实验 6 - 看门狗实验

1. 实验目的

掌握 NRF24LE1 的看门狗的配置和使用。

读取复位原因代码，判断芯片上一次的复位原因。

2. 实验内容

配置 NRF24LE1 的看门狗 10 秒超时复位芯片，演示打开看门狗后没有喂狗系统不断复位的情况。

程序获取上一次的复位原因，并通过串口输出。

3. 实验原理

1. 寄存器配置

表 1：看门狗寄存器

地址	名称	位	复位值	类型	说明
0xAF	WDSV	15:0	0x0000	RW	看门狗初始值寄存器。 字的高位字节和低位字节必须单独读/写。

2. 看门狗超时时间计算

超时时间 = (WDSV * 256)/32768。

所以：最小看门狗超时周期 = 7.8125ms。

最大看门狗超时周期 = 512s。

按照上述内容，要配置看门狗的超时时间为 10 秒，可按照下述方式进行：为 16 位定时器，定时时间为 50ms，产生溢出中断的代码如下：

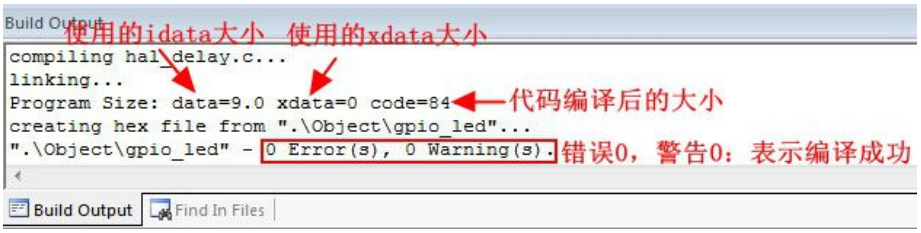
- 计算 WDSV 值：超时时间 10 秒对应的 WDSV = 0x500;
- 将 0x500 写入 WDSV，注意，写入 WDSV 时，必须是两个连续的操作。两个字节写入后，看门狗将被激活。
  - WDSV = LSB(start_value); // Write the 8 LSB to the WD counter
  - WDSV = MSB(start_value); // Write the 8 MSB to the WD counter

在程序中，直接调用库函数 hal_wdog_init(WDSV_10S);参数为超时时间，即可配置并使能看门狗。

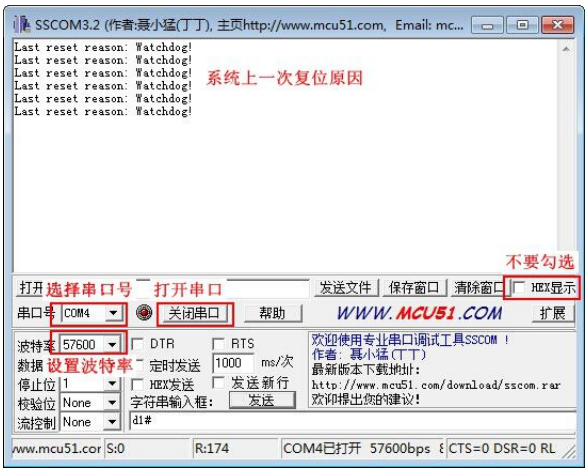
4. 实验步骤

- 在 Keil uVision4 中打开工程“watchdog.uvproj”工程；
- 编译工程，注意查看编译输出栏，观察编译的结果，如果有错误，修改程序，直到

编译成功为止；

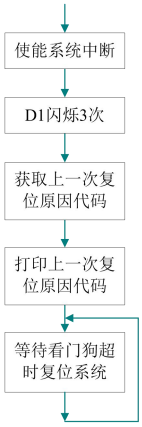


- 将编译生成的 HEX 文件“watchdog.hex” (该文件位于工程目录下的“Object”文件夹中)通过编程器下载到开发板中运行。
- 观察指示灯 D1，芯片复位后 D1 会连续闪烁 3 次。
- 打开串口调试助手，选择串口号，设置波特率为 57600，打开串口，注意不要勾选“HEX 显示”。观察串口输出的信息。串口会输出上一次芯片复位的原因。



5. 实验程序

5.1. 程序流程



5.2. 程序清单

```
#define   D1      P00   //开发板上的指示灯 D1
```

```
#define   WDSV_10S      1280   //看门狗超时时间：10S

uint8_t count=0;    //软件计数变量

/*******************************************************************************
 *描   述 ：配置 IO P0.0 为输出，驱动 LED。P03 输出: UART TXD，P04:输入 UART RXD
 *入   参 ：无
 *返回值 ：无
*******************************************************************************/
void IO_Init(void)
{
    P0DIR &= ~0x01;     //配置 P0.0 为输出
    P0DIR &= ~0x08;     //P03:输出 UART TXD
    P0DIR |= 0x10;      //P04:输入 UART RXD
    D1 = 1;     //设置 D1 初始状态为熄灭
}
/*******************************************************************************
 *描   述 ：配置 Timer0 为 16 位定时器，定时时间 20ms，开启中断
 *入   参 ：无
 *返回值 ：无
*******************************************************************************/
void ClockInit(void)
{
  hal_rtc_start(false);   //关闭 32.768KHz 时钟
  hal_clklf_set_source(HAL_CLKLF_RCOSC32K); //32.768KHz 的时钟源为内部 RC

  hal_rtc_start(true);    //关闭 32.768KHz 时钟

  while((CLKLFCTRL&0x80)==0x80); //等待时钟启动
  while((CLKLFCTRL&0x80)!=0x80);
}
/*******************************************************************************
 *描   述 ：串口打印字符串
 *入   参 ：无
 *返回值 ：无
*******************************************************************************/
void PutString(char *s)
{
  while(*s != 0)
    hal_uart_putchar(*s++);
}

/*******************************************************************************
```

```
 *描   述 ：主函数
 *入   参 ：无
 *返回值 ：无
*******************************************************************************/
void main(void)
{
  uint8_t RstReason;
  uint8_t i;

  IO_Init();   //初始化 IO
  hal_uart_init(UART_BAUD_57K6); // 初始化 UART，波特率 57600
  while(hal_clk_get_16m_source() != HAL_CLK_XOSC16M) // 等待时钟稳定
  ;

  EA = 1;             // 开启全局中断

  ClockInit(); //初始化时钟

  for(i=0;i<6;i++)
  {
       D1 = ~D1;
       delay_ms(100);
  }

  RstReason = RSTREAS;   //获取上一次复位原因代码
  RstReason &= 0x07;

  switch(RstReason)
  {
      case 0x00:    //On-chip reset generator
         PutString("Last reset reason: On-chip reset generatorOn-chip reset generator!\n");
         break;

      case 0x01:    //RST pin
         PutString("Last reset reason: RST pin!\n");
         break;

      case 0x02:    //Watchdog
         PutString("Last reset reason: Watchdog!\n");
         break;

      case 0x04:   //Reset from on-chip hardware debugger
         PutString("Last reset reason: Reset from on-chip hardware debugger!\n");
```

```
            break;

        default:
            break;
    }
    RSTREAS = 0; //RSTREAS 的值是累加的，所以，读以后要清除
    hal_wdog_init(WDSV_10S);  //配置看门狗超时时间 10s，使能看门狗

    while(1);  //等待看门狗超时复位系统
}
```