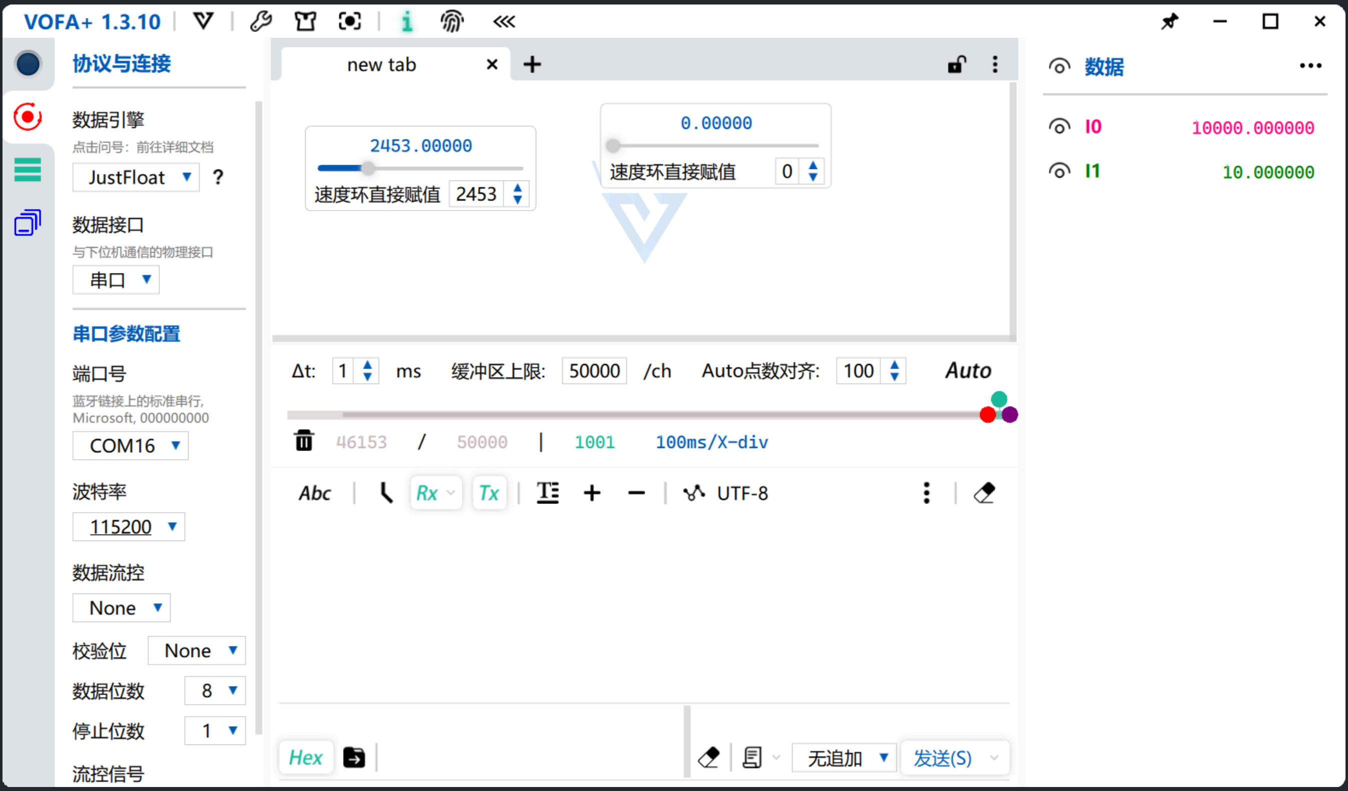
使用方法

主要是用于无线串口调参，使用`vofa`软件的`Justfloat`协议以及命令帧解析，可自定义响应帧。



VOFA+ 1.3.10

协议与连接

数据引擎
点击问号：前往详细文档
JustFloat ?

数据接口
与下位机通信的物理接口
串口

串口参数配置

端口号
蓝牙链接上的标准串行, Microsoft, 000000000
COM16

波特率 115200

数据流控 None

校验位 None

数据位数 8

停止位数 1

流控信号

new tab

2453.00000
速度环直接赋值 2453

0.00000
速度环直接赋值 0

```
Δt: 1 ms    缓冲区上限: 50000 /ch    Auto点数对齐: 100    Auto
46153 / 50000 | 1001    100ms/X-div
```

Abc | Rx | Tx | UTF-8

Hex | 无追加 | 发送(S)

数据

I0	10000.000000
I1	10.000000

一、对外接口

- ./inc/vofa_functions

```c
void vofaSendJustFloat(vofaJustFloatFrame *vofaJFFrame);             //以JustFloat协议发送数据
void vofaSendFirewater(const float *fdata, const uint32_t ulSize);  //以Firewater协议发送数据
void vofaSendRawdata(uint8_t *pData, const uint32_t ulSize);        //以rawdata协议发送数据

void vofaJustFloatInit(void);                                         //Justfloat协议初始化
void uartCMDRecv(uint8_t byte_data);                                  //uart串口接收单字节并存入vofaCommandData数据包
void vofaCommandParse(void);                                          //解析命令

extern vofaJustFloatFrame JustFloat_Data;                             //包含接收到的浮点数据的结构体
extern vofaCommand vofaCommandData;                                   //包含命令的结构体

```

1.1 服务函数接口

以下接口用于三种协议的数据发送

```c
/**
* @param vofaJFFframe: 包含数据帧的结构体
* @return void
*/
void vofaSendJustFloat(vofaJustFloatFrame *vofaJFFrame)
{
    uint8_t i;
    uint8_t u8Array[4];
    for (i = 0; i < CH_COUNT; i++)
    {
        float2uint8Array(u8Array, vofaJFFrame->fdata[i], 0);
        uartSendData(vofaJFFrame->u8Array, sizeof(u8Array));
    }
    uartSendData(vofaJFFrame->frametail, FRAME_TAIL_SIZE);
}

/**
* @param fdata: 指向要发送的浮点数据的指针
* @param ulSize:  要发送的数据个数
* @return void
*/
void vofaSendFirewater(const float *fdata, const uint32_t ulSize)
{
    uint32_t i;
    for (i = 0; i < ulSize - 1; i++)
    {
        printf("%.6f,", *(fdata + i));
    }
    printf("%.6f\n", *(fdata + i));
}

/**
* @param pData: 指向要发送的单字节数据的指针
* @param ulSize:  要发送的数据个数
* @return void
*/
void vofaSendRawdata(uint8_t *pData, const uint32_t ulSize)
{
    uint32_t i;
    for (i = 0; i < ulSize; i++)
    {
        uartSendByte(*(Data + i));
    }
}
```

1.2 命令解析服务相关接口

以下接口用于接收到的命令帧处理

```c
/**
* @breif 初始化JustFloat帧结构体
*/
void vofaJustFloatInit(void)
{
    vofaCommandData.cmdID                   = INVALID;
    vofaCommandData.cmdType                 = INVALID;
    vofaCommandData.completionFlag          = 0;
    JustFloat_Data.frametail[0] = 0x00;
    JustFloat_Data.frametail[1] = 0x00;
    JustFloat_Data.frametail[2] = 0x80;
    JustFloat_Data.frametail[3] = 0x7f;
}

```

```c
/**
* @breif 将串口收到的数据判断并存入数据包中，并比对帧控制接收完成标志位置位
* @param byte_data:  串口接收到的字节数据
*/
void uartCMDRecv(uint8_t byte_data) //此函数放在串口中断中
{
    vofaCommandData.uartRxPacket[vofaRxBufferIndex] = byte_data;

    if (vofaCommandData.uartRxPacket[vofaRxBufferIndex - 1] == '!' && vofaCommandData.uartRxPacket[vofaRxBufferIndex] == '#')
    {
        vofaCommandData.completionFlag = 1;
        vofaRxBufferIndex  = 0;
    }

    else if (vofaRxBufferIndex > (CMD_FRAME_SIZE - 1))
    {
        vofaCommandData.completionFlag = 0;
        vofaRxBufferIndex  = 0;
        memset(vofaCommandData.uartRxPacket, 0, 10);
    }

    else
    {
        vofaRxBufferIndex++;
    }
}

/**
*
*/
void vofaCommandParse(void)
{
    uint8_t* pRxPacket;
    pRxPacket = vofaCommandData.uartRxPacket;

    if (vofaCommandData.uartRxPacket[0] != '@' || vofaCommandData.uartRxPacket[3] != '=' || vofaCommandData.uartRxPacket[CMD_FRAME_SIZE - 2] != '!' || vofaCommandData.
        uartRxPacket[CMD_FRAME_SIZE - 1] != '#')
    {
        memset(vofaCommandData.uartRxPacket, 0, CMD_FRAME_SIZE);
        return;
    }

    switch (vofaCommandData.uartRxPacket[1])
    {
        case 'S': vofaCommandData.cmdType = Speed;
            break;
        case 'P': vofaCommandData.cmdType = Position;
            break;
        default: vofaCommandData.cmdType = INVALID;
            break;
    }

    switch (vofaCommandData.uartRxPacket[2])
    {
        case '1': vofaCommandData.cmdID = Direct_Assignment;
            break;
        case '2': vofaCommandData.cmdID = Increase;
            break;
        case '3': vofaCommandData.cmdID = Decrease;
            break;
        default: vofaCommandData.cmdID = INVALID;
            break;
    }
    memcpy(vofaCommandData.validData, pRxPacket + 4, 4);

    vofaCommandData.floatData = 
uint8Array2Float(vofaCommandData.validData, 0);

    pRxPacket = NULL;
    memset(vofaCommandData.validData, 0, 4);
    memset(vofaCommandData.uartRxPacket, 0, CMD_FRAME_SIZE);
}
```

二、移植接口

- ./src/vofa_uart.c

  在移植时需要将以下串口发送单字节的函数中替换为相应的串口发送单字节的实现

```c
void uartSendByte(const uint8_t c)
{
    HAL_UART_Transmit(&huart1, (uint8_t*)&c, 1, HAL_MAX_DELAY); //修改为串口发送接口
}
```

- ./inc/vofa_function.h

```c
void uartCMDRecv(uint8_t byte_data);        //将此函数放在串口中断中
```

三、简单的使用样例

3.1 串口中断加入接收函数

```c
#include "vofa_function.h"

uint8_t rx_data = 0;

void USART1_RxCallBack(UsartTypedef *usart){
    if (usart->Instance == huart1){
        uartCMDRecv(rx_data);
        HAL_UART_Recieve_IT(&huart1, &rx_data, 1);
    }
}
```

3.2 定时判断接收标志位

```c
#include "vofa_function.h"

/*以主函数中的循环加延时检测为例*/
void main(void){
    /*
    *   一些初始化代码
    */
    for(;;){
        if (vofaCommandData.completionFlag == 1)    //受到命令帧
        {
            vofaCommandData.completionFlag = 0;     //清楚标志位
            vofaCommandParse();                     //解析收到的数据帧
            /*
            * 根据你的命令做相应处理
            * vofaCommandData.cmdID是接收到的命令ID
            * vofaCommandData.cmdType是接收到的命令类型
            * vofaCommandData.floatData是接收到的浮点数据
            */

            //使用举例
            switch(vofaCommandData.cmdType)         //判断收到的命令类型
            {
                case Speed:
                    switch(vofaCommandData.cmdID)   //判断收到的命令ID
                    {
                        case Direct_Assignment:
                            printf("I recv Command Type Speed, ID Direct_Assignment, data: %.6f", vofaCommandData.floatData);
                            break;
                        case Increse:
                            printf("I recv Command Type Speed, ID Increse, data: %.6f", vofaCommandData.floatData);
                            break;
                        case Decrease:
                            printf("I recv Command Type Speed, ID Decrease, data: %.6f", vofaCommandData.floatData);
                            break;
                    }
                    break;
                case Postion:
                    switch(vofaCommandData.cmdID)   //判断收到的命令ID
                    {
                        case Direct_Assignment:
                            printf("I recv Command Type Postion, ID Direct_Assignment, data: %.6f", vofaCommandData.floatData);
                            break;
                        case Increse:
                            printf("I recv Command Type Postion, ID Increse, data: %.6f", vofaCommandData.floatData);
                            break;
                        case Decrease:
                            printf("I recv Command Type Postion, ID Decrease, data: %.6f", vofaCommandData.floatData);
                            break;
                    }
                    break;
            }
        }
        osDelay(50);        //延时50ms
    }
}

```

四、如何定制自己的命令帧？

- ./inc/vofa_function.h

```c
#define CMD_FRAME_SIZE 10              //将此处更改为你需要的命令帧长度
```

修改自己的命令类型和ID

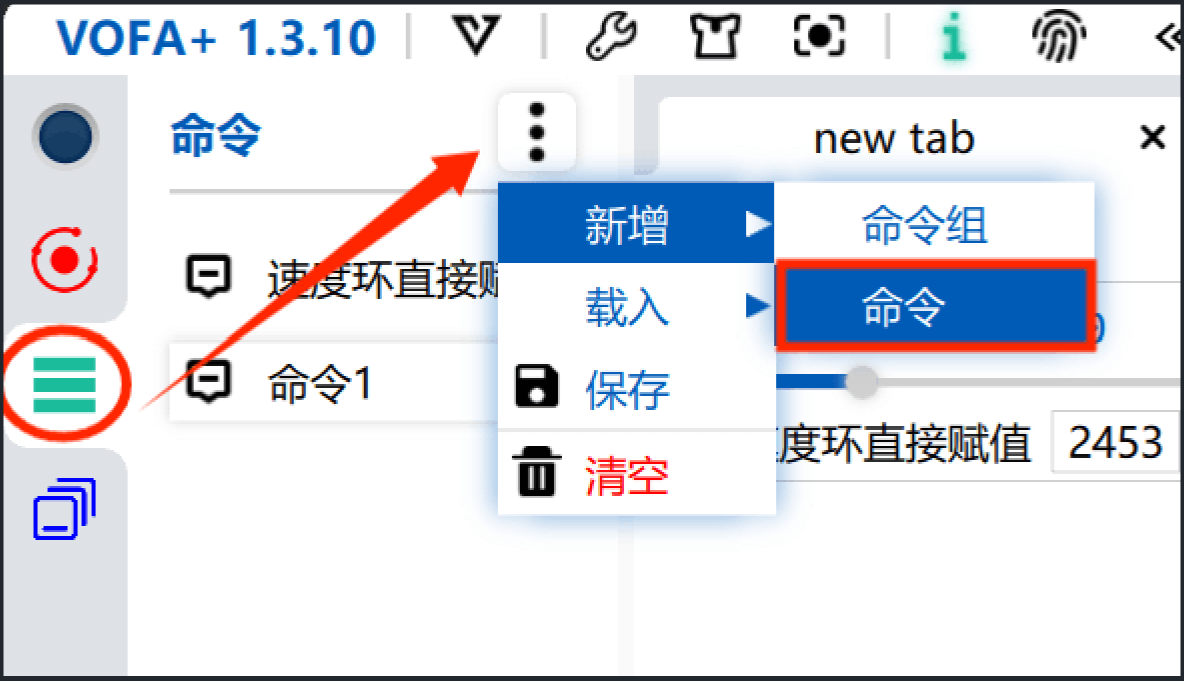
```c
//在此处枚举你需要的命令ID
enum CommandID
{
    Direct_Assignment,
    Increase,
    Decrease
};
//在此处枚举你需要的命令类型
enum CommandType
{
    Speed,
    Position
};
```

- ./src/vofa_function.c

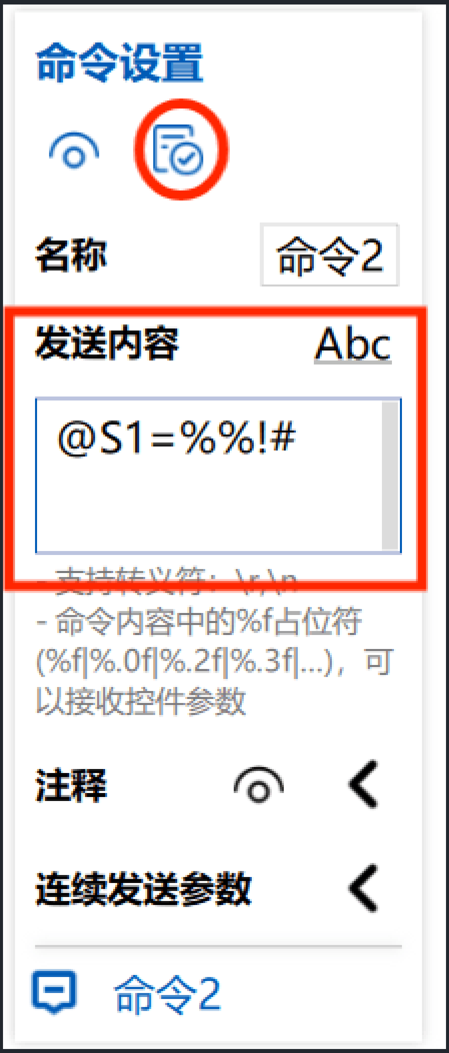
```c
void vofaCommandParse(void)
{
    uint8_t* pRxPacket;
    pRxPacket = vofaCommandData.uartRxPacket;

    //帧格式判断     此处的命令帧是以@+S/P+1/2/3+=四字节浮点数据+!+#为例
    //其中@是帧头 S/P对应命令类型 1/2/3对应命令ID !#为帧尾
    if (vofaCommandData.uartRxPacket[0] != '@' ||
vofaCommandData.uartRxPacket[3] != '=' ||
vofaCommandData.uartRxPacket[CMD_FRAME_SIZE - 2] != '!' ||
vofaCommandData.
        uartRxPacket[CMD_FRAME_SIZE - 1] != '#')
    {
        memset(vofaCommandData.uartRxPacket, 0, CMD_FRAME_SIZE);
        return;
    }

    //此处修改字节比对，改为你需要的类型
    switch (vofaCommandData.uartRxPacket[1])
    {
        case 'S': vofaCommandData.cmdType = Speed;
            break;
        case 'P': vofaCommandData.cmdType = Position;
            break;
        default: vofaCommandData.cmdType = INVALID;
            break;
    }
    //此处修改字节比对，改为你需要的ID
    switch (vofaCommandData.uartRxPacket[2])
    {
        case '1': vofaCommandData.cmdID = Direct_Assignment;
            break;
        case '2': vofaCommandData.cmdID = Increase;
            break;
        case '3': vofaCommandData.cmdID = Decrease;
            break;
        default: vofaCommandData.cmdID = INVALID;
            break;
    }
    memcpy(vofaCommandData.validData, pRxPacket + 4, 4);

    vofaCommandData.floatData = uint8Array2Float(vofaCommandData.validData, 0);

    pRxPacket = NULL;
    memset(vofaCommandData.validData, 0, 4);
    memset(vofaCommandData.uartRxPacket, 0, CMD_FRAME_SIZE);
}

```

五、VOFA命令的配置

5.1 新建命令



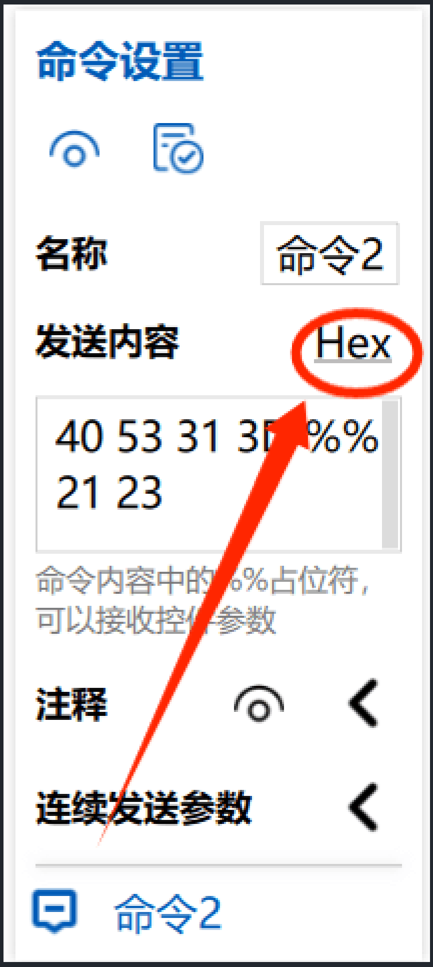
5.2 根据你的命令帧编辑命令



```
@S1=%%!#
```

- 支持转义符：\r \n
- 命令内容中的%f占位符(%f|%.0f|%.2f|%.3f|...)，可以接收控件参数

注意编辑完成后一定要切换为HEX模式保存


命令设置

名称 命令2

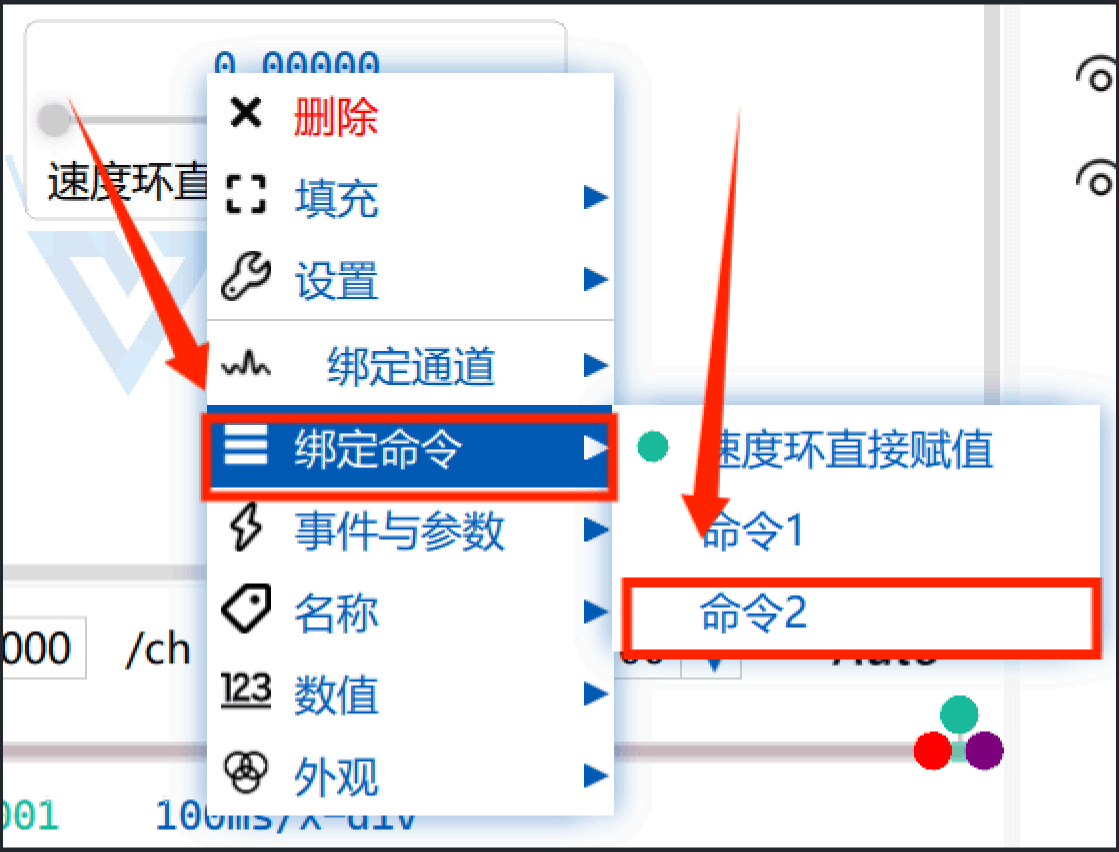
发送内容 Hex

```
40 53 31 35 %%
21 23
```

命令内容中的%%占位符，可以接收控件参数

5.3 将命令绑定到控件

在控件上右键单击，选择`绑定命令`，选择我们刚刚保存的命令即可完成绑定


0.00000
速度环直
删除
填充
设置
绑定通道
绑定命令
事件与参数
名称
数值
外观
速度环直接赋值
命令1
命令2
000
/ch
001
100ms/x-div