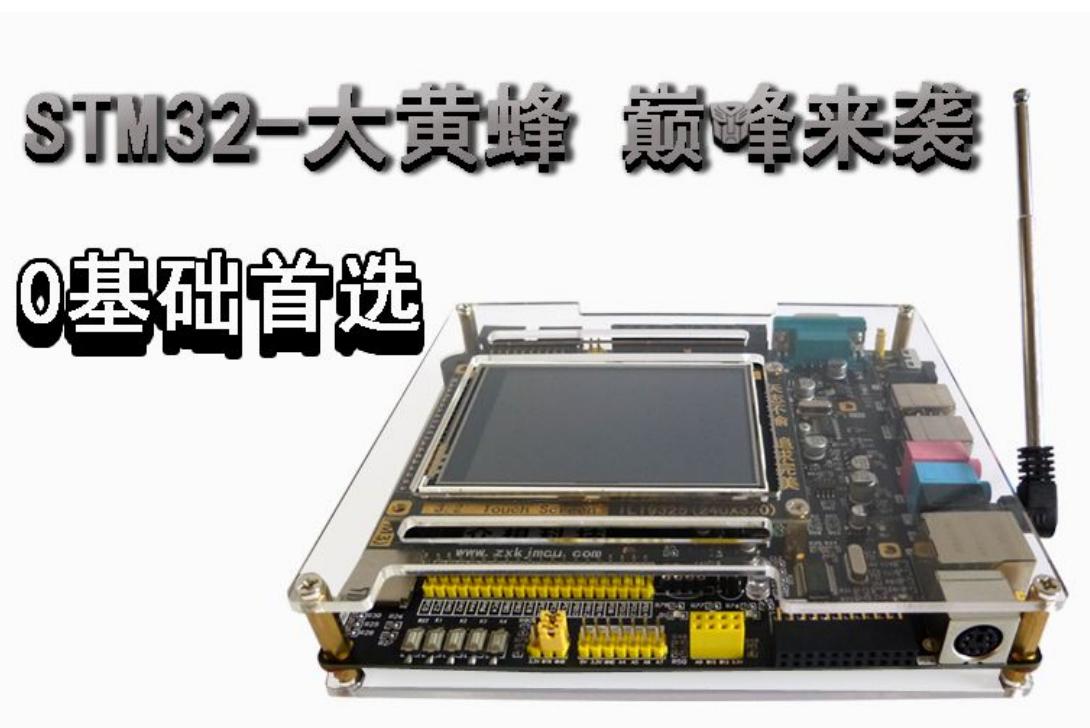


# 学 ARM 从 STM32 开始

STM32 开发板库函数教程—实战篇



官方网站: <http://www.zxkjmcu.com>

官方店铺: <http://zxkjmcu.taobao.com>

官方论坛: <http://bbs.zxkjmcu.com>

刘洋课堂: <http://school.zxkjmcu.com>

## 4.18 STM32 窗口看门狗的工作原理与实验

### 4.18.1 概述

#### 4.18.1.1 看门狗概念

参见第 4.17.1.1 介绍。

#### 4.18.1.2 看门狗设计原理

参见第 4.17.1.2 介绍。

### 4.18.2 STM32 内置看门狗简介

4.18.2.1 STM32F10xxx 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(IWDG)

窗口看门狗(WWDG))

它们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。

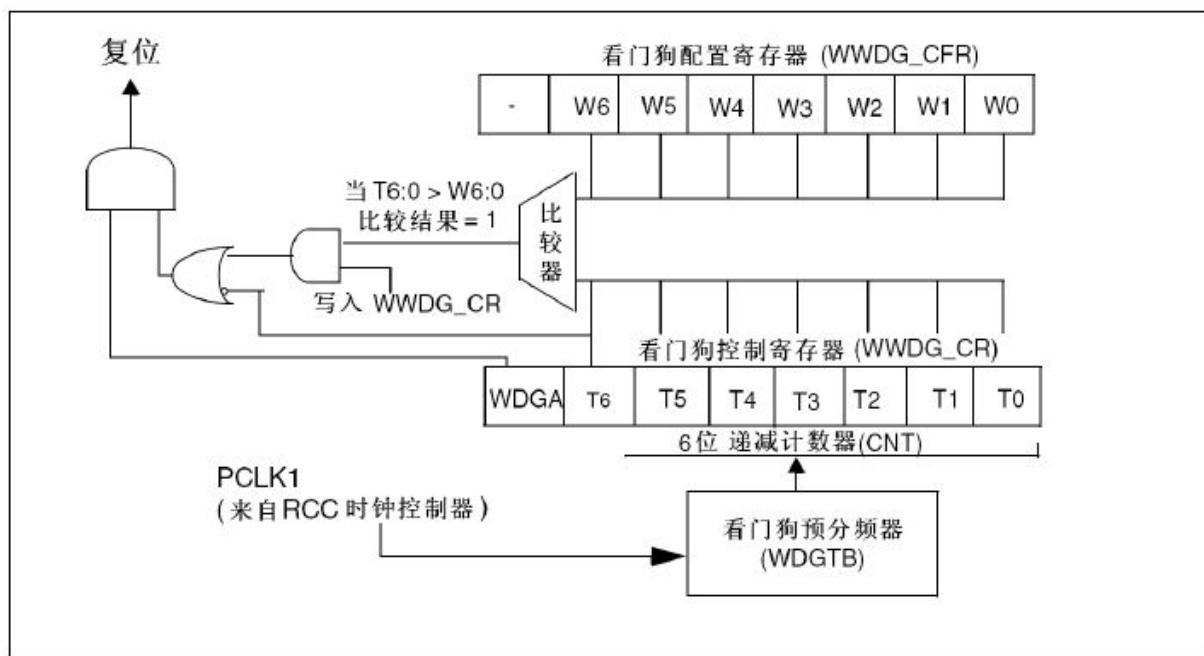
窗口看门狗(WWDG)通常被用来监测，由外部干扰或不可预见的逻辑条件造成应用程序背离正常的运行序列而产生的软件故障。除非递减计数器的值在 T6 位变成 0 前被刷新，看门狗电路在达到预置的时间周期时，会产生一个 MCU 复位。在递减计数器达到窗口寄存器数值之前，如果 7 位的递减计数器数值(在控制寄存器中)被刷新，那么也将产生一个 MCU 复位。这表明递减计数器需要在一个有限的时间窗口中被刷新。

#### 4.18.2.2 WWDG 主要特性

- 可编程的自由运行递减计数器
- 条件复位
  - 当递减计数器的值小于 0x40，(若看门狗被启动)则产生复位。
  - 当递减计数器在窗口外被重新装载，(若看门狗被启动)则产生复位。
- 如果启动了看门狗并且允许中断，当递减计数器等于 0x40 时产生早期唤醒中断(EWI)，它可以被用于重装载计数器以避免 WWDG 复位。

#### 4.18.2.2 WWDG 功能描述

如果看门狗被启动(WWDG\_CR 寄存器中的 WDGA 位被置‘1’)，并且当 7 位(T[6:0])递减计数器从 0x40 翻转到 0x3F(T6 位清零)时，则产生一个复位。如果软件在计数值大于窗口寄存器中的数值时重新装载计数器，将产生一个复位。



图一 窗口看门狗框图

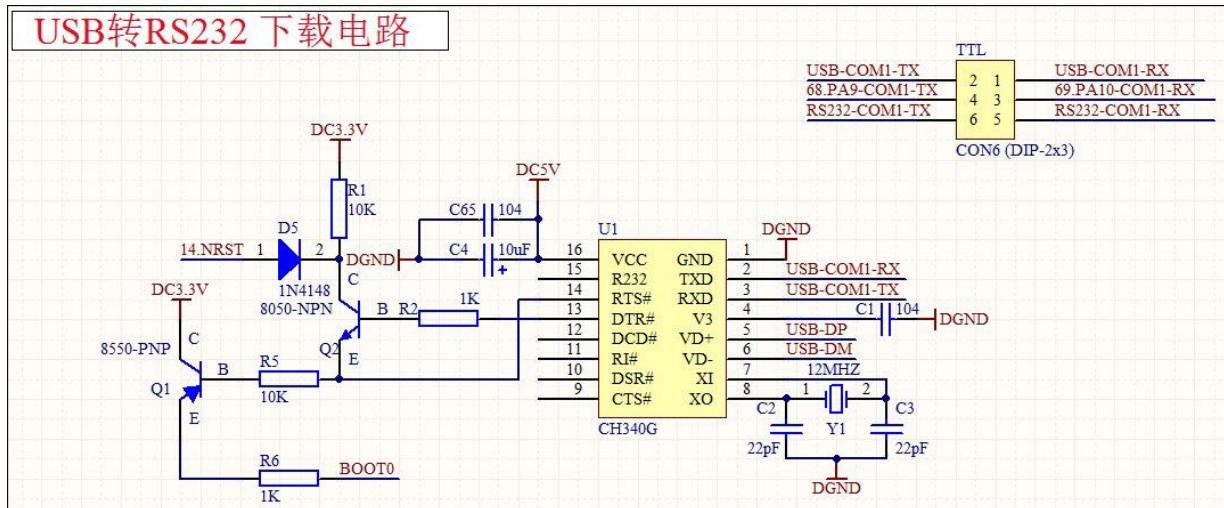
#### 4.18.3 实验目的

- 1、通过下传编写好的看门狗复位程序，看门狗复位后，通过 prinif 重

定向程序把公司名称打印到串口精灵上，也就是在显示器上显示出来了。

#### 4.18.4 硬件设计

利用实验板上的串口电路，可以很方便的实现这个功能。

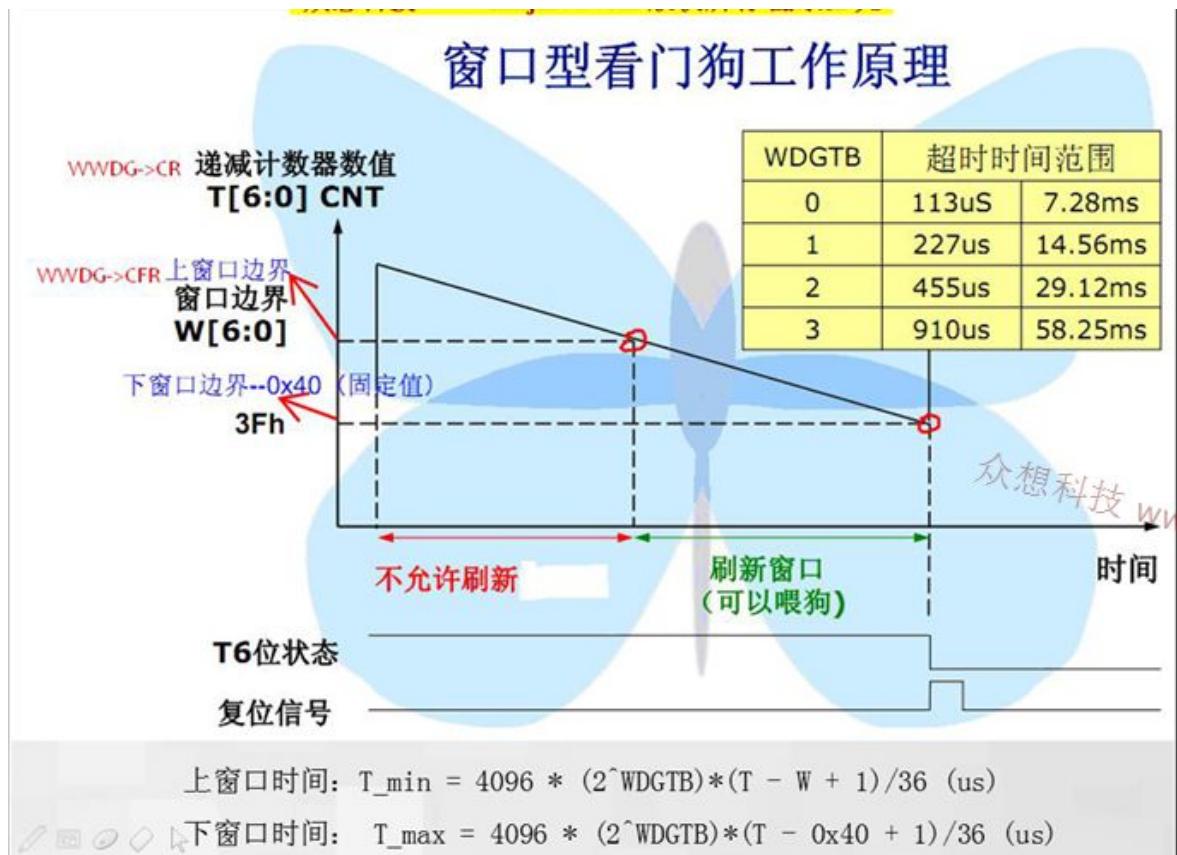


#### 4.17.5 软件设计

软件设计和 prinif 重定向程序很相似，只有那么 3 点不同，下面对软件的设计做一次说明：

- 1、设置 STM32 内部独立看门狗功能；
  - 2、每隔 500ms 定期“喂狗”；
  - 3、当不“喂狗”的时候，打印输出公司名称。

每次显示公司名称的时候，说明看门狗执行成功，给 CPU 复位。程序从最初步执行。



图三 窗口看门狗刷新区间范围

#### 4.17.5.1 STM32 库函数文件

```

stm32f10x_gpio.c
stm32f10x_rcc.c
Misc.c // 中断控制字（优先级设置）库函数
stm32f10x_exti.c // 外部中断库处理函数
stm32f10x_tim.c // 定时器库处理函数
stm32f10x_usart.c // 串口通讯函数
stm32f10x_iwdg.c // 独立看门狗函数

```

本节实验及以后的实验我们都是用到库文件，其中 `stm32f10x_gpio.h` 头文件包含了 GPIO 端口的定义。`stm32f10x_rcc.h` 头文件包含了系统时钟配置函数以及相关的外设时钟使能函数，所以我们要把这两个头文件对应的 `stm32f10x_gpio.c` 和 `stm32f10x_rcc.c` 加到工程中；`Misc.c` 库函数主要包含了中断优先级的设置，`stm32f10x_exti.c` 库函数主要包含了外部中断设置参数，`tm32f10x_tim.c` 库函数主要包含定时器设置，`tm32f10x_usart.c`

库函数主要包含串行通讯设置，stm32f10x\_iwdg.c 库函数主要包含独立看门狗应用设置，这些函数也要添加到函数库中。以上库文件包含了本次实验所有要用到的函数使用功能。

#### 4.17.5.2 自定义头文件

```
pbdata.h  
pbdata.c
```

同时我们自己也创建了两个公共的文件，这两个文件主要存放我们自定义的公共函数和全局变量，以方便以后每个功能模块之间传递参数。

#### 4.17.5.3 pbdata.h 文件里的内容是

```
#ifndef _pbdata_H  
#define _pbdata_H  
  
#include "stm32f10x.h"  
#include "misc.h"  
#include "stm32f10x_exti.h"  
#include "stm32f10x_tim.h"  
#include "stm32f10x_usart.h"  
#include "stm32f10x_iwdg.h"  
#include "stddio.h"  
  
extern u8 dt;//定义变量  
  
void RCC_HSE_Configuration(void); //定义函数  
void delay(u32 nCount);  
void delay_us(u32 nus);  
void delay_ms(u16 nms);  
  
#endif
```

语句 #ifndef、#endif 是为了防止 pbdata.h 文件被多个文件调用时出现错误提示。如果不加这两条语句，当两个文件同时调用 pbdata 文件时，会提示重复调用错误。

#### 4.17.5.4 pbdata.c 文件里的内容是

```
#include "pbdata.h" //很重要，引用这个头文件
```

```
u8 dt=0;

void RCC_HSE_Configuration(void) //HSE 作为 PLL 时钟, PLL 作为 SYSCLK
{
    RCC_DeInit(); /*将外设 RCC 寄存器重设为缺省值 */
    RCC_HSEConfig(RCC_HSE_ON); /*设置外部高速晶振 (HSE) HSE 晶振打开(ON)*/

    if(RCC_WaitForHSEStartUp() == SUCCESS) { /*等待 HSE 起振, SUCCESS: HSE 晶振稳定且就绪*/

        RCC_HCLKConfig(RCC_SYSCLK_Div1);/*设置 AHB 时钟(HCLK)RCC_SYSCLK_Div1——
        AHB 时钟 = 系统时*/
        RCC_PCLK2Config(RCC_HCLK_Div1); /*设置高速 AHB 时钟(PCLK2)RCC_HCLK_Div1——
        APB2 时钟 = HCLK*/
        RCC_PCLK1Config(RCC_HCLK_Div2); /*设置低速 AHB 时钟(PCLK1)RCC_HCLK_Div2——
        APB1 时钟 = HCLK / 2*/

        RCC_PLLConfig(RCC_PLLSource_HSE_Div1, RCC_PLLMul_9);/*设置 PLL 时钟源及倍频
        系数*/
        RCC_PLLCmd(ENABLE); /*使能 PLL */
        while(RCC_GetFlagStatus(RCC_FLAG_PLLRDY) == RESET); /*检查指定的 RCC 标志
        位(PLL 准备好标志)设置与否*/

        RCC_SYSCLKConfig(RCC_SYSCLKSource_PLLCLK); /*设置系统时钟 (SYSCLK) */
        while(RCC_GetSYSCLKSource() != 0x08); /*0x08: PLL 作为系统时钟 */

    }
}

void delay(u32 nCount)
{
    for(;nCount!=0;nCount--);

}/*
*****
* 名    称: delay_us(u32 nus)
* 功    能: 微秒延时函数
* 入口参数: u32  nus
* 出口参数: 无
* 说    明:
* 调用方法: 无
*****/
void delay_us(u32 nus)
{
    u32 temp;
    SysTick->LOAD = 9*nus;
```

```
SysTick->VAL=0X00;//清空计数器
SysTick->CTRL=0X01;//使能，减到零是无动作，采用外部时钟源
do
{
    temp=SysTick->CTRL;//读取当前倒计数值
}while((temp&0x01)&&(!(temp&(1<<16))));//等待时间到达

SysTick->CTRL=0x00; //关闭计数器
SysTick->VAL =0X00; //清空计数器
}

/*****************
* 名      称: delay_ms(u16 nms)
* 功      能: 毫秒延时函数
* 入口参数: u16 nms
* 出口参数: 无
* 说      明:
* 调用方法: 无
********************/
void delay_ms(u16 nms)
{
    u32 temp;
    SysTick->LOAD = 9000*nms;
    SysTick->VAL=0X00;//清空计数器
    SysTick->CTRL=0X01;//使能，减到零是无动作，采用外部时钟源
    do
    {
        temp=SysTick->CTRL;//读取当前倒计数值
    }while((temp&0x01)&&(!(temp&(1<<16))));//等待时间到达
    SysTick->CTRL=0x00; //关闭计数器
    SysTick->VAL =0X00; //清空计数器
}
```

#### 4.17.6 STM32 系统时钟配置 SystemInit()

每个工程都必须在开始时配置并启动 STM32 系统时钟。

#### 4.17.7 GPIO 引脚时钟使能

```
SystemInit() ;//72m
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE) ;
RCC_APB1PeriphClockCmd(RCC_APB2Periph_USART1, ENABLE) ;//设置串口 1 时钟使能
RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO, ENABLE) ;//功能复用 IO 时钟使能
```

本节实验用到了 PA 端口，所以要把 PA 端口的时钟打开；串口 1 时钟源是通过 APB2 预分频器得到的，串口 1 时钟初始化；因为要与外部芯片通讯，所以要打开功能复用时钟。

#### 4.17.8 GPIO 管脚电平控制函数

在主程序中采用 while(1) 循环语句，等待串口通讯中断的到来。

```
while(1)
{
    //喂狗
    //IWDG_ReloadCounter(); //按照 IWDG 重装载寄存器的值重装载 IWDG 计数器
    printf("喂狗\r\n");
    delay_ms(500);
}
```

在程序中有 “\r\n” ，它的意思是回车换行的意思，在程序语句尾部加上 “\r\n” ，在把数据打印到屏幕上时，会自动回车换上打印屏幕的。

IWDG\_ReloadCounter() 函数是重装载看门狗计数器，当在规定的时间内执行这个函数，看门狗计数器就不溢出，不会产生复位信号。如果超出规定的时间内没有执行这个函数，看门狗计数器就会溢出，产生复位信号是单片机复位，程序重新初始化。

#### 4.17.9 stm32f10x\_it.c 文件里的内容是

在中断处理 stm32f10x\_it.c 文件里中仅串口 1 子函数非空，进入中断处理函数后，只有串口 1 有参数输出。

```
#include "stm32f10x_it.h"
#include "stm32f10x_exti.h"
#include "stm32f10x_rcc.h"
#include "misc.h"
#include "pbdata.h"

void NMI_Handler(void)
{
```

```
}

void USART1_IRQHandler(void)
{
    if(USART_GetITStatus(USART1, USART_IT_RXNE) !=RESET)
    {
        USART_SendData(USART1, USART_ReceiveData(USART1));
        while(USART_GetFlagStatus(USART1, USART_FLAG_TXE)==RESET);
    }
}
```

#### 4.17.10 main.c 文件里的内容是

大家都知道 `prinif` 重定向是把需要显示的数据打印到显示器上。在这个试验中主程序不断执行“喂狗”函数，使看门狗不产生复位信号。在显示器上每隔 500ms 打印出“喂狗”字符，如果把“`IWDG_ReloadCounter()`”函数封死，因为没有了在规定时间内执行“重装载寄存器的值重装载 IWDG 计数器”操作，就会产生看门狗复位信号，程序重新初始化，就会显示出公司名称和网址。

```
#include "pbdata.h"

void RCC_Configuration(void);
void GPIO_Configuration(void);
void NVIC_Configuration(void);
void USART_Configuration(void);
void IWDG_Configuration(void);

int fputc(int ch, FILE *f)
{
    USART_SendData(USART1, (u8)ch);
    while(USART_GetFlagStatus(USART1, USART_FLAG_TXE)==RESET);
    return ch;
}

int main(void)
{
    RCC_Configuration(); //系统时钟初始化
    GPIO_Configuration(); //端口初始化
    USART_Configuration();
```

```
NVIC_Configuration();
IWDG_Configuration();

printf("众想科技\r\n");
printf("www.zxkjmcu.com\r\n");

while(1)
{
    //喂狗
    //IWDG_ReloadCounter();
    printf("喂狗\r\n");
    delay_ms(500);
}

void IWDG_Configuration(void)
{
    //使能寄存器 写功能
    IWDG_WriteAccessCmd(IWDG_WriteAccess_Enable);
    //设置预分频 40K/64=0.625k 一个周期是 1.6ms
    IWDG_SetPrescaler(IWDG_Prescaler_64);
    //设置初值
    IWDG_SetReload(800); //800*1.6ms=1.28S
    //喂狗
    IWDG_ReloadCounter();
    //使能独立看门狗
    IWDG_Enable();
}

void RCC_Configuration(void)
{
    SystemInit(); //72m
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1, ENABLE);
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO, ENABLE);
}

void GPIO_Configuration(void)
{
    GPIO_InitTypeDef GPIO_InitStructure;
    //LED
    GPIO_InitStructure.GPIO_Pin=GPIO_Pin_9;//TX
    GPIO_InitStructure.GPIO_Speed=GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_Mode=GPIO_Mode_AF_PP;
```

```
GPIO_Init(GPIOA, &GPIO_InitStructure);

    GPIO_InitStructure.GPIO_Pin=GPIO_Pin_10;//RX
    GPIO_InitStructure.GPIO_Mode=GPIO_Mode_IN_FLOATING;
    GPIO_Init(GPIOA, &GPIO_InitStructure);
}

void NVIC_Configuration(void)
{
    NVIC_InitTypeDef NVIC_InitStructure;

    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_1);

    NVIC_InitStructure.NVIC_IRQChannel = USART1_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 1;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);
}

void USART_Configuration(void)
{
    USART_InitTypeDef USART_InitStructure;

    USART_InitStructureUSART_BaudRate=9600;
    USART_InitStructureUSART_WordLength=USART_WordLength_8b;
    USART_InitStructureUSART_StopBits=USART_StopBits_1;
    USART_InitStructureUSART_Parity=USART_Parity_No;
    USART_InitStructureUSART_HardwareFlowControl=USART_HardwareFlowContr
ol_None;
    USART_InitStructureUSART_Mode=USART_Mode_Rx|USART_Mode_Tx;

    USART_Init(USART1, &USART_InitStructure);
    USART_ITConfig(USART1, USART_IT_RXNE, ENABLE);
    USART_Cmd(USART1, ENABLE);
    USART_ClearFlag(USART1, USART_FLAG_TC);
}
```

在 main(void) 程序体中代码比较多，大部分都是前期课程的延伸，新的知识点不多，请大家注意看门狗的设置和“喂狗”时间的计算，“喂狗”的时间一定要控制在看门狗产生复位信号之前。

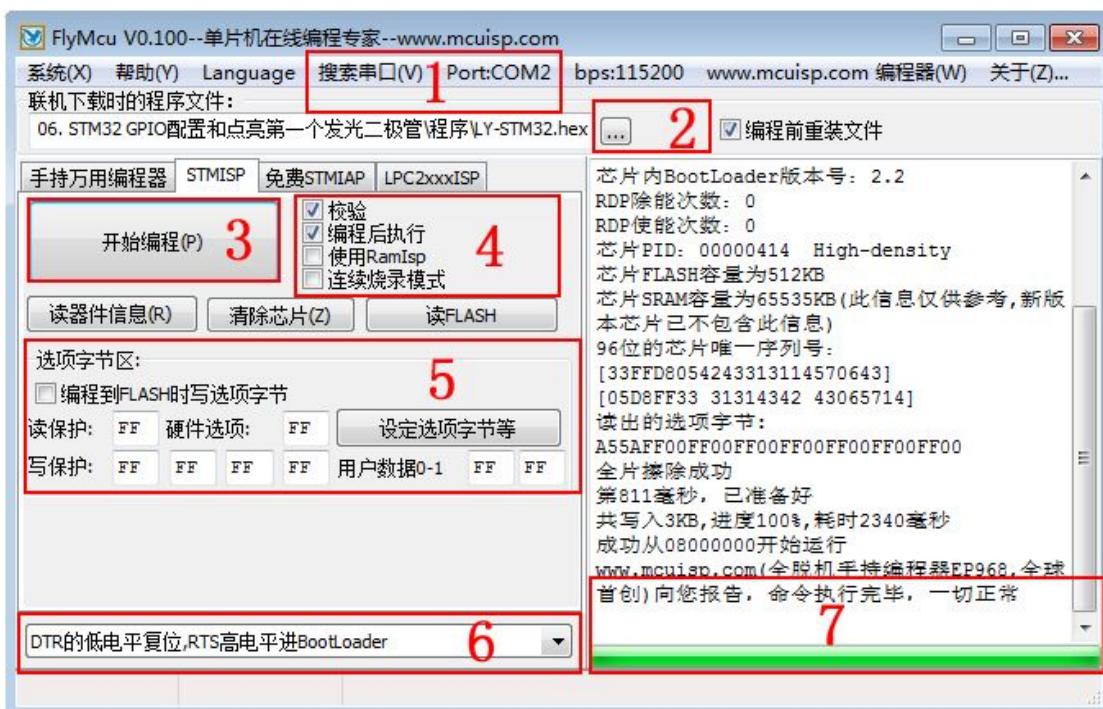
#### 4.17.11 函数 IWDG\_ReloadCounter

表 3 描述了函数 IWDG\_ReloadCounter

函数名称	IWDG_ReloadCounter
函数原型	void IWDG_ReloadCounter(void)
功能描述	按照 IWDG 重装载寄存器的值重装载 IWDG 计数器
输入参数	无
输出参数	无
返回值	无
先决条件	无
被调用函数	无

#### 4.17.12 程序下载

请根据下图所指向的 7 个重点区域配置。其中（1）号区域根据自己机器的实际情况选择，我的机器虚拟出来的串口号是 COM2。（2）号区域请自己选择程序所在的文件夹。（7）号区域当程序下载完后，进度条会到达最右边，并且提示一切正常。（4、5、6）号区域一定要按照上图显示的设置。当都设置好以后就可以直接点击（3）号区域的开始编程按钮下传程序了。



本节实验的源代码在光盘中：(LY-STM32 光盘资料\1. 课程\1, 基础篇 30. STM32 独立看门狗工作原理与实验\程序)

#### 4.17.13 实验效果图

独立看门狗程序写入实验板后，使用公司开发的多功能监视系统，在串口调试界面中的接收区就会接收到程序定时发送过来的数据。请大家注意：独立看门狗和外围设备有个交互，这种方法很重要，但要选择合适的应用场景，有些控制不适合使用看门狗方式，在做产品设计时请仔细考虑产品的安全性。

