青风带你学 stm32f030 系列教程

---------- 库函数操作版本

出品论坛： www.qfv8.com 青风电子社区

作者： 青风
出品论坛: www.qfv8.com
淘宝店：http://qfv5.taobao.com
QQ 技术群：241364123
硬件平台：QF-STM32F030 开发板

2.5 看门狗 WDG

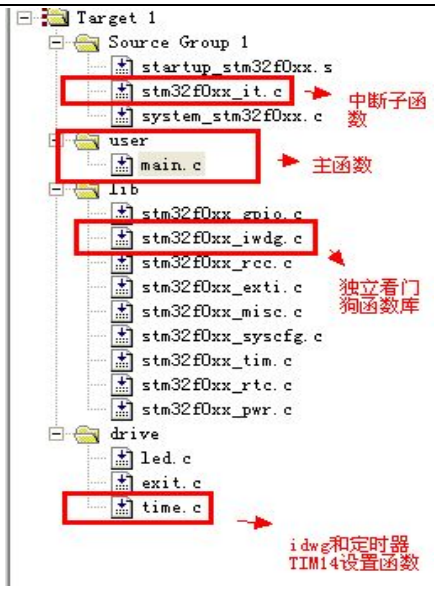
在 stm32f030 系列 CORTEX M0 中内置两个看门狗，提供了更高的安全性、时间的精确性和使用的灵活性。两个看门狗设备(独立看门狗和窗口看门狗)可用来检测和解决由软件错误引起的故障；当计数器达到给定的超时值时，触发一个中断(仅适用于窗口型看门狗)或产生系统复位。

独立看门狗(IWDG) 由专用的低速时钟(LSI)驱动，即使主时钟发生故障它也仍然有效。窗口看门狗由从 APB1 时钟分频后得到的时钟驱动，通过可配置的时间窗口来检测应用程序非正常的过迟或过早的操作。IWDG 最适合应用于那些需要看门狗作为一个在主程序之外，能够完全独立工作，并且对时间精度要求较低的场合。WWDG 最适合那些要求看门狗在精确计时窗口起作用的应用程序。

这里面我们举一个例子，设置 IWDG 的实验，也就是独立看门狗。我们演示了独立看门狗重导计数器如何在一个常规周期内升级并且在设定周期内通过模拟软件错误产生 MCU 的独立看门狗复位。正常情况向程序在实现无限循环，当通过外部中断模拟一个软件错误，就要进行喂狗，整个循环中断跳出，看门狗进行系统复位。这就是我们需要达到的效果。下面就通过软硬件方面进行分析：

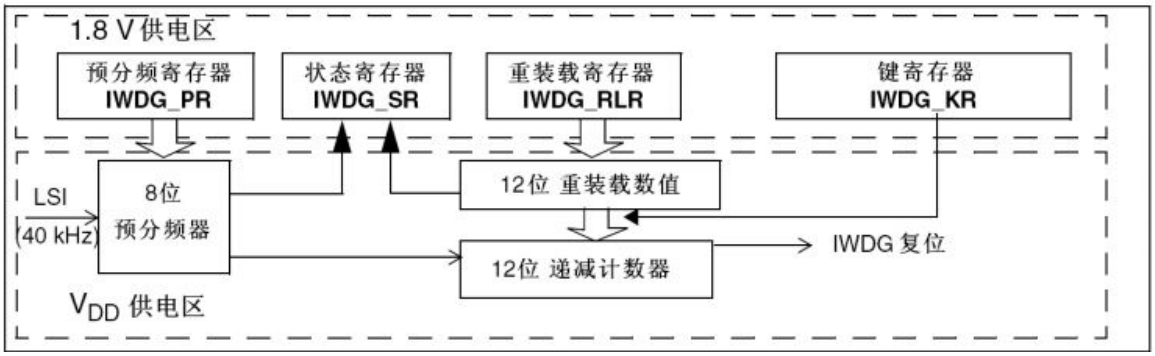
2.5.1 软件准备：

打开 keil 编译环境，设置系统工程树如下图所示：



如上图所示，我们需要配置的就是 IWDG 和定时器 TIM14 的设置，库函数调用 stm32f0xx_iwdg.c,中断函数在 stm32f0xx_it.c 函数内。独立看门狗的整个系统结构如下图所示：

图157 独立看门狗框图



在设置定时器时我们主要配置 TIM14 去测量 LSI 振荡频率，在中断中，写一个无效地址产生一个硬故障异常使得无法回到主程序（这时 IWDG 从导计数器没有被更新），设置代码如下：

```
/* 使能外设时钟 ------------------------------------------------*/
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_PWR, ENABLE);

  /* 允许访问的 RTC */
  PWR_BackupAccessCmd(ENABLE);

  /* 复位 RTC */
  RCC_BackupResetCmd(ENABLE);
  RCC_BackupResetCmd(DISABLE);

  /*!LSI 使能 */
  RCC_LSICmd(ENABLE);

```

```
14.    /*!< 等到 LSI 已准备就绪 */
15.    while (RCC_GetFlagStatus(RCC_FLAG_LSIRDY) == RESET)
16.    {}
17.
18.    /* 选择 RTC 时钟源*/
19.    RCC_RTCCLKConfig(RCC_RTCCLKSource_LSI);
20.
21.    /* 使能 RTC 时钟*/
22.    RCC_RTCCLKCmd(ENABLE);
23.
24.    /* 等待 RTC APB 寄存器同步*/
25.    RTC_WaitForSynchro();
26.
27.    /* 使能 TIM14 时钟 */
28.    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM14, ENABLE);
29.
30.    /* 使能 TIM14 中断 */
31.    NVIC_InitStructure.NVIC_IRQChannel = TIM14_IRQn;
32.    NVIC_InitStructure.NVIC_IRQChannelPriority = 0;
33.    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
34.    NVIC_Init(&NVIC_InitStructure);
35.
36.    /* 配置 TIM14 预分频器        */
37.    TIM_PrescalerConfig(TIM14, 0, TIM_PSCReloadMode_Immediate);
38.
39.    /* 连接内部的 TM14_CH1 输入捕获到 LSI 时钟输出*/
40.    TIM_RemapConfig(TIM14, TIM14_RTC_CLK);
```

上面同时设定了定时器 TIM14 的嵌套中断，中断函数中我们要计算出 LsiFreq，也就是 LSI 的时钟频率,可以按照下面方式进行设定：

```
41. void TIM14_IRQHandler(void)
42. {
43.    if (TIM_GetITStatus(TIM14, TIM_IT_CC1) != RESET)
44.    {
45.      if(CaptureNumber == 0)
46.      {
47.        /* 获得输入比较的值 */
48.        IC1ReadValue1 = TIM_GetCapture1(TIM14);
49.      }
50.      else if(CaptureNumber == 1)
51.      {
52.        /* 获得输入比较的值 */
53.        IC1ReadValue2 = TIM_GetCapture1(TIM14);
54.
55.        /* 比较 2 次的值 */
```

```
56.         if (IC1ReadValue2 > IC1ReadValue1)
57.         {
58.           Capture = (IC1ReadValue2 - IC1ReadValue1);
59.         }
60.         else
61.         {
62.           Capture = ((0xFFFF - IC1ReadValue1) + IC1ReadValue2);
63.         }
64.         /*   频率计算*/
65.         LsiFreq = (uint32_t) SystemCoreClock / Capture;
66.         LsiFreq *= 8;
67.       }
68.
69.       CaptureNumber++;
70.
71.       /* 清除 TIM4 捕获比较中断位*/
72.       TIM_ClearITPendingBit(TIM14, TIM_IT_CC1);
73.     }
74. }
```

然后我们需要确定定时器的工作方式：

```
75.   /* TIM14 配置: 输入捕获模式 ---------------------
76.        LSI 时钟与 TIM14 CH1 通道相连
77.        上升沿被用作有效边沿,
78.        TIM14 CCR1 被用来计算频率值
79.   ------------------------------------------------------------ */
80.   TIM_ICInitStructure.TIM_Channel = TIM_Channel_1;
81.   TIM_ICInitStructure.TIM_ICPolarity = TIM_ICPolarity_Rising;
82.   TIM_ICInitStructure.TIM_ICSelection = TIM_ICSelection_DirectTI;
83.   TIM_ICInitStructure.TIM_ICPrescaler = TIM_ICPSC_DIV8;
84.   TIM_ICInitStructure.TIM_ICFilter = 0;
85.   TIM_ICInit(TIM14, &TIM_ICInitStructure);
86.
87.   /* TIM14 计数器使能   */
88.   TIM_Cmd(TIM14, ENABLE);
```

上面的工作做完后，我们就可以通过 TIM14 定时器来确定 LIS 的工作频率，那么 IWDG 需要通过 LSI 的工作频率驱动装载值，试验中，我们设 240ms 的循环时间，250ms 为超时时间，当超过 250ms 的时候认为出现了软件错误，需要喂狗。在定时方面采用系统时钟滴答精确定时，如下代码所示：

```
89. /*  设置系统滴答时钟为 1s/1000=1ms 发生一次中断  */
90.   if (SysTick_Config(SystemCoreClock / 1000))
91.   {
92.     /* Capture error */
93.     while (1);
94.   }
```

正常工作状态下，没有发生软件错误，LED2 灯按 240ms 的要求进行循环翻转：

```
while (1)
  {
    /* 翻转 LED2 */
   LED2_Toggle();

    /* 加入 240 ms 延迟 */
    Delay(240);

    /* 从新导入 IWDG 计数器 */
    IWDG_ReloadCounter();
  }
```

但是当发生了软件错误后，我们需要喂狗，喂狗的操作就是打开 LED1 灯：

```
/* 检测系统是否由独立看门狗喂狗复位 */
  if (RCC_GetFlagStatus(RCC_FLAG_IWDGRST) != RESET)
  {
    /* 独立看门狗标志位被置位， 则执行打开 LED1 灯*/
    LED1_Open();

    /* 清除复位标志位 */
    RCC_ClearFlag();
  }
  else
  {
    /* 独立看门狗标志位没有被置位， 则执行关掉 LED1 灯*/
    LED1_Close();
  }
```

软件错误我们通过按键中断进行模拟，如下所示，设置一个无效的访问：

```
void EXTI4_15_IRQHandler(void)
{
  if (EXTI_GetITStatus(EXTI_Line7) != RESET)
  {
    /* Clear the USER Button EXTI Line Pending Bit */
    EXTI_ClearITPendingBit(EXTI_Line7);

    /* As the following address is invalid (not mapped), a Hardfault exception
       will be generated with an infinite loop and when the IWDG counter falls to 63
       the IWDG reset occurs */
    *(__IO uint32_t *) 0x00040001 = 0xFF;
  }
}
```

最后 IWDG 的相关参数我们在主函数中如下代码进行配置：

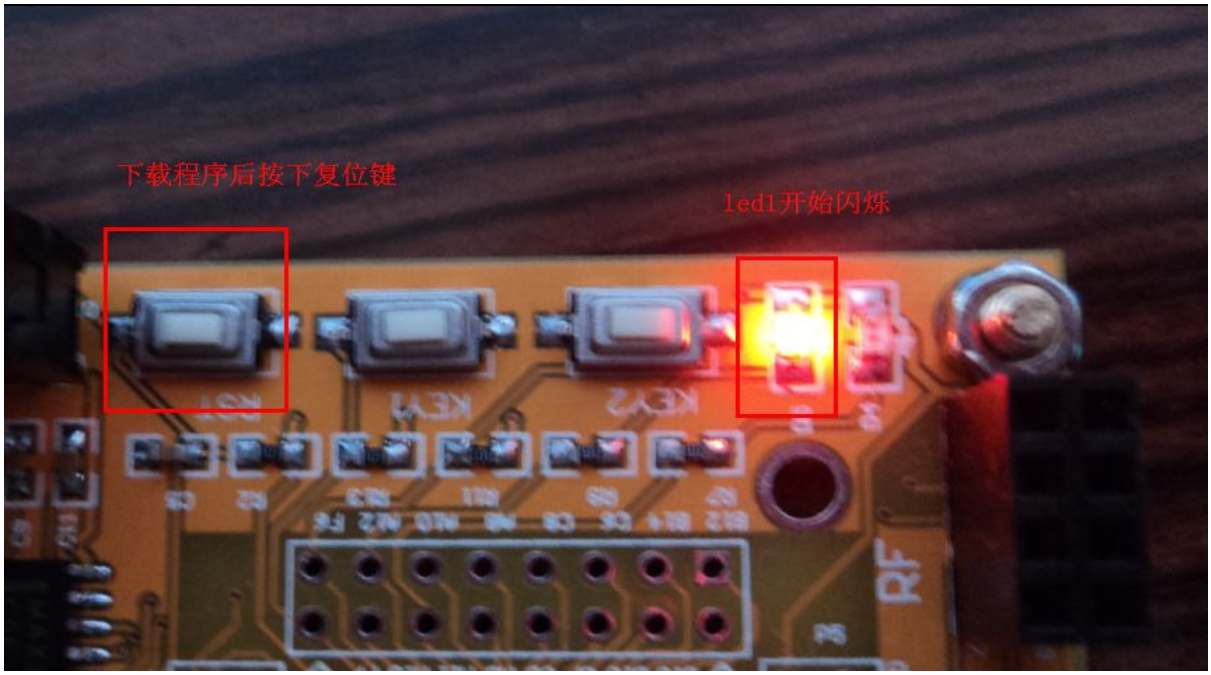
```
#ifdef LSI_TIM_MEASURE
  /* TIM 配置 ----------------------------------------------------*/
```

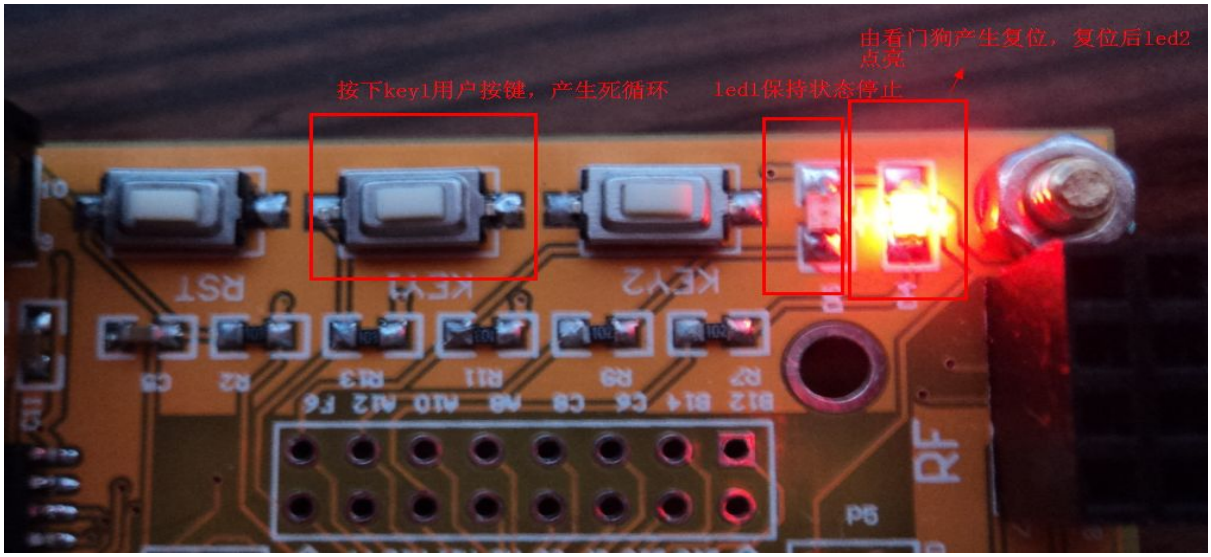
```
30.    TIM14_ConfigForLSI();
31.
32.    /* 等待当 TIM14 捕获 2 个 LSI 边沿 */
33.    while(CaptureNumber != 2)
34.    {
35.    }
36.    /* 关闭 TIM14 CC1 中断应答 */
37.    TIM_ITConfig(TIM14, TIM_IT_CC1, DISABLE);
38. #endif
39.
40.    /* 使能写访问 IWDG_PR and IWDG_RLR 寄存器 */
41.    IWDG_WriteAccessCmd(IWDG_WriteAccess_Enable);
42.
43.    /* IWDG 计数器的时钟: LSI/32 */
44.    IWDG_SetPrescaler(IWDG_Prescaler_32);
45.
46.    /* 设置重导计数器的值 获得 250ms 的 IWDG 超时可以按下面算式计算：
47.        Counter Reload Value = 250ms/IWDG counter clock period
48.                             = 250ms / (LSI/32)
49.                             = 0.25s / (LsiFreq/32)
50.                             = LsiFreq/(32 * 4)
51.                             = LsiFreq/128
52.     */
53.    IWDG_SetReload(LsiFreq/128);
54.
55.    /* 从新导入 IWDG 计数器 */
56.    IWDG_ReloadCounter();
57.
58.    /* 使能 IWDG (LSI 被硬件使能) */
59.    IWDG_Enable();
60.
```

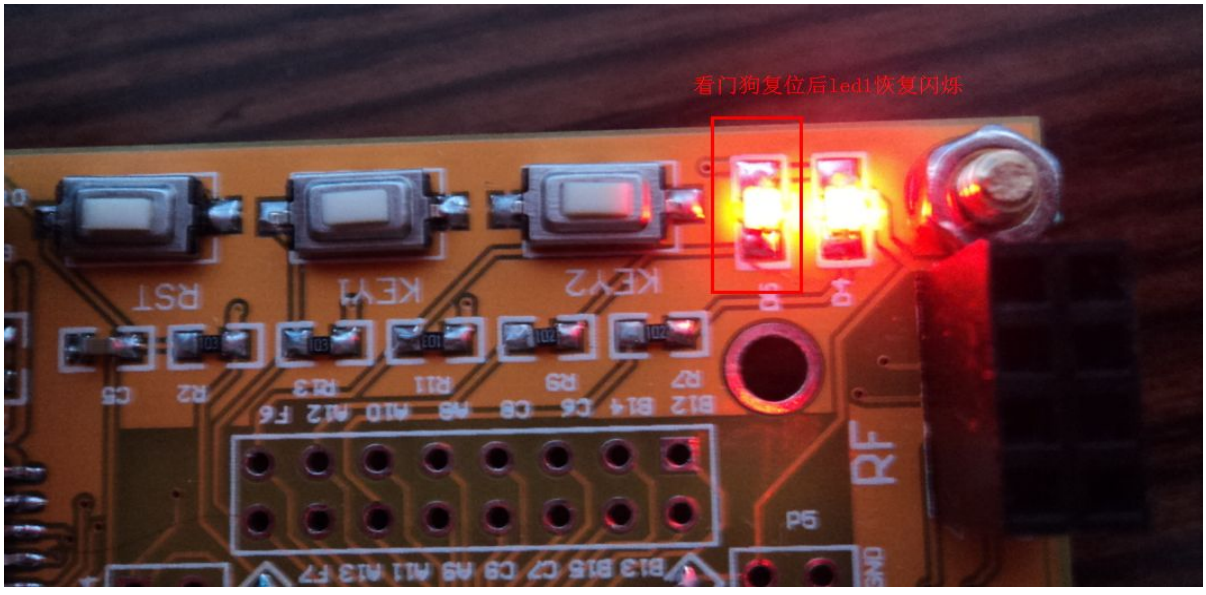
实验现象：

通过 key 中断模拟死循环，通过看门狗喂狗，led 灯观察现象：

030 豪华版请找到对应的 LED 灯，显示效果和 051 开发板类似


下载程序后按下复位键
led1开始闪烁


由看门狗产生复位，复位后led2点亮
按下key1用户按键，产生死循环
led1保持状态停止


看门狗复位后led1恢复闪烁