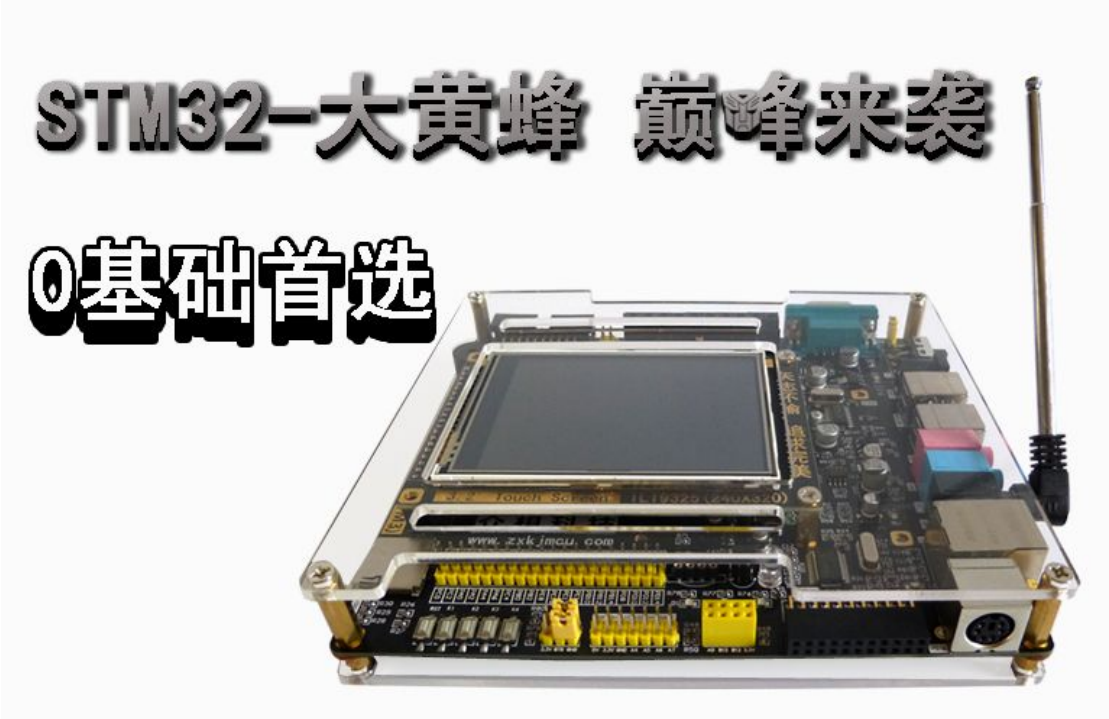
学 ARM 从 STM32 开始

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

4.17 STM32 独立看门狗的工作原理与实验

4.17.1 概述

4.17.1.1 看门狗概念

看门狗分硬件看门狗和软件看门狗。硬件看门狗是利用一个定时器电路，其定时输出连接到电路的复位端，程序在一定时间范围内对定时器清零(俗称“喂狗”)，因此程序正常工作时，定时器总不能溢出，也就不能产生复位信号。如果程序出现故障，不在定时周期内复位看门狗，就使得看门狗定时器溢出产生复位信号并重启系统。软件看门狗在这里先不介绍。

4.7.1.2 看门狗设计原理

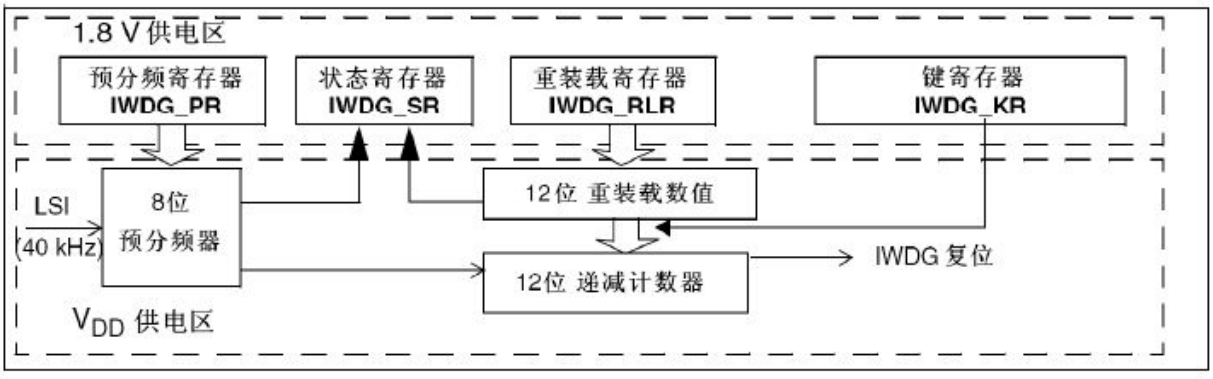
在看门狗（看门狗,又叫 watchdog timer,是一个定时器电路, 一般有一个输入,叫喂狗(kicking the dog or service the dog),一个输出到 MCU 的 RST 端,MCU 正常工作的时候,每隔一段时间输出一个信号到喂狗端,给 WDT 清零,如果超过规定的时间不喂狗,(一般在程序跑飞时),WDT 定时超过,就会给出一个复位信号到 MCU,使 MCU 复位. 防止 MCU 死机. 看门狗的作用就是防止程序发生死循环，或者说程序跑飞。 出于对单片机运行状态进行实时监测的考虑,产生了一种专门用于监测单片机程序运行状态的芯片,俗称"看门狗"(watchdog)）集成电路，该电路提供了响应的输入脉冲流损失锁存故障指示。

4.17.2 STM32 内置看门狗简介

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。



图一 独立看门狗框图

注释：看门狗功能处于 VDD 供电区，即在停机和待机模式时仍能正常工作。

表 1 看门狗超时时间（40kHz 的输入时钟（LSI））

预分频系数	PR[2:0]位	最短时间（ms） RL[11:0]=0x000	最长时间（ms） RL[11:0]=0xFFF
/4	0	0.1	409.6
/8	1	0.2	819.2
/16	2	0.4	1638.4
/32	3	0.8	3276.8
/64	4	1.6	6553.6
/128	5	3.2	13107.2
/256	6 或 7	6.4	26214.4

键寄存器(IWDG_KR) 地址偏移：0x00 复位值：0x0000 0000（在待机模

式复位）

地址偏移量：0x00

复位值：0x0000 0000(在待机模式复位)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

表 2 位详细说明

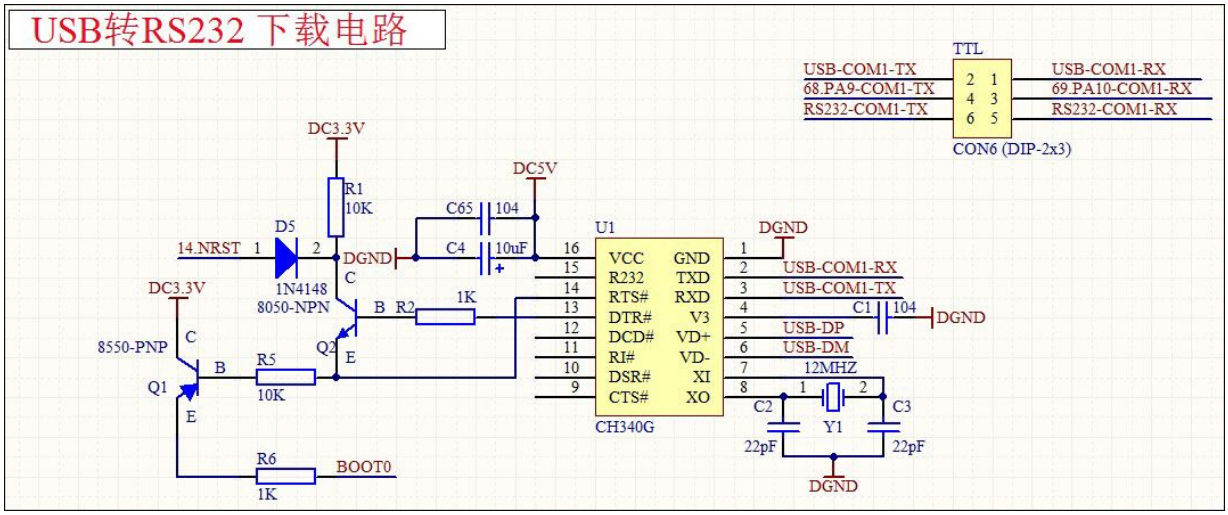
位 31:16	保留，始终读为 0。
位 15:0	KEY[15:0]：键值（只写寄存器，读出值为 0x0000）（Key value） 软件必须以一定的间隔写入 0xAAAA,否则，当计数器为 0 时，看门狗会产生复位。 写 0x55555 入表示允许访问 IWDG_PR 和 IWDG_RLR 寄存器。 写入 0xCCCC，启动看门狗工作（若选择了硬件看门狗则不受此命令字限制）

4.17.3 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。

4.17.4 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。

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，tm32f10x_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 "sttdio.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)//独立看门狗子函数
   {
       IWDG_WriteAccessCmd(IWDG_WriteAccess_Enable);//使能寄存器 写功能
       IWDG_SetPrescaler(IWDG_Prescaler_64);//设置预分频 40K/64=0.625k 一个
周期是 1.6ms

       IWDG_SetReload(800); //800*1.6ms=1.28S //设置初值
       IWDG_ReloadCounter();//喂狗
       IWDG_Enable();//使能独立看门狗
   }

   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_InitStructure.USART_BaudRate=9600;
    USART_InitStructure.USART_WordLength=USART_WordLength_8b;
    USART_InitStructure.USART_StopBits=USART_StopBits_1;
    USART_InitStructure.USART_Parity=USART_Parity_No;
    USART_InitStructure.USART_HardwareFlowControl=USART_HardwareFlowContr
ol_None;
    USART_InitStructure.USART_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，请大家注意在这段看门狗程序中，看门狗复位时间是 1.28 秒，也就是说在 1.28 秒时间内“喂狗”，独立看门狗就不会产生复位信号。具体的时间计算请参考主程序内的注释。

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 函数 IWDG_SetPrescaler

表 4 描述了函数 IWDG_SetPrescaler

函数名称	IWDG_SetPrescaler
函数原型	void IWDG_SetPrescaler(u8 IWDG_Prescaler)
功能描述	设置 IWDG 预分频值
输入参数	IWDG_Prescaler：IWDG 预分频值 参阅 Section：IWDG_Prescaler 查阅更多该参数允许取值范围
输出参数	无
返回值	无
先决条件	无
被调用函数	无

该参数设置 IWDG 预分频值（见附表 4）

附表 4 IWDG_Prescaler 值

IWDG_Prescaler	描述
IWDG_Prescaler_4	设置 IWDG 预分频值为 4
IWDG_Prescaler_8	设置 IWDG 预分频值为 8
IWDG_Prescaler_16	设置 IWDG 预分频值为 16
IWDG_Prescaler_32	设置 IWDG 预分频值为 32
IWDG_Prescaler_64	设置 IWDG 预分频值为 64
IWDG_Prescaler_128	设置 IWDG 预分频值为 128
IWDG_Prescaler_256	设置 IWDG 预分频值为 256

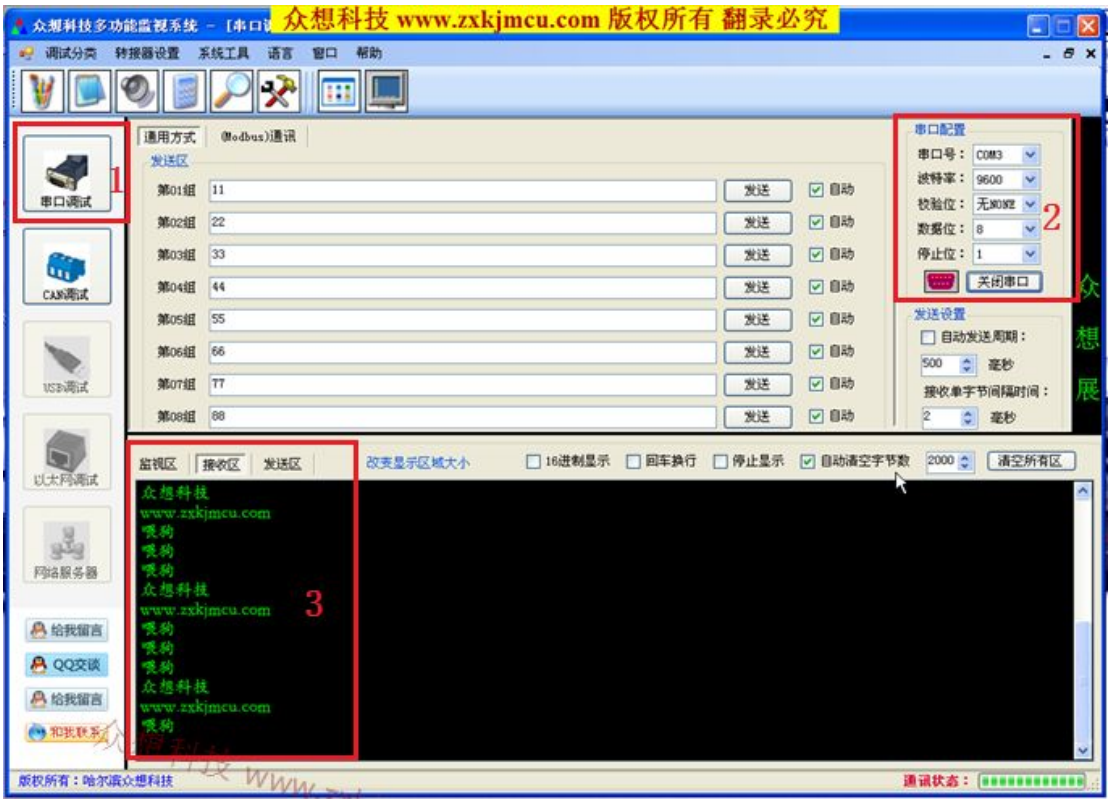
4.17.13 程序下载

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\基础篇 30. STM32 独立看门狗工作原理与实验\程序）

4.17.14 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合，有些控制上不适合使用看门狗复位方式。所以在做产品设计时请仔细考虑产品的安全性。


众想科技 www.zxkjmcu.com 版权所有 翻录必究
众想科技多功能监视系统
调试分类
转接器设置
系统工具
语言
窗口
帮助
串口调试
CAN调试
USB调试
以太网调试
网络服务器
给我留言
QQ交谈
给我留言
和我联系
通用方式
(Modbus)通讯
发送区
第01组 11
第02组 22
第03组 33
第04组 44
第05组 55
第06组 66
第07组 77
第08组 88
发送
自动
串口配置
串口号：COM3
波特率：9600
校验位：无(NONE)
数据位：8
停止位：1
关闭串口
发送设置
自动发送周期：
500 毫秒
接收单字节间隔时间：
2 毫秒
众想展
监视区
接收区
发送区
改变显示区域大小
16进制显示
回车换行
停止显示
自动清空字节数
2000
清空所有区
众想科技
www.zxkjmcu.com
喂狗
1
2
3
版权所有：哈尔滨众想科技
通讯状态：