# 匿名无线串口使用教程

撰写人 广西大学机器人创新实验基地 ROBOT-Z实验室

电控组电科193覃春阳

# 前言

“匿名无线串口”有什么用？

它是一个方便的无线参数调试工具。

能够解决需要实时传输数据，实时远程观测调参具体效果，查看效果波形的需求，不需要整定一次参数，就重新编译、下载程序，然后再观测效果。

如：无线实时调试移动小车速度PID，实时更改发射机构的参数（力矩，速度，电流）并查看效果。

价格：120￥/个

淘宝连接：[https://item.taobao.com/item.htm?spm=a230r.1.14.16.8093628fK5OcyV&id=624766005093&ns=1&abbucket=20#detail](https://item.taobao.com/item.htm?spm=a230r.1.14.16.8093628fK5OcyV&id=624766005093&ns=1&abbucket=20" \l "detail)



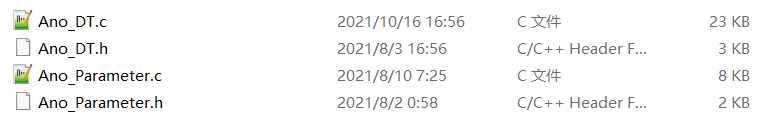
实例程序使用stm32f407的串口2进行测试

UART2 : PA2 PA3 ------>用于与上位机通信---->波特率使用：500000 TX RX

# 移植“无线串口”代码

## 移植文件

在代码工程中加入，这四个文件——“无线串口”驱动文件。



下面的值板级支持文件（BSP），根据单片机的情况进行设计。这里用作讲解。



## 移植“无线串口”需要的改动。

需要设置：

串口初始化；

串口发送、接收中断；

自定义发送数据函数设置

### 串口初始化

波特率：500000，其他参数参看配套文件。这里不做讲解；

void Usart2\_init(u32 bound)

{

//GPIO端口设置

GPIO\_InitTypeDef GPIO\_InitStructure;

USART\_InitTypeDef USART\_InitStructure;

USART\_ClockInitTypeDef USART\_ClockInitStruct;

NVIC\_InitTypeDef NVIC\_InitStructure;

RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOA,ENABLE); //使能GPIOA时钟

RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_USART2,ENABLE);//使能USART2时钟

//串口2对应引脚复用映射

GPIO\_PinAFConfig(GPIOA,GPIO\_PinSource2,GPIO\_AF\_USART2); //GPIOA2复用为USART2

GPIO\_PinAFConfig(GPIOA,GPIO\_PinSource3,GPIO\_AF\_USART2); //GPIOA3复用为USART2

//USART2端口配置

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_2 ; //GPIOA2 TX

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_AF;//复用功能

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz; //速度50MHz

GPIO\_InitStructure.GPIO\_OType = GPIO\_OType\_PP; //推挽复用输出

GPIO\_InitStructure.GPIO\_PuPd = GPIO\_PuPd\_UP; //上拉

GPIO\_Init(GPIOA,&GPIO\_InitStructure); //初始化PA3

/////////////

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_3; //GPIOA3 RX

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_AF;//复用功能

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz; //速度50MHz

GPIO\_InitStructure.GPIO\_OType = GPIO\_OType\_OD; //开漏

GPIO\_InitStructure.GPIO\_PuPd = GPIO\_PuPd\_NOPULL;

GPIO\_Init(GPIOA,&GPIO\_InitStructure); //初始化PA3

//USART2 初始化设置

USART\_InitStructure.USART\_BaudRate = bound;//波特率设置

USART\_InitStructure.USART\_WordLength = USART\_WordLength\_8b;//字长为8位数据格式

USART\_InitStructure.USART\_StopBits = USART\_StopBits\_1;//一个停止位

USART\_InitStructure.USART\_Parity = USART\_Parity\_No;//无奇偶校验位

USART\_InitStructure.USART\_HardwareFlowControl = USART\_HardwareFlowControl\_None;//无硬件数据流控制

USART\_InitStructure.USART\_Mode = USART\_Mode\_Tx | USART\_Mode\_Rx; //收发模式

//配置USART2时钟

USART\_ClockInitStruct.USART\_Clock = USART\_Clock\_Disable; //时钟低电平活动

USART\_ClockInitStruct.USART\_CPOL = USART\_CPOL\_Low; //SLCK引脚上时钟输出的极性->低电平

USART\_ClockInitStruct.USART\_CPHA = USART\_CPHA\_2Edge; //时钟第二个边沿进行数据捕获

USART\_ClockInitStruct.USART\_LastBit = USART\_LastBit\_Disable; //最后一位数据的时钟脉冲不从SCLK输出

//Usart2 NVIC 配置

NVIC\_InitStructure.NVIC\_IRQChannel = USART2\_IRQn;//串口2中断通道

NVIC\_InitStructure.NVIC\_IRQChannelPreemptionPriority=NVIC\_UART2\_P;//抢占优先级2

NVIC\_InitStructure.NVIC\_IRQChannelSubPriority =NVIC\_UART2\_S; //子优先级0

NVIC\_InitStructure.NVIC\_IRQChannelCmd = ENABLE; //IRQ通道使能

NVIC\_Init(&NVIC\_InitStructure); //根据指定的参数初始化VIC寄存器

////////////////////////////////////////////////////////

USART\_Init(USART2, &USART\_InitStructure); //初始化串口2

USART\_ClockInit ( USART2, &USART\_ClockInitStruct );

////////////////////////////////////////////////////////

USART\_ITConfig(USART2, USART\_IT\_RXNE, ENABLE);//开启接收相关中断

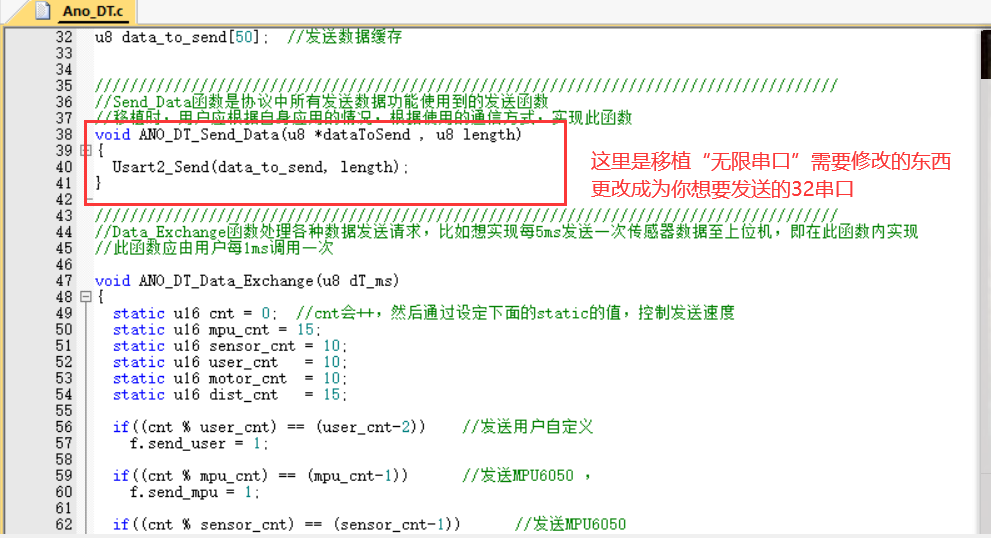
USART\_Cmd(USART2, ENABLE); //使能串口2

}

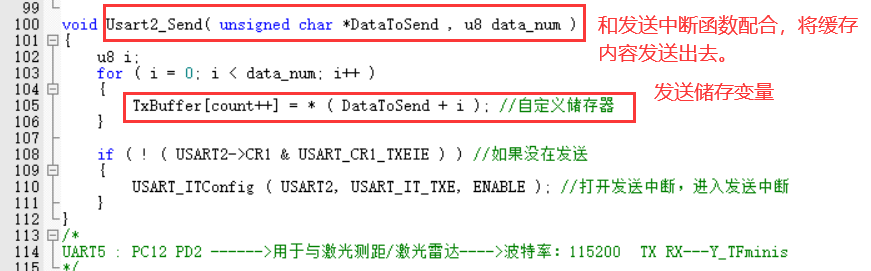
### 串口发送中断设置

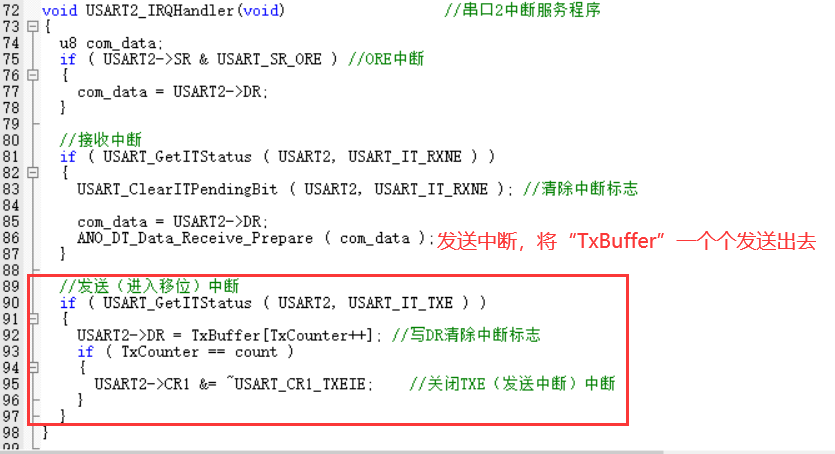
如图代码所示，文字加以说明。里面的函数名称如无说明，必须一致，不能更改。

Ano\_DT.C：



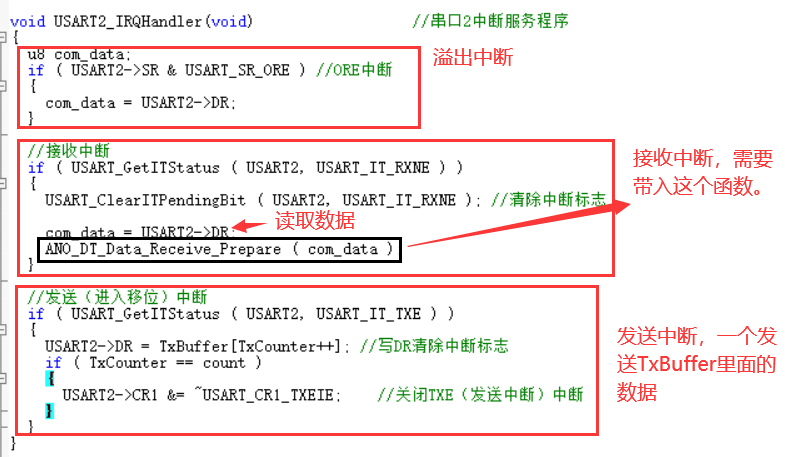
MY\_usart.c





### 串口接收中断设置

对于ORE溢出中断按照着下图处理就可以；接收中断如图讲解。

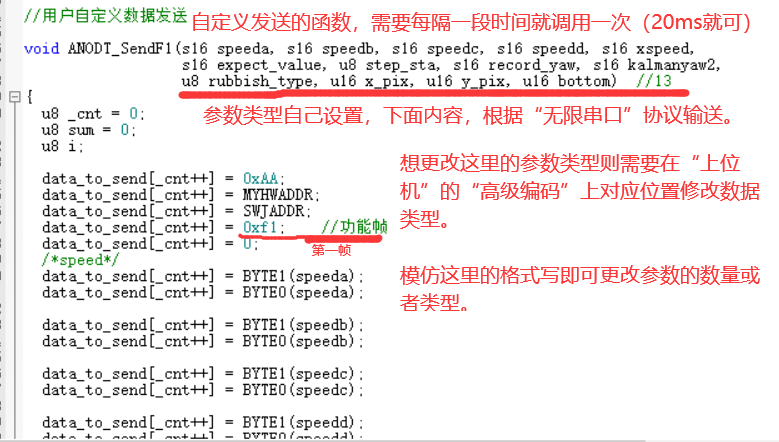


再下一层到ANO\_DT\_Data\_Receive\_Anl\_Task()函数不需要深究，是属于“无线串口”的内部驱动文件。

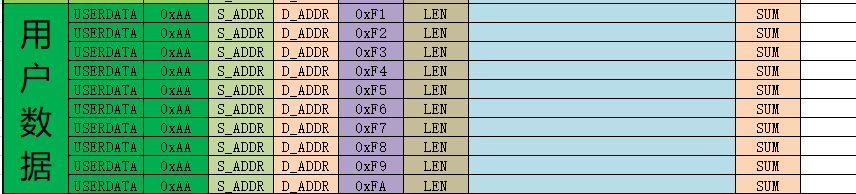
### 自定义发送变量ANODT\_SendF1（）设置

根据**协议**，先设置参数类型，数量，再根据协议进行函数编写。

用以在上位机上查看的自定义数据。

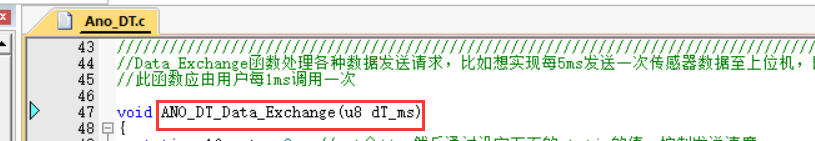


协议：其他具体参看“ANO\_XIEYI.xlsx”文件



### ANO\_DT\_Data\_Exchange(u8 dT\_ms)函数说明

是用户需要每隔1ms时间调用的函数：//1ms时间通过定时器，或者系统RTOS进行设定。



ANO\_DT\_Data\_Exchange(u8 dT\_ms)函数引用有ANODT\_SendF1()函数。

## 3.相关变量说明

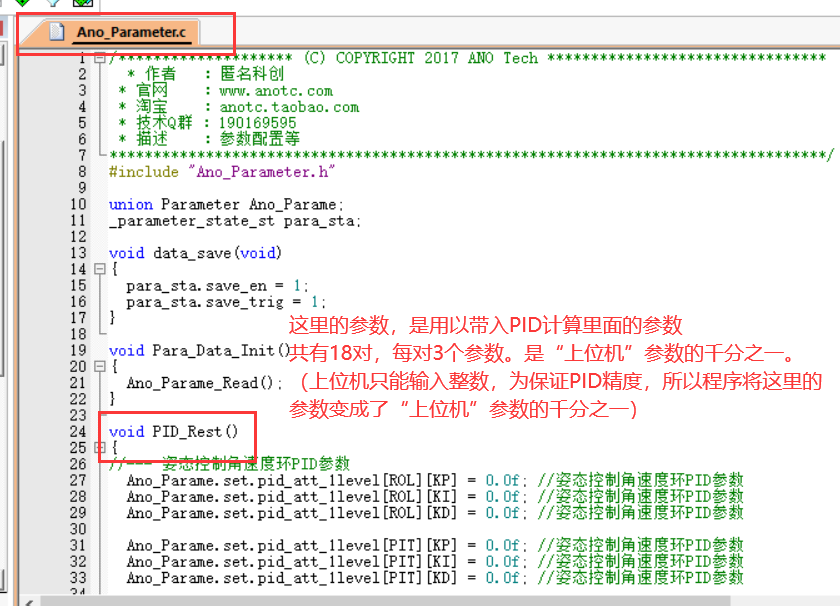
分为**“32中PID输入参数”**以及和**“上位机显示的参数”**。两者只是相差1000、100倍，其作用顾名思义。

### “32中PID输入参数”

32中PID输入参数，如：Ano\_Parame.set.pid\_att\_1level[ROL][KP]

（在PID\_Rest()之中，可以进行数值初始化。）

这类变量，是直接带入32的PID参数之中的。



### “上位机显示的参数”

上位机上显示的数据变量，如ParValList[PAR\_PID\_1\_P]

### 两者转换关系

ANO\_DT\_ParListToParUsed(void)和ANO\_DT\_ParUsedToParList(void)函数对应转换，有的1000:1，有的100:1，具体参看文件Ano\_Parameter.c。

# 上位机使用说明

完成上面软件的移植之后，进行如下操作。

## 硬件连接

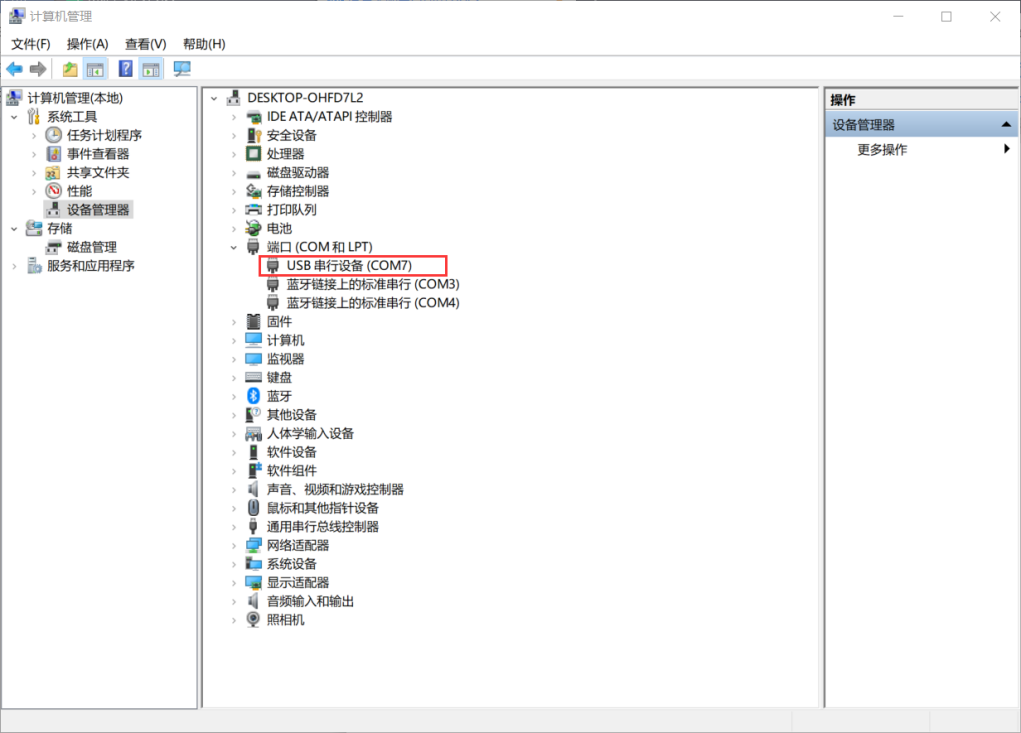
准备两个“匿名无线串口”，一台电脑，一个stm32f407板,两条micro usb数据线——用来供电

### 电脑端

一个“匿名无线串口”通过micro usb线连接电脑。



需要查看电脑“设备管理器”中的“端口”是否有显示接入，因为有的数据线只能充电，无法进行数据传输。如果发现这个情况，只能换一根数据线（Micro  USB公头）这点要检查好！



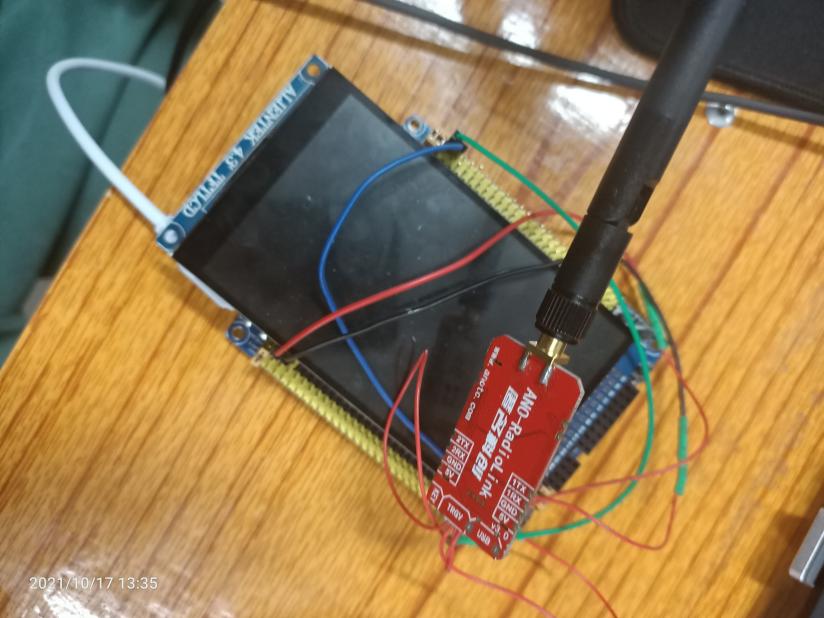
### STM32端

另一个“匿名无线串口”，接上32软件设置的串口引脚，由于“匿名无线串口”的数据线不是杜邦线，所以需要间断焊上杜邦线母头，才可以接入32。

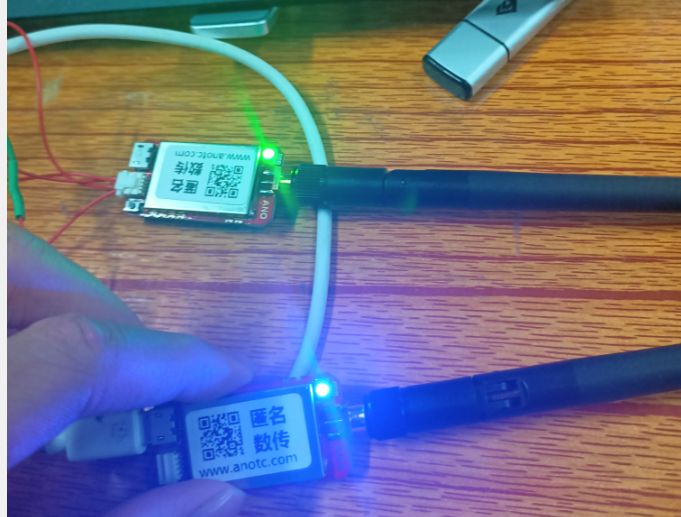
这里我使用的是f407的串口2，所以硬件连接如下：

UART2 : PA2 PA3 ------>用于与上位机通信---->波特率使用：500000 TX RX

然后开发板记得供电，并且开启。



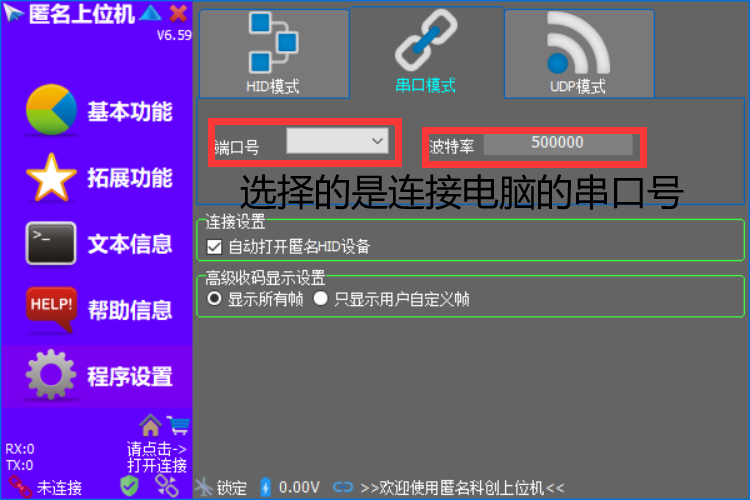
③连接成功，蓝绿灯常亮。（闪烁说明连接不良）



## “数据波形”显示设置步骤

### 设定端口号、波特率

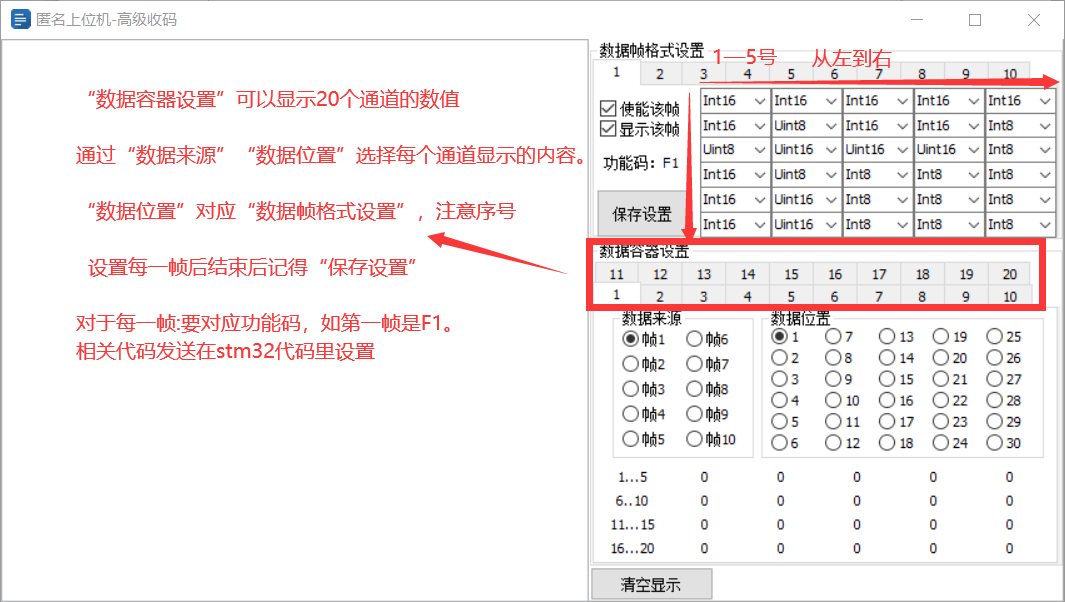
选择的是连接电脑的“无线串口”的端口。波特率：500000.



### 设置“高级收码”



**具体步骤**



First进入第1帧，然后设置好数据的类型；

Second设置好“数据容器设置”的“数据来源”、”数据位置“，

这里由于我们使用的是第1帧来传入13个参数，所以

|  |  |  |
| --- | --- | --- |
| 数据容器位置 | 数据来源 | 数据位置——接收顺序 |
| 1 | 1 | 1 |
| 2 | 1 | 2 |
| 3 | 1 | 3 |

Third上位机串口接收顺序说明：

第一排数据是1-5

第二排数据是6-10，以此类推，对应这32软件的ANODT\_SendF1（）函数的参数， 可以看到这里每一帧一共可以带入30个参数，而用不到的参数不必设置。这里只用13个参数，所以14号之后的参数类型就不用管。

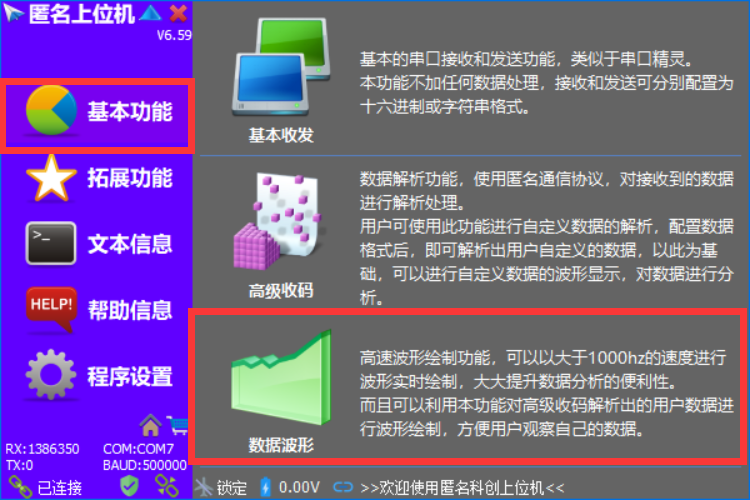
## 3.查看“数据波形”

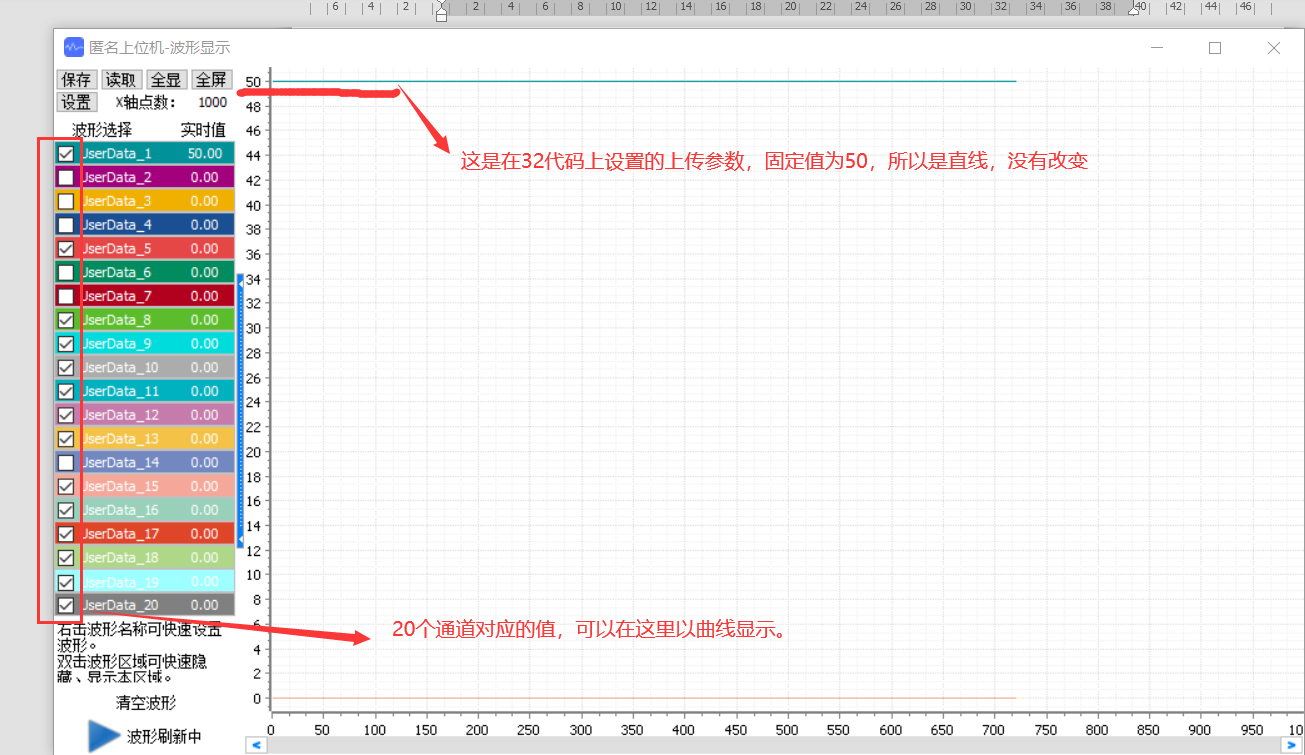
获取变量的值，查看调整效果。

### 连接串口。



### 进入“数据波形”





本地例子：32软件上ANODT\_SendF1（）第一个变量的值设置为常量50，所以显示固定数值50，这里没有任何变化；

调试PID的时候，可以设置PID的参数作为传入的值进行查看。

# PID参数调定方法

PID参数调定方法：向32输入PID值进行调参。

在32代码上的文件都移植完毕，上位机设置好了之后，进行如下操作。

### 进入“飞控设置”



### “读取飞控”

如下图，先“读取飞控”，这里读取的值，就等于32中“上位机显示的参数”，如ParValList[PAR\_PID\_1\_P]。



### “写入飞控”

读取成功之后，就可以“写入飞控”，

即在18对PID中输入参数，然后“写入参数”

然后“无线串口”将数据发送给32，在32中把“上位机显示的参数”改变，与此同时，调用ANO\_DT\_ParListToParUsed(void)，把“32中PID输入参数”缩小1000或者100倍。

而我们要用PID中的参数，就是用“32中PID输入参数”作为输入，从而实现PID参数整定。

具体调整效果显示，如调整电机速度pid参数，则可以在ANODT\_SendF1（）中，加入实际速度的变量，然后通过“数据波形”，就可以看到波形效果了。

# 数据保存

最后的变量数据，其实需要在keil上更改烧录后才能真正留在程序之中，在PID参数整定中，参数数值只是临时保存的。