，rlv 是重装载寄存器的值。

键寄存器：键寄存器（IWDG_KR）写 0xCCCC 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。

表 13.1.3 键寄存器

键值	键值作用
0xAAAA	把 RLR 的值重装载到 CNT
0x5555	允许访问 IWDG_PR 和 IWDG_RLR 寄存器
0xCCCC	启动 IWDG

IWDG 溢出时间计算：

$$psc = 4 * 2\text{^}prer$$

$$T_{out} = \frac{(4 * 2\text{^}prer) * rlr}{F_{IWDG}}$$

T_{out}是看门狗溢出时间　　F_{IWDG}是看门狗的时钟源频率

psc是看门狗预分频系数　　prer 是IWDG_PR 的值

rlr 是看门狗重装载值

操作步骤：①取消寄存器写保护（向 IWDG_KR 写入 0x5555）；

②设置独立看门狗的预分频系数和重装载值；

③重载计数值喂狗（向 IDWG_KR 写入 0xAAAA）；

④启动看门狗（向 IWDG_KR 写入 0xCCCC）；

⑤应用程序喂狗。

➢ 窗口看门狗(Window watchingdog)，WWDG

本质：能产生**系统复位信号**和**提前唤醒中断**的计数器

特性：递减的计数器。当递减计数器值从 0x40 减到 0x3F 时复位（即 T6 位跳变到 0），计数器的值大于 W[6:0]值时喂狗会复位，提前唤醒中断：当递减计数器等于 0x40 时可产生。

喂狗：在窗口期内重装载计数器的值，防止复位。

作用：用于监测单片机程序运行时效是否精准，主要检测**软件**异常。

应用：需要精准检测程序运行时间的场合。

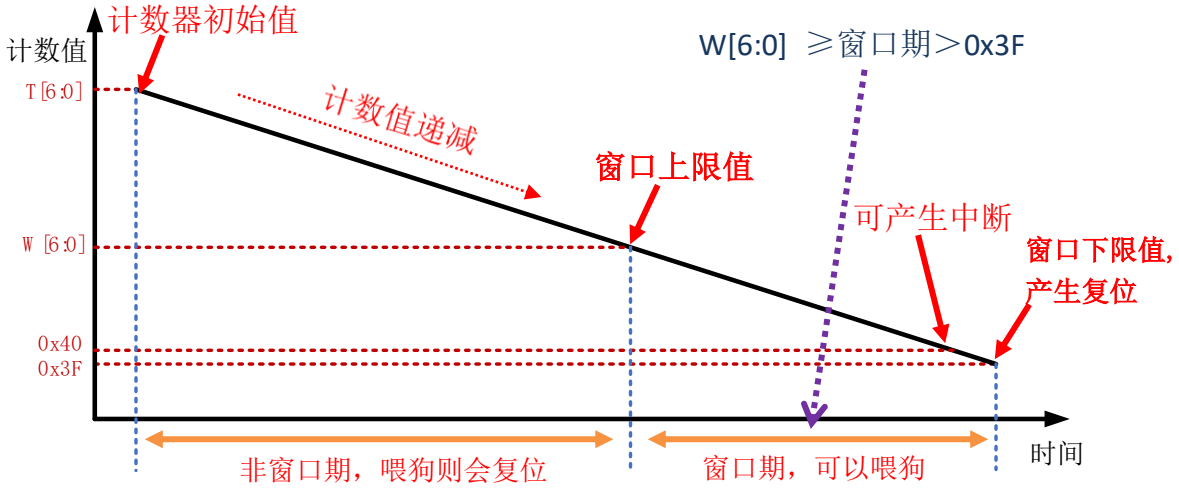

图 13.1.4 WWDG 工作原理

注意：W[6:0]必须大于窗口下限值 0x3F，否则无窗口期

图158　看门狗框图

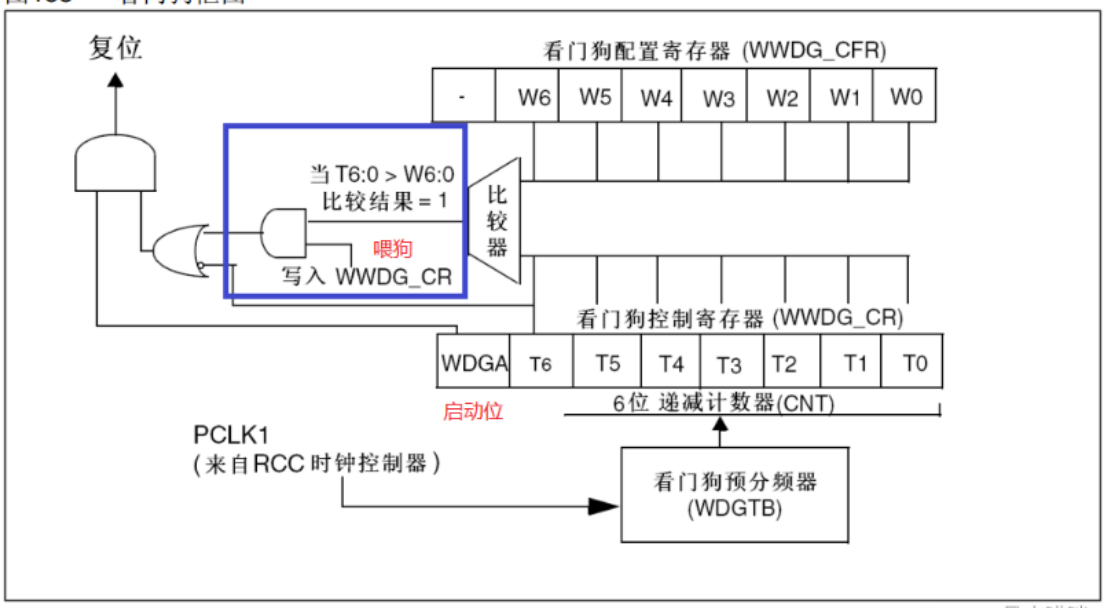

图 13.1.5 看门狗框图

时钟源：F103：PCLK1(36MHz)。

在PCLK1 = 36MHz时的最小-最大超时值

WDGTB	最小超时值	最大超时值
0	113μs	7.28ms
1	227μs	14.56ms
2	455μs	29.12ms
3	910μs	58.25ms

图 13.1.6 WWDG 最短最长超时时间

产生复位信号的逻辑分析：

第一条途径：

- 将 WWDG 模块启动，也就是将 WDGA 位置 1；
- 计数器中的值 T6:0 与上窗口中的值 W6:0 通过比较器进行比较，当 T6:0 大于 W6:0 时产生的结果是 1，这个结构与喂狗后的结果通过与门后的结果仍然是 1；
- 上面产生的 1 通过或门后还是 1，并且与 WDGA 产生的 1 经过与门产生复位信号。

第二条途径：

- 将 WWDG 模块启动，也就是将 WDGA 位置 1；

- 当计数器中的 T6 位变成 0 以后，通过或非门变成 1；
- 产生的 1 与 WDGA 产生的 1 经过与门后产生复位信号。

WWDG 的喂狗可以在中断中实现的，该模块有提前唤醒中断功能。

WWDG 超时时间计算：

$$Tout = \frac{4096 * 2\text{^}WDGTB * (T[5:0] + 1)}{F_{wwdg}}$$

$Tout$是 WWDG 超时时间（没喂狗） F$wwdg$是 WWDG 的时钟源频率

4096 是 WWDG 固定的预分频系数 T[5:0]是 WWDG 计数器低 6 位

2^$WDGTB$是 WWDG_CFR 寄存器设置的预分频系数值

在实验代码中注释部分，详细解释了如何计算非窗口期时间和窗口期时间。

操作步骤：①使能看门狗时钟: RCC_APB1PeriphClockCmd()；

②设置分频系数: WWDG_SetPrescaler()；

③设置上窗口值: WWDG_SetWindowValue()；

④开启提前唤醒中断并分组(可选):

WWDG_SetWindowValue()、NVIC_Init()：

⑤使能看门狗: WWDG_Enable()；

⑥喂狗: WWDG_SetCounter()；

⑦编写中断服务函数: WWDG_IRQHandler(),中断函数中还需要清除中断标志位 WWDG_ClearFlag();。

13.2 硬件设计

本章需要用到的硬件资源有：

1）极风 STM32 开发板

2）STLINK 下载器

下面介绍一下 STM32 开发板和 STLINK 下载器的连接，STLINK 的 3.3V、SWCLK、SWDIO、GND 分别连在 STM32 开发板的 3V3、CLK、DIO、GND 上。总体连接实物图如图 13.2.1 所示：

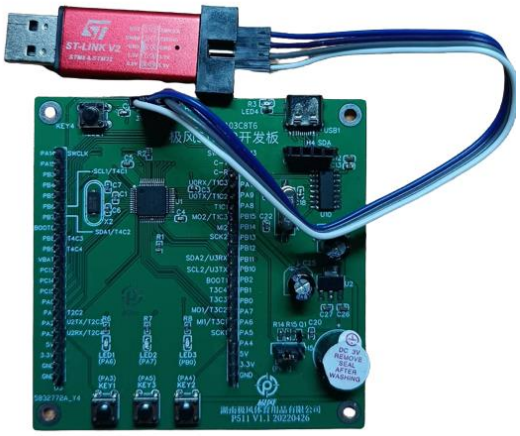

图 13.2.1 总体连接实物图

13.3 软件设计

➢ 验证不及时喂狗，系统将复位重启的实验(IWDG)

（1） 程序实现的功能

验证不及时喂狗，系统将复位重启的实验——按 KEY1 键来喂狗，每隔 1s 至少喂狗一次，此时 LED1 常亮，打开串口，可看见 Yes，喂狗成功；如果没有按 KEY1 键，那么 LED1 闪烁，因为初始状态 LED1 是灭的，打开串口，可看见 No，未喂狗。

（2） 程序的实现

打开**实验 1 IWDG\motor.uvprojx**，我们可以看见工程中拥有的主要文件有 iwdg.c、iwdg.h、led.c、led.h、key.c、key.h、main.c。iwdg.c 文件存放 iwdg 驱动代码，led.c 文件存放 led 驱动代码，key.c 文件存放 key 驱动代码，main.c 文件存放应用代码。这里我们主要介绍 iwdg.c、iwdg.h 和 main.c 文件。

打开 iwdg.c 文件，代码如下：

```
#include "iwdg.h"

/**********************************
* 函数全称:
* void IWDG_Init(uint8_t prer,uint16_t rlr)
*
* 辅助解释：
* prer:分频数
* 分频因子=4*2^prer.最大值只能是 256!
* rlr:重装载寄存器值
```

```
* 溢出时间计算:Tout=((4*2^prer)*rlr)/40 (ms).
*
* 函数作用:
* 初始化独立看门狗
* ***********************************/
void IWDG_Init(uint8_t prer,uint16_t rlr)
{
    IWDG_WriteAccessCmd(IWDG_WriteAccess_Enable);  //取消写保护
    IWDG_SetPrescaler(prer);  //设置 IWDG 预分频值
    IWDG_SetReload(rlr);  //设置 IWDG 重装载值
    IWDG_ReloadCounter();  //按照 IWDG 重装载寄存器的值重装载 IWDG 计数器
    IWDG_Enable();  //使能 IWDG
}
```

下面我们看看头文 iwdg.h 的代码，代码中包含一些头文件定义，数据类型、函数声明，代码如下：

```
#ifndef __IWDG_H
#define __IWDG_H
#include "stdlib.h"
#include "stm32f10x.h"

/* 初始化 */
void IWDG_Init(uint8_t prer,uint16_t rlr);

#endif
```

最后，我们在 main 函数里编写应用代码，main.c 文件如下：

```
#include "stm32f10x.h"
#include <string.h>
#include <stdlib.h>

#include "delay.h"
#include "usart.h"
#include "led.h"
#include "key.h"
#include "iwdg.h"

int main(void)
{
    uint8_t key;
    uart_init(115200);
    LED_Init();              //初始化 LED
    KEY_Init();              //按键初始化
    delay_ms(200);           //延时 200ms 再初始化看门狗,LED1 的变化"可见"
```

```
    IWDG_Init(4,625);    // Tout=((4*2^4)*625)/40 (ms) = 1000 ms 溢出时间为 1s
    GPIO_SetBits(GPIOA, GPIO_Pin_6);

    printf("No!\r\n");    //未喂狗
    while(1)
    {
        key = KEY_Scan(); /*  得到键值  */
        if(key == 1)          /*  按下 KEY1 */
        {
            IWDG_ReloadCounter();//如果按下 KEY1,则喂狗
            printf("Yes!\r\n");//喂狗
        }
    }
}
```

至此，实验 1 IWDG 的软件设计部分就完成了。

- 验证窗口看门狗的功能的实验

（1）程序实现的功能

验证窗口看门狗的实验——方法① 通过 LED1 来指示是否喂狗成功：若 LED1 先灭 500ms 后一直亮，表示喂狗成功；若 LED1 闪烁表示喂狗失败。方法② 在中断服务函数中进行喂狗：若看见 LED2 闪烁，表示喂狗成功；若在中断服务函数中增加延时（由于设置的 WDGTB = 3，最小超时值为 910us），所以延时 1ms 后喂狗会失败，LED2 会不亮。

（2）程序的实现

打开**实验 2 WWDG\motor.uvprojx**，我们可以看见工程中拥有的主要文件有 wwdg.c、wwdg.h、led.c、led.h、key.c、key.h、main.c。iwdg.c 文件存放 iwdg 驱动代码，led.c 文件存放 led 驱动代码，key.c 文件存放 key 驱动代码，main.c 文件存放应用代码。这里我们主要介绍 wwdg.c、wwdg.h 和 main.c 文件。

打开 iwdg.c 文件，代码如下：

```
#include "wwdg.h"
#include "led.h"
#include "delay.h"

/**********************************
*  函数全称:
* void WWDG_Init(void)
*
```

```
* 辅助解释：
* 超时时间计算:Tout=((4096*2^WDGTB)*（T[5:0]+1）)/Fwwdg
* ((4096*8)*（127 - 95))/36000 = 29.13ms 非窗口期，喂狗产生复位，127 是 0x7F
* ((4096*8)*（127 - 63))/36000 = 58.25ms，初始化之后经过 58.25ms 到达窗口期下限值，63 是 0x3F
* 所以可以得到在 20.13ms 到 58.25ms 之间喂狗不产生复位
*
* 函数作用:
* 初始化窗口看门狗
* ***********************************/
void WWDG_Init(void)
{
    NVIC_InitTypeDef NVIC_InitStructure;
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_WWDG,ENABLE);// 使 能  APB1 时钟

    WWDG_SetPrescaler(WWDG_Prescaler_8);//预分频系数设为 8
    WWDG_SetWindowValue(0x5F);//上窗口值为 0x5F，95 是 0x5F
    WWDG_Enable(0x7F);//使能 WWDG

    NVIC_InitStructure.NVIC_IRQChannel = WWDG_IRQn;//WWDG 中断
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 2; //设置抢占优先级
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 3;    // 设置响应优先级
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;    // 使能中断通道
    NVIC_Init(&NVIC_InitStructure);   // 调用 NVIC 初始化函数
    WWDG_ClearFlag();//清除提前唤醒中断标志位

    WWDG_EnableIT();//开启中断
}

/*********************************
* 函数全称:
* void WWDG_IRQHandler(void)
*
* 辅助解释：
* 当计数值为 0x40 时可产生中断
*
* 函数作用:
* 中断服务函数
* ***********************************/
void WWDG_IRQHandler(void)
{
    //delay_ms(1);//由于设置的 WDGTB = 3,最小超时值为 910us，所以延时 1ms 后，喂狗会失败，LED2 会不亮
```

```
//      WWDG_SetCounter(0x7F); //设置计数值
//      WWDG_ClearFlag();//清除提前唤醒中断标志位
//      LED2_Turn(); //LED2 翻转，喂狗成功可以看见 LED2 闪烁
}
```

下面我们看看头文 wwdg.h 的代码，代码中包含一些头文件定义，数据类型、函数声明，代码如下：

```
#ifndef _WWDG_H
#define _WWDG_H
#include "stdlib.h"
#include "stm32f10x.h"

/*  初始化  */
void WWDG_Init(void);
#endif
```

最后，我们在 main 函数里编写应用代码，main.c 文件如下：

```
#include "stm32f10x.h"
#include <string.h>
#include <stdlib.h>

#include "stdio.h"
#include "delay.h"
#include "led.h"
#include "wwdg.h"

int main(void)
{
    LED_Init();                                          //初始化 LED
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);//设置中断优先级分组为组 2
    delay_ms(500);
    WWDG_Init();                                         //初始化窗口看门狗
    GPIO_SetBits(GPIOA, GPIO_Pin_6);                  //LED1 灯亮

    while(1)
    {   //若 LED1 先灭 500ms 后一直亮，表示喂狗成功
        //若 LED1 闪烁表示喂狗失败
        delay_ms(30);//理想下延时在 29.13ms~58.25ms 之间（窗口期）
                     //但是执行设置计数值还需要时间，所以在 30ms~56ms 内能够喂狗成功
        WWDG_SetCounter(0x7F);//设置计数值
    }
}
```

至此，实验 2 WWDG 的软件设计部分就完成了。

13.4 下载验证

在代码编译成功下载程序完成后，在实验 1 IWDG 中按 KEY1 键来喂狗，每隔 1s 至少喂狗一次，此时 LED1 常亮，打开串口，可看见 Yes，喂狗成功；如果没有按 KEY1 键，那么 LED1 闪烁，因为初始状态 LED1 是灭的，打开串口，可看见 No，未喂狗。在实验 2 WWDG 中可以看见 LED1 先灭 500ms 后一直亮，表示喂狗成功；若在中断服务函数中喂狗（需要注释掉 main 函数中的喂狗函数），LED2 闪烁则喂狗成功。