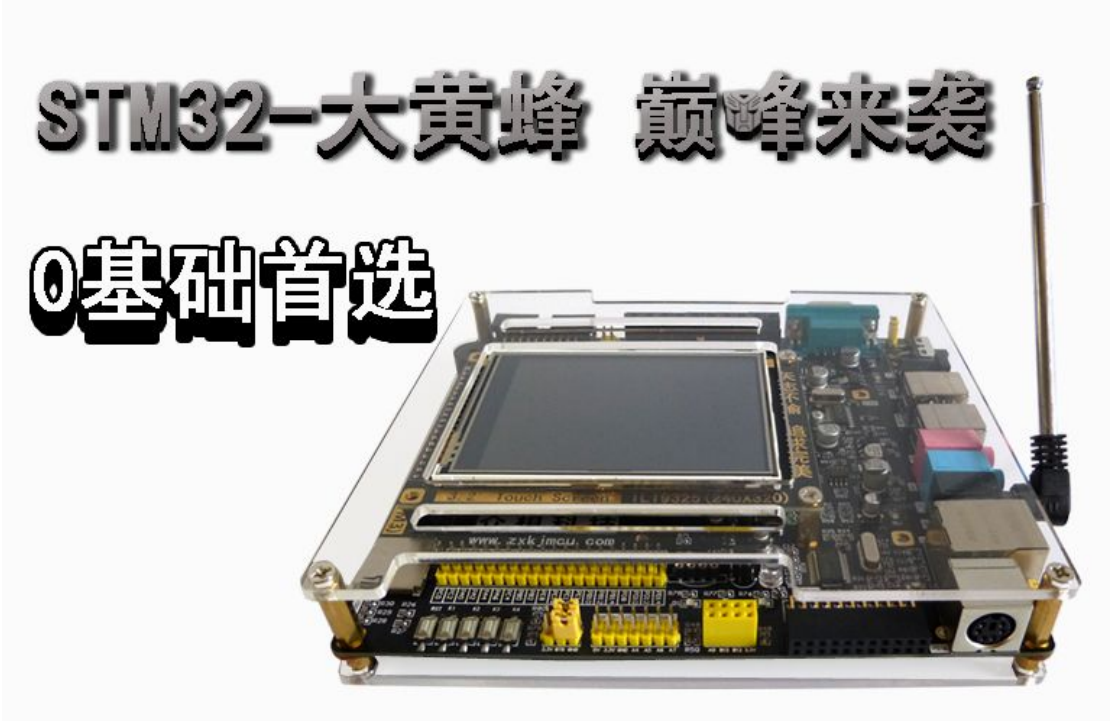
学 ARM 从 STM32 开始

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

4.9 STM32 printf 重定向 实验

4.9.1 概述

4.9.1.1 prinif 重定向概念

prinif 重定向，对于大多数初学者都很陌生。如果你是从汇编、51 单片机入手开始学习，就没有接触过 prinif 重定向，因为我们在这个过程中从来没用过这个功能；如果你是软件专业的，学过 QB7、学过安卓、学过 C++，那么你们就对这个 prinif 就很熟悉了。简单通俗的讲 prinif 重定向就是把数据打印到屏幕上。

4.7.1.2 prinif 重定向原理

简单讲 prinif 重定向原理就是通过串口输出，把你想要打印的数据打印到串口精灵上，也就是显示到显示器上了。

4.7.1.3 USART 库函数说明

表 2 USART 库函数

序号	USART 库函数
1	USART_ReceiveData
2	USART_SendBreak
3	USART_SetPrescaler
4	USART_CardCmd
5	USART_CardNackCmd
6	USART_HalfDuplexCmd
7	USART_IrDAConfig
8	USART_IrDACmd
9	USART_GetFlagStatus
10	USART_ClearFlag
11	USART_GetITStatus
12	USART_ClearITPendingBit

以上库函数要求大家要了解，知道在什么地方能方便的找到就可以，不需要

背下来，最主要的是做到活学活用。

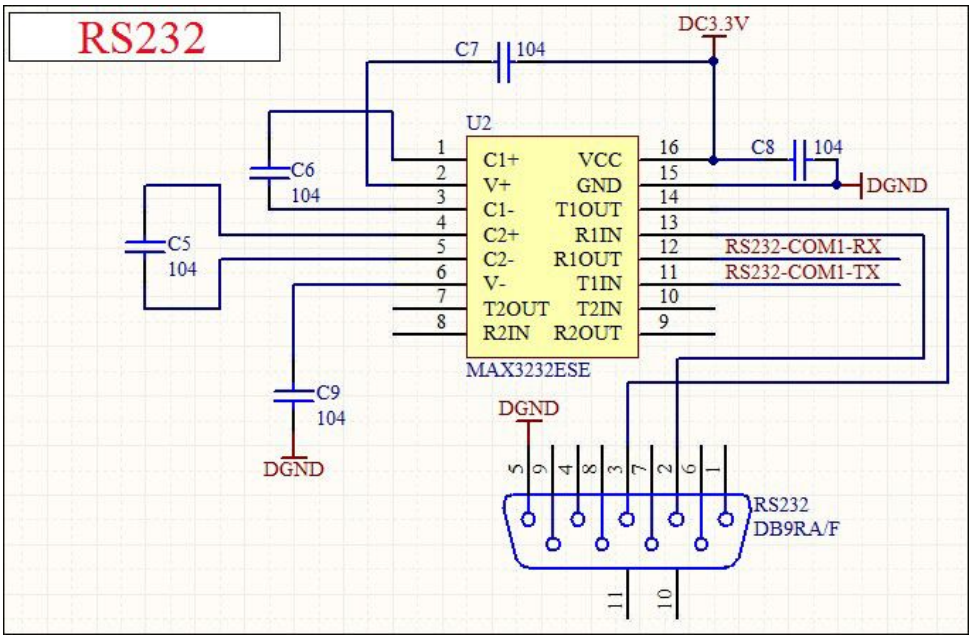
4.9.2 实验目的

1、下传编写好的 prinif 重定向程序，通过串口把需要显示的内容打印到串口精灵上，也就是在显示器上显示出来了。

2、prinif 重定向在以后的试验中要经常使用，比如调试触摸屏，AD/DA 及做实验时要检验数据的规律，在这里我们先要掌握一个方便的显示方法，为以后的学习和工作打下一个基础。

4.9.3 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。



图一 串口通讯应用电路

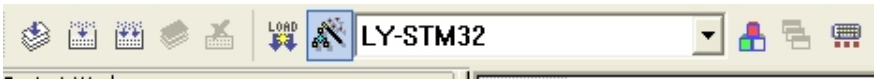
4.9.4 软件设计

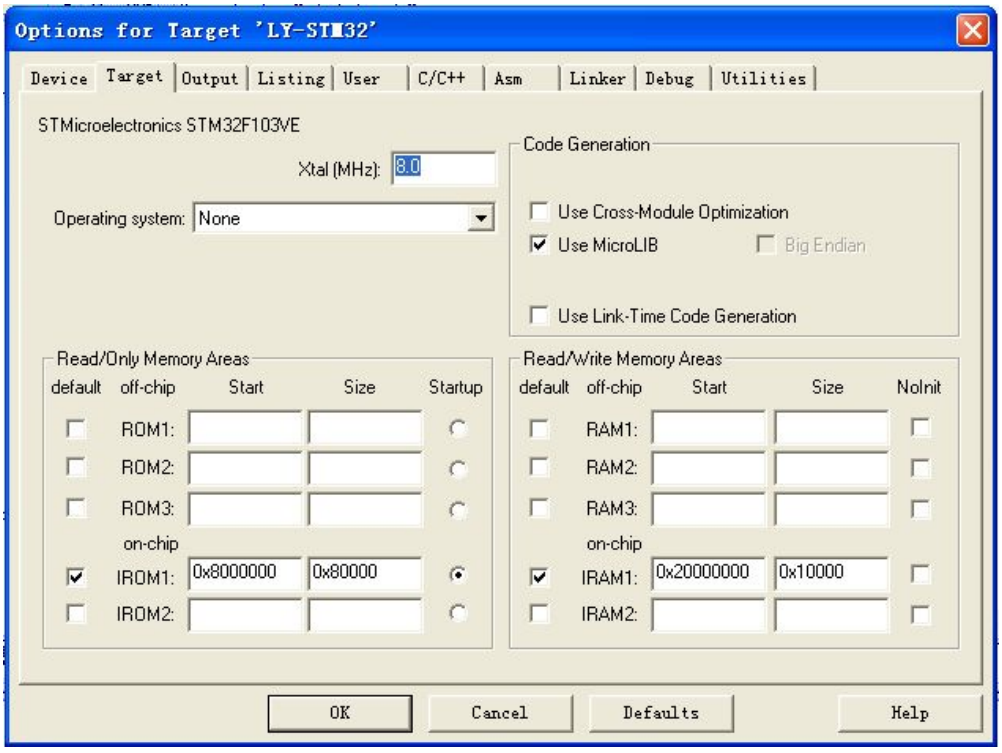
软件设计和 RS232 很相似，只有那么 3 点不同，下面对软件的设计做一次说明：

1、引入头文件“stdio.h”，加入到自己定义的公共头文件中；

2、重新改变一个函数“int fputc(int ch,FILE *f”，如果有数据需要输出打印的时候，通过这个函数把数据送到串口上；

3、进入“options”功能，打开“target”选项卡，把“Use MicroLIB”勾选上。





4.9.4.1 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_usart.c  // 串口通讯函数
```

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_usart.c 库函数主要包含串行通讯设置，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。

4.9.4.2 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。

4.9.4.3 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_usart.h"
#include "sttdio.h"

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。

4.9.4.4 pbdata.c 文件里的内容是

```
#include "pbdata.h" //很重要，引用这个头文件

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
}
```

4.9.5 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。

4.9.6 GPIO 引脚时钟使能

```
    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。

4.9.7 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)
   {
      //printf("众想科技\r\n");
     //printf("www.zxkjmcu.com\r\n");//回车换行

     //d1=123456;//给临时变量 d1 付值
     //printf("10 进制整型输出 d1=%10d\r\n",d1);

     //d1=123456;
     //printf("16 进制输出 d1=%X\r\n",d1);//X 大写输出  x 小写输出

     //d2=12.345;
     //printf("浮点数输出 d2=%f\r\n",d2);
     ///printf("浮点数输出 d2=%.2f\r\n",d2);
     //printf("浮点数输出 d2=%10.2f\r\n",d2);

     //d3='a';
     //printf("字符输出 d3=%c\r\n",d3);

     d4[0]='a';
     d4[1]='b';
     d4[2]='c';
     d4[3]='d';
     d4[4]='\0';

     printf("字符串输出 d4=%s\r\n",d4);

     //注意 delay_ms 函数输入范围是 1-1863
     //所以最大延时为 1.8 秒
     delay_ms(1000);
     delay_ms(1000);
     delay_ms(1000);
   }
}
```

在程序中有“\r\n”，它的意思是回车换行，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。

4.9.8 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。还有就是中断处理子函数都包含在一个“stm32f10x_it.h”文件中，初学者如果不明白各种中断处理子函数的具体书写方式，可以进入这个头文件中查找，找到后复制/粘贴到中断处理“stm32f10x_it.c”文件中使用。

```
#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);
   }
}
```

4.9.9 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有一些处理过程，现在粘贴到下面，方便大家阅读。

```
#include "pbdata.h"

void RCC_Configuration(void);
void GPIO_Configuration(void);
void NVIC_Configuration(void);
void USART_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)
{
   u32 d1=0;
   float d2=0;
   u8   d3=0;
   u8   d4[5];

   RCC_Configuration();  //系统时钟初始化
   GPIO_Configuration();//端口初始化
   USART_Configuration();
   NVIC_Configuration();

   while(1)
   {
      //printf("众想科技\r\n");
      //printf("www.zxkjmcu.com\r\n");

      //d1=123456;
      //printf("10 进制整型输出 d1=%10d\r\n",d1);

      //d1=123456;
      //printf("16 进制输出 d1=%X\r\n",d1);//X 大写输出  x 小写输出

      //d2=12.345;
      //printf("浮点数输出 d2=%f\r\n",d2);
      ///printf("浮点数输出 d2=%.2f\r\n",d2);
      //printf("浮点数输出 d2=%10.2f\r\n",d2);

      //d3='a';
      //printf("字符输出 d3=%c\r\n",d3);


      d4[0]='a';
      d4[1]='b';
      d4[2]='c';
      d4[3]='d';
      d4[4]='\0';

      printf("字符串输出 d4=%s\r\n",d4);
```

```
        //注意 delay_ms 函数输入范围是 1-1863
        //所以最大延时为 1.8 秒
        delay_ms(1000);
        delay_ms(1000);
        delay_ms(1000);
    }
}

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，就是注意“int fputc(int ch,FILE *f)” 的书写方式和含义，要做到熟练使用。

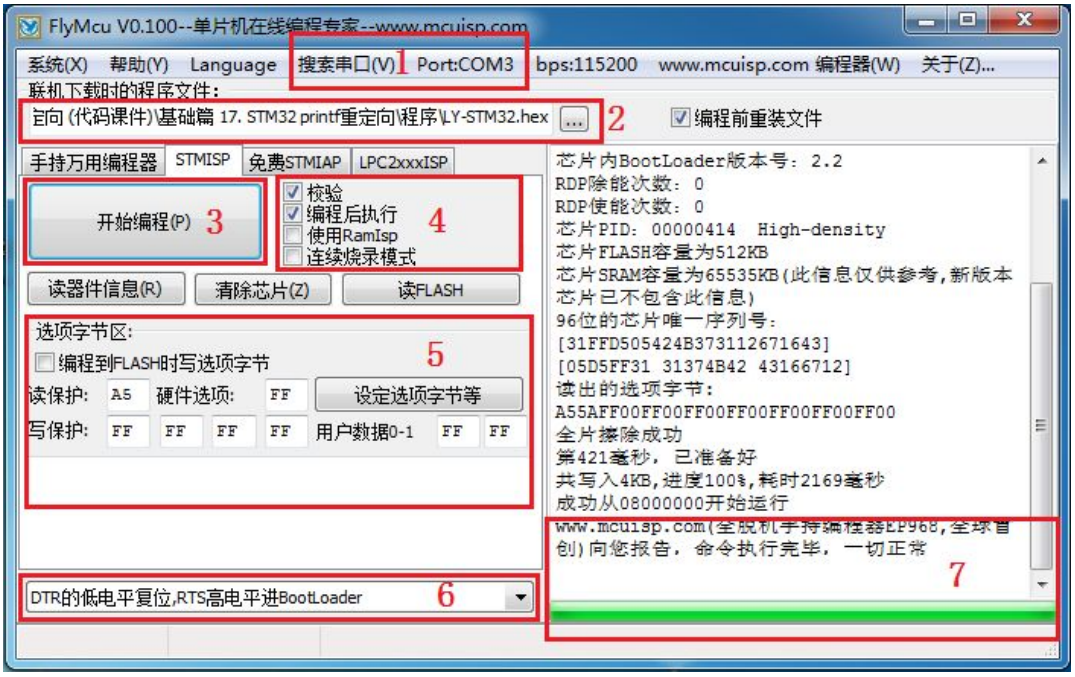
附表 1　打印方式书写格式说明

序号	打印方式书写格式	打印方式说明
1	%d	按照十进制整数打印
2	%6d	按照十进制整数打印，至少 6 个字符宽度
3	%f	按照浮点数打印
4	%6f	按照浮点数打印，至少 6 个字符宽度
5	%.2f	按照浮点数打印，小数点后 2 位有效数字
6	%6.2f	按照浮点数打印，至少 6 个字符宽度，小数点后 2 位有效数字
7	%x	按照十六进制打印
8	%c	打印字符
9	%s	打印字符串

4.9.10 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 COM3。（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 17. STM32 printf 重定向\程序）

4.9.11 实验效果图

printf 重定向程序写入实验板后，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。和串口通讯涉程序相差无几，串口程序掌握了这个就很快掌握。具体实验效果参考下图。


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
给我留言
给我留言
和我联系
通用方式
(Modbus)通讯
发送区
第01组 11 发送 自动
第02组 22 发送 自动
第03组 33 发送 自动
第04组 44 发送 自动
第05组 55 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：
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
www.zxkjmcu.com
10进制整型输出 d1=123456
众想科技
版权所有：哈尔滨众想科技
通讯状态：